Sequence-to-Sequence Prediction of Vehicle Trajectory via LSTM Encoder-Decoder Architecture

Seong Hyeon Park, ByeongDo Kim, Chang Mook Kang, Chung Choo Chung and Jun Won Choi*
Hanyang University

***Abstract*— In this paper, we propose a deep learning based vehicle trajectory prediction technique which can generate the future trajectory sequence of surrounding vehicles in real time. We employ the encoder-decoder architecture which analyzes the pattern underlying in the past trajectory using the long short-term memory (LSTM) based encoder and generates the future trajectory sequence using the LSTM based decoder. This structure produces the K most likely trajectory candidates over occupancy grid map by employing the *beam search* technique which keeps the K locally best candidates from the decoder output. The experiments conducted on highway traffic scenarios show that the prediction accuracy of the proposed method is significantly higher than the conventional trajectory prediction techniques.**

I. Introduction

Ensuring safety is a top priority for the autonomous driving and advanced driver assistance systems (ADAS). In order to promise high degree of safety, the ability to perceive surrounding situations and predict their development in the future is critical. The vehicle on driving encounters various types of dynamic traffic participants such as car, motor bike, and pedestrian which could be a potential threat to safe driving. In order to avoid an accident, the system should be able to analyze the pattern in their motion and predict the future trajectories in advance. If the system predicts where the surrounding vehicles are heading in the near future, the vehicle can plan its driving path in response to the situation to come such that the probability of collision is minimized. However, the trajectory of the surrounding vehicles is quite complex to analyze since it is governed by various latent factors determined by complex traffic situations and the state of these latent factors can change dynamically in real time.

Thus far, various vehicle trajectory analysis techniques have been proposed. The traditional approaches are the Bayesian filtering methods such as Kalman and extended Kalman filters [1], [2]. However, the structure of these methods might be too simple to analyze the complicated pattern of the vehicle motion and they do not often perform well for long term prediction, e.g., $\Delta = 2$ sec. In order to overcome the limitation, more sophisticated trajectory models were introduced, including Gaussian process model [3], [4], Gaussian mixture model [5], and dynamic Bayesian network (DBN) [6]. In particular, the DBN provides a flexible trajectory analysis framework where various latent factors determining the vehicle trajectory are described using the graphical model, and the interactions among these factors are learnt from the data. Though the DBN leads to explicit modeling of physical process that generates a vehicle's trajectory, the performance for real traffic scenarios is limited in that the model structure determined by the designer's intuition is not sufficient to capture a variety of dynamic traffic scenarios. Furthermore, the computational complexity for its inference step is quite high for the real time applications.

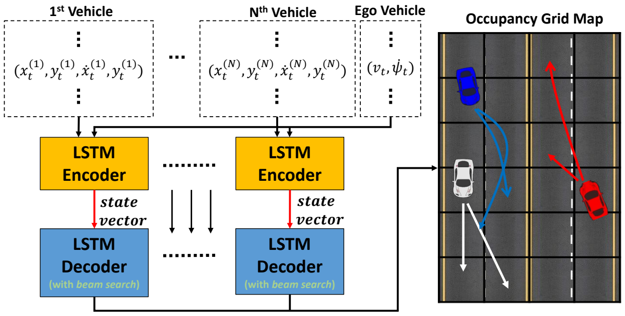


Fig. 1. The proposed trajectory prediction system. $(x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)})$ denotes the nth surrounding vehicle's relative position and velocity at time t and $(v_t, \dot{\psi}_t)$ denotes the ego vehicle's speed and yaw rate at time t.

Meanwhile, the era of deep learning has begun with the success of many revolutionary deep neural network (DNN) architectures [7]–[9]. Being applied to numerous machine learning tasks, DNN architectures have successfully learned the representation that generalizes well for various situations arising in the real data. Among various kinds of DNN architectures available, recurrent neural network (RNN) is widely used to analyze the structure of the time series data. One popular variant of RNN is *long short-term memory* (LSTM) model, which can control the information flow between the input, output, and cell memory through the gating mechanism [10]. The LSTM has shown excellent performance for various tasks such as speech recognition, image captioning, language translation and so on [11]. Recently, LSTM or similar RNN variants have also been applied to analyze the vehicle trajectory [12]–[15]. In [12], the LSTM is used to track the position of the object based on the ranging sensor measurements. In [13], the driver's intention is identified based on the trajectory data using the LSTM. In [14], the LSTM is applied to predict the location of the vehicle after Δ seconds using the past trajectory data. In [15], another RNN variant called gated recurrent unit (GRU) combined

* corresponding author
The authors are with the Department of Electrical Engineering, Hanyang University, Seoul, Korea.
{shpark,bdkim}@spo.hanyang.ac.kr
{kcm0728,cchung,junwchoi}@hanyang.ac.kr

with conditional variational auto-encoder (CVAE) is used to predict the vehicle trajectory. Although these RNN based vehicle trajectory models are effective for their own tasks, they either cannot generate the full trajectory sequence or do not provide a simple probabilistic framework that does not require extra components such as CVAE.

In this paper, we propose a new vehicle trajectory analysis and prediction technique to generate the future trajectory of surrounding vehicles given the sequence of the latest sensor measurements. The proposed method is built upon the LSTM encoder-decoder architecture which has shown excellent performance for *sequence to sequence* tasks [8]. Fig. 1 depicts the structure of the proposed technique. The LSTM encoder takes the latest trajectory samples for the surrounding vehicles as well as the state information on the ego vehicle and produces the fixed length vector which captures the temporal structure of the past trajectory. Based on the fixed length vector, the LSTM decoder generates the future trajectory on the occupancy grid map (OGM). The decoder recursively uses the predicted trajectory sample to generate the subsequent trajectory samples adopting *beam search* algorithm. With the algorithm, the decoder keeps K locally best sequence candidates in generating the future trajectory sample for each time step [16]. As a result, the proposed model can predict the K most probable hypotheses of the vehicle trajectory under the probabilistic framework. Our experiment results show that the proposed scheme generates the reasonable trajectory hypotheses of the surrounding vehicles and its prediction accuracy is significantly better than the conventional prediction methods.

The rest of this paper is organized as follows. In Section II, we briefly review the LSTM encoder-decoder architecture and the beam search algorithm. In Section III, we describe the proposed trajectory prediction model in details. In Section IV, the experimental results are presented and Section V concludes this paper.

II. Review on LSTM Encoder-Decoder Architecture

A. Long-Short Term Memory (LSTM)

The long short-term memory (LSTM) is an RNN variant which effectively overcomes the vanishing gradient issue in naively designed RNNs [10]. The LSTM consists of the cell memory that stores the summary of the past input sequence, and the gating mechanism by which the information flow between the input, output, and cell memory are controlled. The following recursive equations describe how the LSTM works;

$$\boldsymbol{f}_t = \sigma(W_{uf}\boldsymbol{u}_t + W_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \quad (1)$$

$$\boldsymbol{i}_t = \sigma(W_{ui}\boldsymbol{u}_t + W_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \quad (2)$$

$$\boldsymbol{o}_t = \sigma(W_{uo}\boldsymbol{u}_t + W_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \quad (3)$$

$$\begin{aligned}\boldsymbol{c}_t = {} & \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \\ & + \boldsymbol{i}_t \odot \tanh(W_{uc}\boldsymbol{u}_t + W_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)\end{aligned} \quad (4)$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t), \quad (5)$$

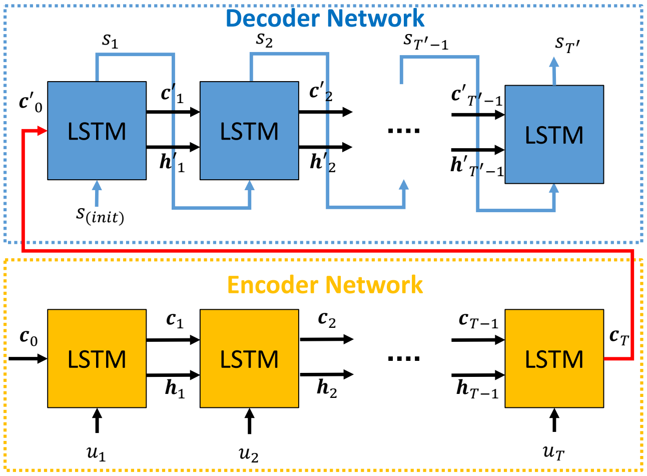


Fig. 2. The LSTM encoder-decoder architecture.

where

- $\sigma(x) \triangleq \frac{1}{1+\exp(-x)}$: sigmoid function (element-wise)
- $x \odot y$: element wise product
- $\boldsymbol{u}_t$: input vector
- $W_{ui}, W_{hi}, W_{uf}, W_{hf}, W_{uo}, W_{ho}, W_{uc}, W_{hc}$: linear transformation matrices
- $\boldsymbol{b}_i, \boldsymbol{b}_f, \boldsymbol{b}_o, \boldsymbol{b}_c$: bias vectors
- $\boldsymbol{i}_t, \boldsymbol{f}_t, \boldsymbol{o}_t$: gating vectors
- $\boldsymbol{c}_t$: cell memory state vector
- $\boldsymbol{h}_t$: state output vector.

The amount of information for the cell memory to update, forget, and output its state is determined by the gating vectors in (1), (2), and (3). The cell state and output are updated according to (4) and (5). Note that depending on the state of the forget gating vector $\boldsymbol{f}_t$, the cell state can be reset or restored, and the other two gating vectors $\boldsymbol{i}_t$ and $\boldsymbol{o}_t$ operate in the similar manner to regulate the input and output.

B. LSTM Encoder-Decoder Architecture

The LSTM encoder-decoder architecture was first introduced for machine translation task [8], [16], [17]. It has an ability to read and generate a sequence of arbitrary length as illustrated in Fig. 2. The architecture employs two LSTM networks called the encoder and decoder. The encoder processes the input sequence $u_1, ..., u_T$ of the length T and produces the summary of the past input sequence through the cell state vector $\boldsymbol{c}_t$. After T times of recursive updates from (1) through (5), the encoder summarizes the whole input sequence into the final cell state vector $\boldsymbol{c}_T$. Then, the encoder passes $\boldsymbol{c}_T$ to the decoder so that the decoder uses it as initial cell state (i.e., $\boldsymbol{c}'_0 = \boldsymbol{c}_T$) for the sequence generation. The decoding step is initiated with a dummy input $s_{(\text{init})}$. The decoder recursively generates the output sequence $s_1, ..., s_{T'}$ of the length T'. In every update, the decoder feeds the output s_{t-1} obtained in the previous update to the input for the current update. Note that the output of the decoder are derived by applying the affine transformation followed by the function that suits for the specific tasks (e.g. Softmax function for classification task).

Basically, the LSTM encoder-decoder aims to model the conditional probability of the output sequence given the input sequence, i.e., $p(s_1, ..., s_{T'}|u_1, ..., u_T)$. The encoder provides the summary of the input sequence $u_1, ..., u_T$ through the LSTM cell state $\boldsymbol{c}_T$. Given the encoder cell state $\boldsymbol{c}_T$, the conditional probability is approximated to

$$p(s_1, ..., s_{T'}|u_1, ..., u_T) \approx \prod_{t=1}^{T'} p(s_t|\boldsymbol{c}_T, s_1, ..., s_{t-1}). \quad (6)$$

The decoder successively produces the probability distribution of $p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1})$ given the decoder cell state $\boldsymbol{c}'_{t-1}$ and the $(t-1)$th sample of the output sequence s_{t-1}, i.e.,

$$p(s_1, ..., s_{T'}|u_1, ..., u_T) \approx \prod_{t=1}^{T'} p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1}). \quad (7)$$

Unfortunately, the decoder does not know the true value of the previous output sample. Hence, in every decoding step, the decoder makes a decision on s_t based on the probability distribution $p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1})$ obtained from the decoder output and use the tentative decision for the next update of the decoder state.

C. The Beam-Search Algorithm

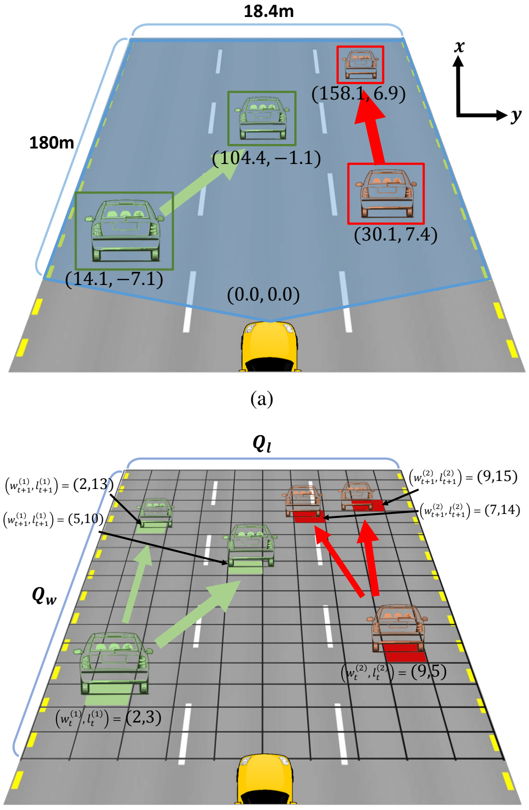
As mentioned, the LSTM decoder aims to produce the probability distribution of s_t given the decoder cell state $\boldsymbol{c}'_{t-1}$ and the $(t-1)$th output sample s_{t-1}. One way to determine s_t is the greedy search strategy that simply picks the value for s_t that maximizes the probability $p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1})$ and feed it back to the decoder to generate the next output sample. Unfortunately, such greedy strategy suffers from the error propagation since wrong decision made at the current time step would be propagated to the subsequent time steps. Furthermore, $p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1})$ might be a multi modal distribution having multiple peaks which correspond to the equally promising candidates for s_t. In this case, generating only a single hypothesis is not sufficient to represent all probable outcomes.

In order to alleviate the error propagation, the *beam search* algorithm has been introduced for the machine translation tasks [16]. The basic idea of the beam search is to keep the K most probable hypotheses for each sequence generation step where the parameter K is called *beam width*. For each of K hypothetical sequences found in the previous decoding step, the decoder generates $|S|$ candidates for s_t resulting in total $K \times |S|$ candidates where $|S|$ denotes the cardinality of the set of all possible selections for s_t. Then, it chooses the K best hypotheses according to the conditional probability $p(s_t|\boldsymbol{c}'_{t-1}, s_{t-1})$ produced at the decoder output. After T' iterations, the K best hypotheses would survive as a final result of the decoding step. Note that the beam search with $K = 1$ reduces to the greedy search algorithm.

III. Proposed Trajectory Analysis Technique

In this section, we describe the system design, the detailed network structure, and the training methodology of the proposed technique.

A. System Description


18.4m
180m
x
y
(158.1, 6.9)
(104.4, −1.1)
(30.1, 7.4)
(14.1, −7.1)
(0.0, 0.0)
Q_l
Q_w
$(w_{t+1}^{(1)}, l_{t+1}^{(1)}) = (2,13)$
$(w_{t+1}^{(1)}, l_{t+1}^{(1)}) = (5,10)$
$(w_{t+1}^{(2)}, l_{t+1}^{(2)}) = (9,15)$
$(w_{t+1}^{(2)}, l_{t+1}^{(2)}) = (7,14)$
$(w_t^{(2)}, l_t^{(2)}) = (9,5)$
$(w_t^{(1)}, l_t^{(1)}) = (2,3)$

(a)

(b)

Fig. 3. The description of (a) relative coordinate system and (b) grid representation on occupancy grid map.

Our system assumes that the ego vehicle can estimate the current state of the relative coordinate and velocity of the surrounding vehicles from the measurements acquired by the sensors (e.g. camera, lidar, and radar sensors). In addition, it can obtain the information on its own motion using the inertial measurement unit (IMU) sensor. The set of the observations used for prediction of the trajectory of the nth surrounding vehicle at the time step t is given by

$$\mathcal{O}_t^{(n)} = \{v_t, \dot{\psi}_t, x_t^{(n)}, y_t^{(n)}, \dot{x}_t^{(n)}, \dot{y}_t^{(n)}\} \quad (8)$$

where

- v_t: the ego vehicle's speed
- $\dot{\psi}_t$: the ego vehicle's yaw rate
- $\{x_t^{(n)}, y_t^{(n)}\}$: the nth vehicle's relative coordinate
- $\{\dot{x}_t^{(n)}, \dot{y}_t^{(n)}\}$: the nth vehicle's relative velocity.

As illustrated in Fig. 3 (a), we adopt the relative coordinate system where the ego vehicle's location is fixed to $(0, 0)$. The longitudinal and lateral ranges are set to be $x \in [0, 180]$ and $y \in [-9.2, 9.2]$ in meters which are determined according to the valid detection range of the sensors.

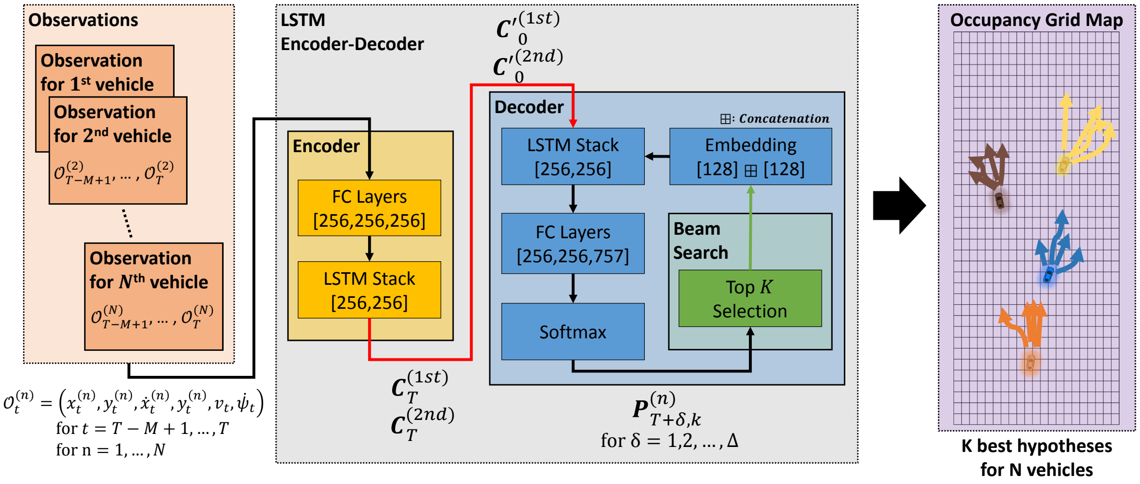

Observations
Observation for 1st vehicle
Observation for 2nd vehicle
$O^{(2)}_{T-M+1}, \dots, O^{(2)}_{T}$
Observation for Nth vehicle
$O^{(N)}_{T-M+1}, \dots, O^{(N)}_{T}$
$O^{(n)}_t = \left(x^{(n)}_t, y^{(n)}_t, \dot{x}^{(n)}_t, \dot{y}^{(n)}_t, v_t, \dot{\psi}_t\right)$
for $t = T - M + 1, \dots, T$
for $n = 1, \dots, N$
LSTM Encoder-Decoder
$C'^{(1st)}_0$
$C'^{(2nd)}_0$
Encoder
FC Layers [256,256,256]
LSTM Stack [256,256]
$C^{(1st)}_T$
$C^{(2nd)}_T$
Decoder
⊞: Concatenation
LSTM Stack [256,256]
Embedding [128] ⊞ [128]
FC Layers [256,256,757]
Beam Search
Top K Selection
Softmax
$P^{(n)}_{T+\delta,k}$
for $\delta = 1,2,\dots,\Delta$
Occupancy Grid Map
K best hypotheses for N vehicles

Fig. 4. The overall structure of the proposed trajectory analysis technique.

In order to represent the future trajectory of the surrounding vehicles in the proposed system, we use the occupancy grid map (OGM) which has been widely used in robotics for the object localization [18]. The OGM divides the region around the ego vehicle into $Q_w \times Q_l$ rectangular grid elements. As shown in Fig. 3 (b), we represent each element of OGM by the grid index. In the proposed system, we use (36×21) grids, where each grid span 5.0 meter by 0.875 meter region in longitudinal and lateral directions, respectively. This design is adopted in order for a single grid to cover the length of a typical sedan and the quarter lane width. Using the OGM, the trajectory prediction task becomes a classification problem with $Q = Q_w \times Q_l + 1$ classes, each of which corresponds to one of the $Q_w \times Q_l$ grid elements on the OGM or the out-of-map state. We formulate the vehicle trajectory prediction as a sequential multiclass classification problem where the grid element occupied by a surrounding vehicle should be chosen sequentially for each time step.

The overall workflow of the system is illustrated in Fig. 4 and described as follows. For the nth surrounding vehicle, the sequence of the latest M observations $\mathcal{O}^{(n)}_{T-M+1}, \dots, \mathcal{O}^{(n)}_{T}$ is fed into the encoder. Given the latest cell memory produced by the encoder through M updates, the decoder sequentially generates the future trajectory for Δ time steps ahead. Note that using the beam search, the decoder generates the K most likely trajectory sequences in parallel. At the $(T+\delta)$th time step ($\delta \in \{1, \dots, \Delta\}$), the decoder calculates the probability $P^{(n)}_{T+\delta,k,q}$ that the nth vehicle occupies the OGM element indexed by $q \in \{1, \dots, Q\}$. This constructs the probability map of size Q i.e., $\boldsymbol{P}^{(n)}_{T+\delta,k} = \{P^{(n)}_{T+\delta,k,1}, \dots, P^{(n)}_{T+\delta,k,Q}\}$. Since this decoding process is performed for all survived K trajectory sequences, we come up with K probability maps $\boldsymbol{P}^{(n)}_{T+\delta,1}, \dots, \boldsymbol{P}^{(n)}_{T+\delta,K}$. Then, based on the K probability maps, we pick the K best trajectory candidates which yield the largest probability of occupancy. Note that these candidates are fed back to the LSTM decoder through the data embedding process. Repeating the above process with Δ updates, the decoder ends up with the final K best trajectory sequences for the nth surrounding vehicle. Sharing the parameters of the encoder and decoder for all surrounding vehicles, we can collect the results of trajectory prediction for N surrounding vehicles on the single OGM. A total of $K \times N$ future trajectory hypotheses shown on the single OGM presents the unified view on how the state of the dynamic objects around the ego vehicle would develop in Δ time steps ahead.

B. Network Structure

1) Encoder: The encoder consists of three fully connected (FC) layers followed by two LSTM layers stacked. Each FC layer contains affine transformation followed by ReLU (Rectified Linear Unit) activation function. They are designed for two purposes; 1) transforming the 6 dimensional input data into 256 dimensional feature being consistent with the LSTM cell dimension and 2) extending the network capacity enough to capture the complex structure of the trajectory data. The output from the last FC layer is then fed into the LSTM stack. The LSTM stack is constructed with two LSTMs with 256 dimensional cell memory for each. Inside the stack, the output vector from the the first LSTM is fed to the second LSTM as an input. After M recursive updates in the two LSTMs, their latest cell states $\boldsymbol{C}^{(1\text{st})}_T$ and $\boldsymbol{C}^{(2\text{nd})}_T$ are determined and passed to the decoder.

2) Decoder: The decoder consists of the two LSTM layers stacked followed by three FC layers, the Softmax layer, and the embedding layer. The decoder LSTMs use the cell state vectors passed from the encoder as their initial cell states. The first LSTM of the decoder is initialized by $\boldsymbol{C}^{(1\text{st})}_T$ and the second decoder LSTM is by $\boldsymbol{C}^{(2\text{nd})}_T$. The output from the second LSTM is fed to the FC layers. While the first and second FC layers have the same structure as those in the encoder, the last FC layer has the output dimension of 757 corresponding to the number of class (i.e., $Q = 36 \times 21 + 1$). Then, the output of the FC layers is passed through the Softmax function, which produces the probability of the occupancy on the probability map of the size 757. Based on all K probability maps derived for all hypothesis, most probable K trajectory candidates are selected by the beam search algorithm. The selected K OGM indices are fed back to the decoder through the embedding process. In the embedding step, we employ two embedding matrices of size 128×37 and 128×22 in which each column vector corresponds to the longitudinal ($w_t^i \in \{1, ..., 36\}$) and lateral ($l_t^i \in \{1, ..., 21\}$) grid indices on the OGM as well as the out-of-map state. Among these column vectors, K vectors are gathered from each embedding matrix, that is, $2 \times K$ vectors in total according to the selected longitudinal and lateral OGM indices. Then, they are concatenated into 256 dimensional vectors where each of them represents the selected OGM indices and delivered to the decoder LSTM to be used in the next decoding step. This procedure is repeated until we reach the prediction length of Δ.

C. Training Methodology

The parameters of the LSTM encoder-decoder (including the embedding matrices) are trained in an end to end fashion. We generate the training data set by cropping all available $(M + \Delta)$ length trajectory samples from the trajectory record for each surrounding vehicles. Assuming that the trajectory data collected from the sensors $\mathcal{O}_{T-M+1}^{(n)}, ..., \mathcal{O}_{T+\Delta}^{(n)}$ is sufficiently good in terms of accuracy, the OGM grid indices extracted from the subsequent Δ measurements (i.e., $\mathcal{O}_{T+1}^{(n)}, ..., \mathcal{O}_{T+\Delta}^{(n)}$) can be used as the label for the supervised training. The loss function to be minimized is given by the negative log likelihood function

$$L(\theta) = -\sum_{j=1}^{J}\sum_{\delta=1}^{\Delta}\sum_{q=1}^{Q}\Big(o_{j,T+\delta,q}\ln z_{j,T+\delta,q} + (1 - o_{j,T+\delta,q})\ln(1 - z_{j,T+\delta,q})\Big) \quad (9)$$

where J is the total number of the training samples, Δ is the prediction length, and Q is the total number of classes. The variable $o_{j,T+\delta,q}$ is the OGM grid label represented in the one hot encoding. For the jth example, the label $o_{j,T+\delta,q}$ becomes one if the surrounding vehicle occupies the qth grid element and zero otherwise. Note also that $z_{j,T+\delta,q}$ is the qth output of the Softmax layer in the LSTM decoder for the jth example. For the minimization of the loss function, we adopt the stochastic gradient decent method with a momentum, called *ADAM* optimizer [19] with mini batch size B. The training is stopped if the validation error (obtained from 15% of the training data) does not decrease anymore.

IV. Experiments

In this section, we present the performance of the proposed trajectory prediction technique based on the experiments conducted on real highway driving.

A. Data Collection

We collected the large set of vehicle trajectory data from several hours of highway driving around Seoul, Korea. The test vehicle was Hyundai Genesis equipped with Delphi long range front radars (Fig. 5). The sampling rate for the data was set to 10ms during the data collection, but the raw data was too noisy and often imbued with cut-offs because of asynchronous sampling. To cope with this problem, we averaged the samples over 100ms duration and thus the update period of the final data became 100ms. As a result, there were total 1325 trajectory sequences recorded from 26 different highway scenarios including lane changes, cut-in, and merging at the junction. From the training dataset, we used 85% (1126 sequences) for the training and 15% (199 sequences) for validation.

Fig. 5. The test vehicle.

B. Experiment Setup

In our experiments, the maximum prediction range is set to $\Delta = 20$ (corresponding to 2 seconds). The initial learning rate is set to 0.0008 and halved whenever the validation error is plateaued. The mini batch size B is set to 256. Through intensive empirical optimization, we determine the hyperparameters of the LSTM encoder-decoder network as

- The depth of fully connected layers: 3
- The LSTM cell state dimension: 256
- The depth of LSTM stack: 2
- The beam width K: 10
- The observation length M: 30

Predicting trajectory sample every 0.1s requires 20 decoder updates to reach the prediction horizon of 2 sec. We found that this is not desirable for the accuracy of long term prediction and it leads to high computational cost for inference. Hence, we trained the decoder to generate the trajectory sample every 0.2 sec. This allows the proposed system to reach the prediction horizon of 2 sec only with 10 updates.

For the performance evaluation, we use the Top-Ω mean absolute error (MAE) as a performance metric. Given the top Ω trajectory candidates among the trajectories produced by the proposed scheme, the Top-Ω MAE is obtained by finding the trajectory which is closest to the ground truth and evaluating the absolute error between them, i.e.,

$$\text{Top-}\Omega\ \text{MAE} = \frac{1}{L}\sum_{i=1}^{L}\left\|\begin{bmatrix} w_i^* \\ l_i^* \end{bmatrix} - \begin{bmatrix} w_i^{(gt)} \\ l_i^{(gt)} \end{bmatrix}\right\|_2 \quad (10)$$

where L is the number of the test examples and (w_i^*, l_i^*) is the candidate grid index closest to the ground truth and $(w_i^{(gt)}, l_i^{(gt)})$ is the corresponding ground truth grid index, respectively. This metric explains how well the ground truth trajectory can be predicted by one of top Ω trajectory candidates. Note that the Top-Ω MAE is measured in the unit of the grid on the OGM. We can calculate the Top-Ω MAE

TABLE I

MAE, MAE_X, AND MAE_Y OF SEVERAL TRAJECTORY PREDICTION ALGORITHMS

MAE (Grids)

	Proposed	**Proposed**	**Proposed**	**Baseline 1**	**Baseline 2**	**Baseline 3**
Δ	$(\Omega = 1)$	$(\Omega = 3)$	$(\Omega = 5)$	**(Kalman filter)**	**(Basic LSTM)**	**[14]**
0.4s	0.64	0.46	0.41	1.52	N/A	N/A
0.8s	0.84	0.66	0.59	2.79	N/A	N/A
1.2s	0.99	0.79	0.73	3.92	N/A	N/A
1.6s	1.14	0.95	0.83	5.27	N/A	N/A
2.0s	1.27	1.02	0.93	6.36	1.93	1.31

MAE_X (Grids)

	Proposed	**Proposed**	**Proposed**	**Baseline 1**	**Baseline 2**	**Baseline 3**
Δ	$(\Omega = 1)$	$(\Omega = 3)$	$(\Omega = 5)$	**(Kalman filter)**	**(Basic LSTM)**	**[14]**
0.4s	0.24	0.15	0.14	0.54	N/A	N/A
0.8s	0.30	0.20	0.18	1.17	N/A	N/A
1.2s	0.36	0.25	0.23	1.38	N/A	N/A
1.6s	0.43	0.33	0.27	1.80	N/A	N/A
2.0s	0.50	0.35	0.32	2.07	0.96	0.44

MAE_Y (Grids)

	Proposed	**Proposed**	**Proposed**	**Baseline 1**	**Baseline 2**	**Baseline 3**
Δ	$(\Omega = 1)$	$(\Omega = 3)$	$(\Omega = 5)$	**(Kalman filter)**	**(Basic LSTM)**	**[14]**
0.4s	0.45	0.34	0.31	1.29	N/A	N/A
0.8s	0.63	0.52	0.46	2.35	N/A	N/A
1.2s	0.75	0.63	0.58	3.51	N/A	N/A
1.6s	0.87	0.75	0.67	4.78	N/A	N/A
2.0s	0.95	0.81	0.73	5.84	1.34	1.06

for each future time step $\delta \in \{1, ..., \Delta\}$, which is denoted as "Top-Ω MAE(δ)". We can measure the error in longitudinal and lateral directions using Top-Ω MAE_X(δ) and Top-Ω MAE_Y(δ), respectively.

C. Experiment Results

We first present the MAE performance of the proposed prediction method. For performance comparison, we compare our method with the conventional Kalman filter based on the constant velocity model. We also consider the basic LSTM model trained to predict the probability of the occupancy over OGM and more complex LSTM based prediction method proposed in [14]. We also present the performance of the proposed algorithm with $\Omega = 1, 3$ and 5. Table I presents the MAE, MAE_X, and MAE_Y achieved by the trajectory prediction algorithms of interest for the prediction time steps of $\Delta = 0.4, 0.8, 1.2, 1.6$ and 2.0 sec. Since Kalman filter can also generate the future sequence by repeating the prediction update steps, we provide the MAEs for all time steps of Δ. On the contrary, the LSTM model and the prediction scheme [14] only predict the future location of the target vehicle after particular time ahead (e.g., 2.0 sec). Hence, we provide the MAEs corresponding to $\Delta = 2.0$ sec only. We see that the prediction accuracy for all schemes of interest decreases with Δ since it is more difficult to predict the distant future. We observe that the proposed algorithm with $\Omega = 1$ significantly outperforms both the Kalman filter and the basic LSTM model. With $\Omega = 1$, the proposed scheme achieves the performance comparable to the baseline [14]. But with $\Omega = 3$ and 5, the performance of the proposed scheme exceeds that of the baseline [14] since our method generates more trajectory samples which are strongly likely to appear. Note that as compared to the baseline [14], the proposed scheme has the advantage of generating the full sequences of the promising trajectory candidates in each time step. The above mentioned trend appears for all MAE, MAE_X, and MAE_Y results provided in Table I.

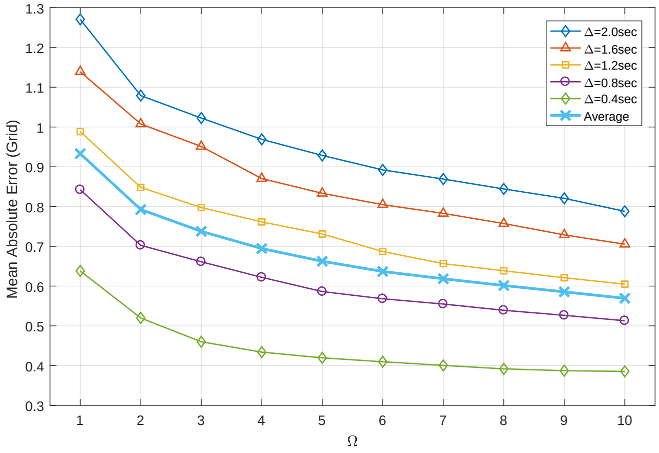


Fig. 6. MAE versus Ω of the proposed method

Next, we investigate how the proposed method behaves with respect to the parameters Δ and Ω. Fig. 6 shows the plot of MAE as a function of Ω for several values of Δ. We also include the MAE averaged over all values of Δ considered. As Ω increases, the proposed system generates more trajectory hypotheses and the MAE performance improves for all cases. Note that the performance improvement due to increasing Ω is more dramatic for the case of higher Δ, yielding the decrease of MAE by around 0.2 grid for the change from $\Omega = 1$ to

3. This implies that including more trajectory hypotheses increases the probability that one of the trajectory candidates found by our algorithm is close to the ground truth trajectory. Therefore, the ability of the proposed method to generate the multiple trajectories would provide more informative and reliable prediction results to the subsequent path planning and control steps for autonomous driving.

V. Conclusions

In this paper, we proposed the new vehicle trajectory prediction method based on the LSTM encoder-decoder neural network architecture. The proposed system employs the LSTM encoder to analyze the past sensor measurements and the LSTM decoder to generate the future trajectory samples based on the encoder output. In order to alleviate the error propagation issue appearing in iterative decoding step, we applied the beam search algorithm, which keeps the K best trajectory candidates for each decoding iteration. As a result, the proposed method can generate multiple trajectory hypotheses that might develop in many different ways given the same previous situation. Our experiment results confirmed that the proposed method can achieve the significant improvement over the existing methods in terms of the prediction accuracy while being able to generate the full sequence of the predicted trajectory in one shot.

ACKNOWLEDGMENT

This work was supported by the Technology Innovation Program(10083646) funded By the Ministry of Trade, Industry & Energy(MOTIE, Korea) and Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIT) (No. R7117-16-0164, Development of wide area driving environment awareness and cooperative driving technology which are based on V2X wireless communication)

References

[1] S. Ammoun and F. Nashashibi, "Real time trajectory prediction for collision risk estimation between vehicles," in *IEEE Inter. Conf. on Intel. Computer Commun. and Proc. (ICCP)*, 2009, pp. 417–422.

[2] C. Barrios and Y. Motai, "Improving estimation of vehicle's trajectory using the latest global positioning system with kalman filtering," *IEEE Trans. on Instru. and Meas.*, vol. 60, no. 12, pp. 3747–3755, 2011.

[3] Q. Tran and J. Firl, "Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression," in *IEEE Proc. Intel. Vehicles Symp. (IV)*, 2014, pp. 918–923.

[4] C. Laugier, I. E. Paromtchik, M. Perrollaz, M. Yong, J.-D. Yoder, C. Tay, K. Mekhnacha, and A. Nègre, "Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety," *IEEE Intel. Transport. Syst. Magazine*, vol. 3, no. 4, pp. 4–19, 2011.

[5] J. Wiest, M. Höffken, U. Kreßel, and K. Dietmayer, "Probabilistic trajectory prediction with gaussian mixture models," in *IEEE Intel. Vehicles Symp. (IV)*, 2012, pp. 141–146.

[6] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intel. Transport. Syst. Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Proc. Syst.*, 2012, pp. 1097–1105.

[8] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proc. of 2014 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, Oct. 2014, pp. 1724–1734.

[9] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[11] A. Karpathy, "The unreasonable effectiveness of recurrent neural networks," *http://karpathy.github.io/2015/05/21/rnn-effectiveness*, 2015.

[12] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proc. AAAI Conf. on Artificial Intelligence*, 2016, pp. 3361–3367.

[13] A. Khosroshahi, E. Ohn-Bar, and M. M. Trivedi, "Surround vehicles trajectory analysis with recurrent neural networks," in *IEEE Conf. on Intel. Transport. Syst. (ITSC)*, 2016, pp. 2267–2272.

[14] B. Kim, C. M. Kang, S. H. Lee, H. Chae, J. Kim, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," *arXiv preprint arXiv:1704.07049*, 2017.

[15] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: distant future prediction in dynamic scenes with interacting agents," in *IEEE Conf. on Computer Vision and Pattern Recog. (CVPR)*, 2017, pp. 2165–2174.

[16] G. Neubig, "Neural machine translation and sequence-to-sequence models: A tutorial," *arXiv preprint arXiv:1703.01619*, p. 31, 2017.

[17] M. Luong, E. Brevdo, and R. Zhao, "Neural machine translation (seq2seq) tutorial," *https://github.com/tensorflow/nmt*, 2017.

[18] A. Milstein, "Occupancy grid maps for localization and mapping," in *Motion Planning*. InTech, 2008.

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. of 2015 Inter. Conf. on Learning Representations (ICLR)*, 2014.