View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Heriot Watt Pure

Interacting Vehicle Trajectory Prediction with Convolutional Recurrent Neural Networks

Saptarshi Mukherjee, Sen Wang and Andrew Wallace

***Abstract*— Anticipating the future trajectories of surrounding vehicles is a crucial and challenging task in path planning for autonomy. We propose a novel Convolutional Long Short Term Memory (Conv-LSTM) based neural network architecture to predict the future positions of cars using several seconds of historical driving observations. This consists of three modules: 1) Interaction Learning to capture the effect of surrounding cars, 2) Temporal Learning to identify the dependency on past movements and 3) Motion Learning to convert the extracted features from these two modules into future positions. To continuously achieve accurate prediction, we introduce a novel feedback scheme where the current predicted positions of each car are leveraged to update future motion, encapsulating the effect of the surrounding cars. Experiments on two public datasets demonstrate that the proposed methodology can match or outperform the state-of-the-art methods for long-term trajectory prediction.**

I. Introduction

Self-drive vehicle technology has the potential to improve safety, reduce congestion, improve fuel efficiency and provide increased mobility [1]. Despite considerable progress [2], significant technical improvements are needed to achieve unrestricted level 5 operation [3]. The verifiable interpretation of sensor data to predict the trajectories of another vehicle is challenging because the future trajectory is the outcome of a set of decisions made by a human driver, subject to previous experience, training and external factors. Such prediction is difficult as human behaviour is very diverse.

Common tracking and predictive algorithms that use simple kinematic models, e.g. Kalman [4] or particle filters [5], do not encode either the interaction with other vehicles or with road infrastructure, which is essential for long term trajectory prediction. Although various interaction-based models have been developed, such as the Modified Social Force Model (MSFM) [6] and the n-body collision avoidance scheme [7], modelling these interacting force fields between individual agents in a realistic fashion is very challenging. To avoid the development of complicated force-based models, various learning-based approaches using, for example, a Support Vector Machine (SVM) [8], a Hidden Markov Model (HMM) [9] or artificial neural networks (ANNs), have been proposed for both lane change prediction on a highway [10] and turn prediction at a junction [11].

Recently, the trajectory prediction problem has been addressed as a sequence-to-sequence problem based on an encoder-decoder architecture [12], [13], [14], [15], adding the surrounding cars in the input sequence to encode interaction. However, it is difficult to consider all the surrounding cars in the decoder as this is designed only to learn future movement from the encoded features. Therefore, the interaction predicted in the future horizon may not be fully exploited.

To avoid this problem, in grid-based approaches a sequence of occupancy maps is provided to the model which is trained to predict the future map sequence [15], [14]. Two major problems with this approach are car association and loss function design. First, associating each predicted car's position with its corresponding ground-truth position can be difficult when there are multiple cars close to each other. Therefore, the positions predicted by the network during training may be wrongly assigned, leading to a wrongly computed loss function. Second, the common loss function used in these cases is the pixel to pixel between the predicted and ground truth occupancy map which can be misguiding in a scenario where more than one car is moving at the same speed.

In this paper, a new Convolutional Long Short-Term Memory (Conv-LSTM) architecture is proposed to address these challenges. The behaviour prediction problem is formulated as a sequence-input-single-output (SeqToOne) structure instead of a sequence-input-sequence-output (SeqToSeq) model. Feedback is also introduced to incorporate other cars' movements for the next prediction. The major contributions are:

- We design a novel architecture including Conv-LSTM [16] layers which captures spatial and temporal movement simultaneously for future movement prediction, since these two factors are highly co-dependent.
- To solve the vehicle association problem efficiently, separate Occupancy Grid Maps (OGM) are generated with respect to each target vehicle.
- To incorporate the social effect of surrounding vehicles' movement on a target vehicle's future motion, we propose a feedback mechanism to update the input OGMs after each prediction using the current predicted positions of all the surrounding cars.

II. Problem Formulation

A. Vehicle Trajectory Representation

A vehicle trajectory is represented as a sequence of points, and each is associated with a unique time instance $[(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)]$ where n is the length of the sequence and (x_n, y_n) is the co-ordinate position at time n.

The authors are with the School of Engineering and Physical Sciences at Heriot-Watt University (email: {sm236, s.wang, a.m.wallace}@hw.ac.uk).

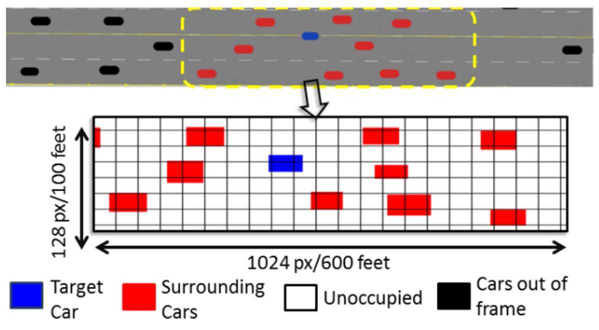


Fig. 1: Generated occupancy grid map from the traffic scene with respect to a specific target car.

For N vehicles in a scene, their trajectories T is represented as a 2D matrix below, where the ith row holds the trajectory of the ith vehicle:

$$T = \begin{bmatrix} (x_1, y_1)^1 & (x_2, y_2)^1 & \dots & (x_n, y_n)^1 \\ \dots & \dots & \dots & \dots \\ (x_1, y_1)^N & (x_2, y_2)^N & \dots & (x_n, y_n)^N \end{bmatrix}$$

Our goal is to predict the future trajectory T_{pred} of all vehicles for the next F time instances given T. T_{pred} can be given by

$$\begin{bmatrix} (x_{n+1}, y_{n+1})^1 & (x_{n+2}, y_{n+2})^1 & \dots & (x_{n+F}, y_{n+F})^1 \\ (x_{n+1}, y_{n+1})^2 & (x_{n+2}, y_{n+2})^2 & \dots & (x_{n+F}, y_{n+F})^2 \\ \dots & \dots & \dots & \dots \\ (x_{n+1}, y_{n+1})^N & (x_{n+2}, y_{n+2})^N & \dots & (x_{n+F}, y_{n+F})^N \end{bmatrix}$$

B. Spatio-Temporal Scene Representation Using OGM

Generally, within an Occupancy Grid Map (OGM), each cell value indicates the probability that cell is occupied. In our case, since the positions and trajectory information T are labelled manually we considered a binary cell which means that cells are either occupied ('1') or vacant ('0') as shown in Fig. 1. Thus a single OGM will hold the spatial relationship between the target and surrounding cars and a sequence of these OGMs will hold the temporal information. The joint spatio-temporal features can be represented as a 3D matrix $\mathbf{O} \in R^{(R\times C\times K)} \,\forall R, C, K > 0$ where R, C are the number of rows, columns of each OGM and K is the number of frames within the observation sequence. Separate observation sequences are created for each vehicle in the scene considering that particular vehicle as target vehicle. $\mathbf{O}^c_{t-K,t} \,\forall\, c = 1, 2, ..., N$ denotes a sample observation sequence created with position information from $t - K$ to t considering c as the target vehicle, where t is the current time instance. The origin of each observation sequence will be the position of the target vehicle at time t. To keep the target car and surrounding cars separate in each OGM we use two separate channels, where channel one and two keep the target car and surrounding car information respectively.

III. Conv-LSTM Based Future Trajectory Prediction

Recurrent neural networks are designed for sequential problems, specifically when dealing with temporal information. However vanilla RNN [17] and LSTM [18], [19] networks are incapable of using both the temporal and

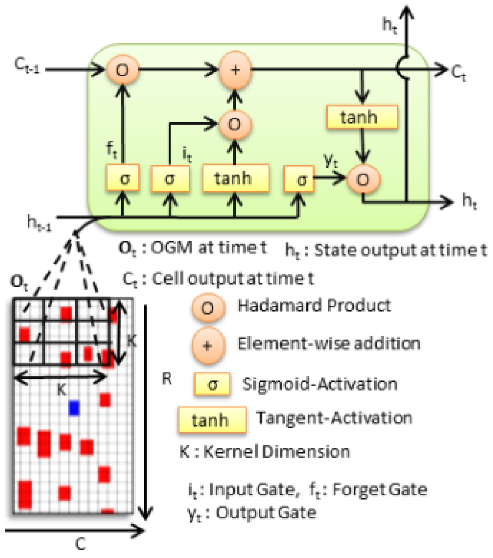


Fig. 2: Conv-LSTM Cell structure.

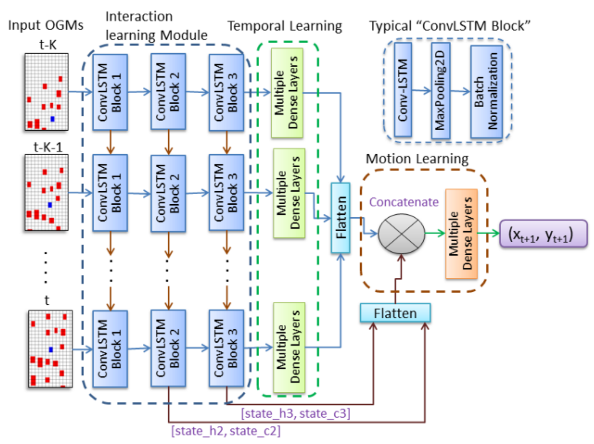


Fig. 3: Proposed Conv-LSTM based architecture.

spatial information simultaneously. To address this problem we propose a Conv-LSTM deep network architecture [16]. As shown in Fig. 2, a Conv-LSTM layer works in a similar fashion to that of a Vanilla-LSTM (V-LSTM) except the inner representations and input are both two-dimensional (provided the input image is a single channel). This enables the model to capture both temporal and spatial correlations at the same time. This layer can be further formulated as,

$$i_t = \sigma(W_{oi} * \mathbf{O}_t + W_{hi} * h_{t-1} + b_i) \quad (1)$$

$$f_t = \sigma(W_{of} * \mathbf{O}_t + W_{hf} * h_{t-1} + b_f) \quad (2)$$

$$y_t = \sigma(W_{oy} * \mathbf{O}_t + W_{hy} * h_{t-1} + b_y) \quad (3)$$

$$h_t = y_t \circ \tanh(C_t) \quad (4)$$

$$C_t = f_t \circ C_{t-1} + i_t \circ \tanh(W_{oc} * \mathbf{O}_t + W_{hc} * h_{t-1} + b_c) \quad (5)$$

where W_{oi}, W_{of}, W_{oy} and W_{oc} are input weights, W_{hi}, W_{hf}, W_{hy}, W_{hc} are previous state weights, b_i, b_f, b_y, b_c are biases and $*$ denotes convolution.

A. Proposed Network Architecture

The proposed network architecture consists of three modules, Interaction Learning, Temporal Learning and Motion Learning. It is shown in Fig. 3.

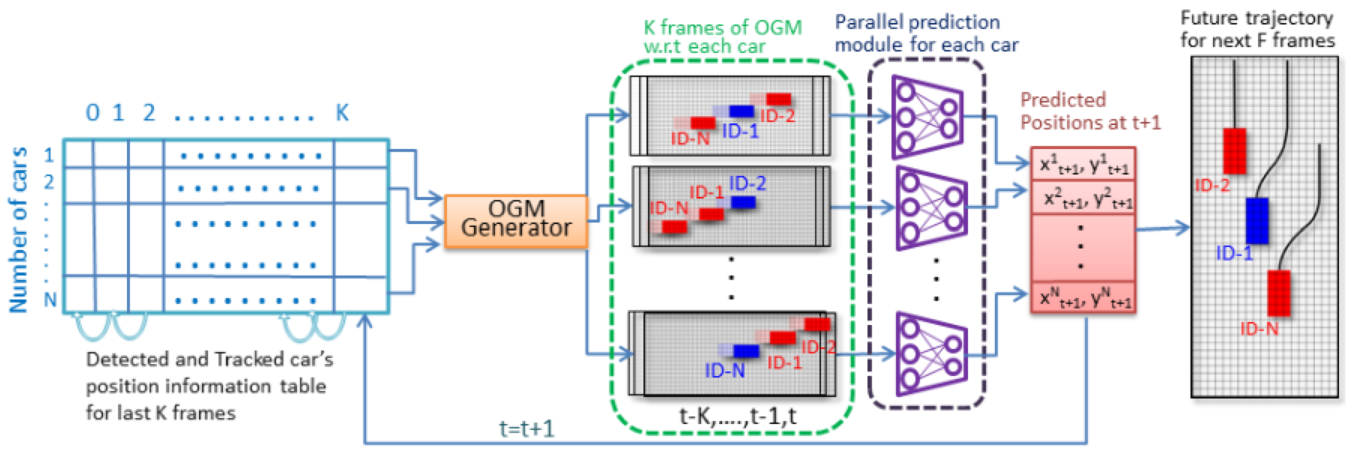

0 1 2 .... K
Number of cars
1
2
N
Detected and Tracked car's position information table for last K frames
OGM Generator
K frames of OGM w.r.t each car
ID-N
ID-1
ID-2
t-K,....,t-1,t
Parallel prediction module for each car
Predicted Positions at t+1
x^1_{t+1}, y^1_{t+1}
x^2_{t+1}, y^2_{t+1}
x^N_{t+1}, y^N_{t+1}
Future trajectory for next F frames
t=t+1

Fig. 4: Proposed feedback prediction scheme. The vehicle position table (left, blue) holds the historical position in the last K frames for each car in the scene. The orange OGM Generator block is responsible for N observation sequence generation considering each vehicle as the target vehicle once. The N violet Prediction blocks predict the position (x_{t+1}, y_{t+1}) in the next frame. Finally, the predicted positions are stored (the last red box) and fed back to the position information table (first blue table) to update the current scene simultaneously.

1) Interaction Learning: The Interaction Learning module learns the dependency between the movement of target vehicle c and its surrounding vehicles. It takes $\mathbf{O}^c_{t-K,t}$ as input. As shown in Fig. 3, this part has three "ConvLSTM Blocks", each of which is composed of a 2D Conv-LSTM layer followed by MaxPooling and BatchNormalization layers. The kernel dimensions in the three different Conv-LSTM layers are (3,3), (5,5) and (7,7). We choose a small dimension in the first layer to successfully extract the relative car positions. In the following two layers, we use two different kernel dimensions to separate the dependency of near and distant cars on the target car's future movement. To maintain the OGM dimension, the same padding is employed. The filter counts for the three Conv-LSTM layers from the beginning are 2, 4 and 8, respectively, with LeakyReLU as the activation function to avoid neuron death.

2) Temporal Learning: Recent frames within the historical observation sequence are more informative than older frames for future motion estimation. To capture this varied contribution of individual frames, we maintain a set of fully connected layers for temporal instances. Extracted features from each temporal Conv-LSTM slice are fed directly to the fully connected layers. The sizes of the layers in this module are $128 \rightarrow 64 \rightarrow 32 \rightarrow 16$ with LeakyReLU as the activation function.

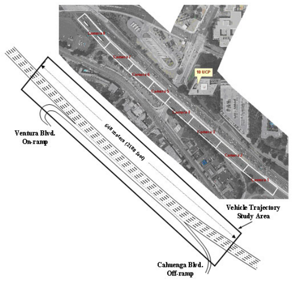
3) Motion Learning: The last part of the model is responsible for motion learning based on the extracted features from the previous two parts. All the temporal slices were concatenated with the extracted state from the Conv-LSTM layers and fed through fully connected layers to learn the correlation between previously extracted features and future movement in the next frame. The sizes of the layers are $128 \rightarrow 64 \rightarrow 32 \rightarrow 16 \rightarrow 8 \rightarrow 2$ with LeakyReLU as the activation function. The output of the network is the 2D position of the target car in the next frame.

B. Feedback Based Prediction Technique

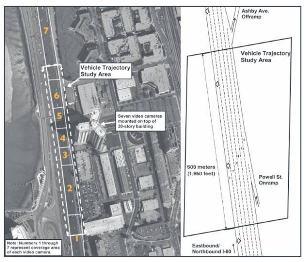
As mentioned earlier, the model is trained with the last K OGM frames as input and the position of target car in the next frame as output. To update the positions of the target and surrounding vehicles in the observation sequence after each instance prediction, a feedback based scheme shown in Fig. 4 is introduced.

The first part is the vehicle past position table of shape (N,K) which holds the positions of each car for the previous frames from $t-K$ to t from trajectories T. Whenever there is a "birth" or the appearance of a new car in the scene, it appends its location at the end of the table. Similarly whenever there is a "death" or a car goes out of the scene, the row associated with that car is removed. This vehicle position table is then passed to the OGM Generator block which creates N observation sequences considering each vehicle as a target vehicle once. Each observation sequence is then passed through the pre-trained model to predict the next position (x_{t+1}, y_{t+1}) at $t+1$. Once all the vehicle's future positions are predicted, they are concatenated to represent the full set prediction at $t+1$. This updates the vehicle position table by replacing the oldest frame. After the first prediction the vehicle position table holds position information from $t-K+1$ to $t+1$ for each vehicle. Once new positions are appended to the vehicle position table, their OGMs are generated through the OGM generator block. Similarly, after the second prediction, the position table has the observation from $t-K+2$ to $t+2$. The process is repeated until it reaches the maximum prediction horizon F.

During future movement prediction of one particular vehicle at $t+f, \forall f = 2, 3, ..., F$ where f is the current prediction instance, the predicted positions of all the other cars between $t+1$ and $t+f-1$ are also considered in the observation sequence. Hence, this feedback updates the input sequence repeatedly to incorporate potential future social interaction of other cars more effectively.



(a) US-101 Highway



(b) I-80 Highway

Fig. 5: NGSIM Train and Test scenario. Each sub figure consists of satellite view of the observation area and schematic diagram with lane counts.

IV. Implementation Details

We now discuss data formatting, training, and validation on the publicly available datasets used to validate our model.

A. Datasets

1) NGSIM Dataset: To establish the universality of the proposed model we used the Next Generation Simulation (NGSIM) US-101 [20] and I-80 [21] data collected and published by the US Federal Highway Administration. US-101 covers a 640 meters (2,100 feet) long, 5 lane wide (3.66m or 12ft each) section on Hollywood Freeway in Los Angeles (see Fig. 5(a)) and I-80 covers a 503 meters (1,650 feet) long, 6 lane wide (3.66m or 12ft each) section on I-80 freeway in Emeryville, California (see Fig. 5(b)). These separate scenarios allow us to train and test the model on two completely different datasets. Each consists of more than 5000 trajectories of real traffic captured at 10Hz for 45 minutes with three different traffic conditions, i.e. mild, moderate and congested. Each data point in the trajectory information includes lateral and longitudinal position, instantaneous speed, acceleration, vehicle length, width and vehicle type. The lateral and longitudinal positions are predicted with respect to the local coordinate system.

2) High-D Dataset: To evaluate the model performance further, the trained models are tested on a different dataset, High-D [22]. High-D is a naturalistic vehicle trajectory dataset including more than 110500 vehicles collected on 6 different highways with different lane numbers and speed limits using an unmanned aerial vehicle. State-of-the-art computer vision algorithms were used to detect and localize each vehicle. The generated trajectories were further smoothed using Bayesian smoothing. The information associated with each trajectory is similar to NGSIM dataset.

B. Data Formatting

The pixel dimensions of the OGM considered in our work are 1024 rows in the longitudinal and 128 columns in the lateral direction with 3 ($K = 30$ frames) and 5 ($F = 50$ frames) seconds as the observation and prediction time windows, respectively. The physical dimensions of the grid map are 600 feet ($\sim$180 m) by 100 feet ($\sim$30 m). In these datasets, the highest vehicle velocity recorded is roughly 75 feet/sec. In the lateral direction, a 100 feet wide grid map allows us to accommodate the adjacent lanes along with the target car's ego-lane with some extra margin on both sides.

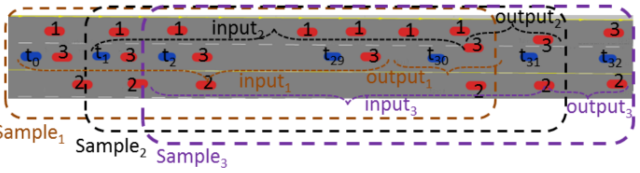


Fig. 6: Sliding window sampling scheme for training the network. The blue and the red are the target vehicle and surrounding vehicles respectively. Multiple copies of the same vehicle id indicates different time instances.

C. Sampling the Sequence

A sliding window is used to generate sequence samples which can be used for the SeqToOne architecture. Consider generating samples for one specific target vehicle (the blue vehicle in Fig. 6) with surrounding vehicles 1,2 and 3 (the red cars in Fig. 6). To generate the first sample ($Sample_1$) of the observation sequence associated with the target vehicle, it uses its historical position information from t_0 to t_{29} (for $K = 30$ frames) with the corresponding surrounding cars in the same co-ordinate frame considering the target vehicle's position at t_{29} as origin. The target car's position at t_{30} in the same co-ordinate frame will be the ground truth output. For the next sample ($Sample_2$), for the same target vehicle we move the time window one unit forward, which means the observation sequence will be generated based on the target vehicle's position from t_1 to t_{30} considering the position at t_{30} as the origin and the position at t_{31} as the output. Using the same time window shift approach, the rest of the samples can be generated.

D. Training

The Conv-LSTM models are trained by minimizing the Euclidean distance between the predicted and the ground truth position over all the samples:

$$\mathcal{L} = \sqrt{(x_{true} - x_{pred})^2 + (y_{true} - y_{pred})^2} \qquad (6)$$

where $\mathcal{L}$ is the computed loss, x_{true} and y_{true} are the ground truth, and x_{pred} and y_{pred} are the predicted lateral and longitudinal positions, respectively.

We trained the model using the RMSprop optimizer [23] as this works better for recurrent models. The step decay learning rate starts from 0.01 and reduces by 30 percent after each 5 epochs. The model is implemented in Keras [24] with the TensorFlow backend.

V. Results and Performance Evaluation

A. Compared Models

We compare several existing models (defined below) with our proposed model, either recoding (CV,V-LSTM), downloading public code (CS-LSTM), or using published results (D-LSTM).

- **Constant Velocity (CV):** In the constant velocity (CV) model, we use the instantaneous velocity of the target

TABLE I: RMSE comparison on NGSIM dataset [20] between CV, V-LSTM, Dual-LSTM, CS-LSTM and the proposed technique at different time horizons. The Dual-LSTM results are from the original paper [25] as the code or model is not publically available.

	Prediction Horizon (meter)									
Model	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**	**6 s**	**7 s**	**8 s**	**9 s**	**10 s**
CV	0.74	2.05	3.72	5.65	7.62	9.64	11.75	13.98	16.28	18.47
V-LSTM	0.81	1.78	2.85	4.06	5.39	6.84	8.38	10.05	11.85	13.77
Dual-LSTM [25]	0.47	1.39	2.57	4.04	5.77	–	–	–	–	–
CS-LSTM [13]	0.58	1.26	2.11	3.18	4.53	6.21	8.42	11.12	14.20	17.90
Proposed	0.80	1.67	2.39	3.23	4.50	5.66	6.92	8.43	9.73	11.40

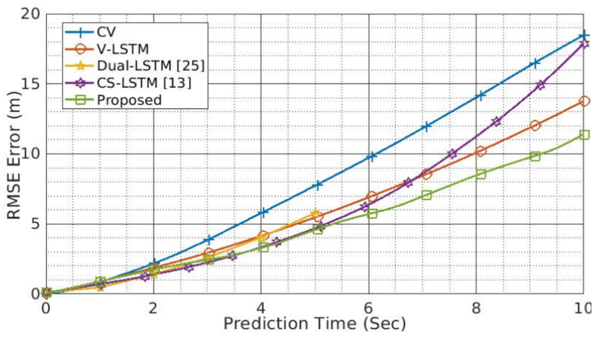
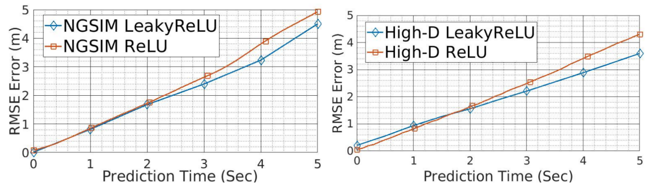


Fig. 7: The RMSE comparison of CV, V-LSTM, Dual-LSTM [25], CS-LSTM [13] and the proposed method on 4000 sequences selected from NGSIM US-101 [20] dataset. This figure shows their average mean squared errors for the prediction time horizon from 1s to 10s.



(a) NGSIM [20]

(b) High-D [22]

Fig. 8: The RMSE on NGSIM [20] and HighD [22] datasets with two different activation functions (ReLU and LeakyReLU) for Conv-LSTM layers. For each evaluation metric, we plot its average for the prediction time horizon from 1s to 5s.

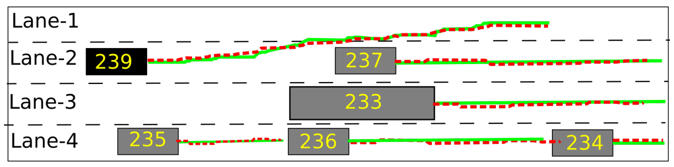
car to calculate the succeeding longitudinal and lateral positions.

- **Vanilla-LSTM (V-LSTM):** In the V-LSTM model we use only the target car's past trajectory to predict the future trajectory without using any information about the surrounding cars. To ensure fair comparison, we train this model as SeqToOne and use the proposed feedback scheme to predict the whole future sequence. The reason we compare our model with V-LSTM is to understand whether our proposed technique can capture and benefit from other surrounding cars' information to make the long term prediction more accurate.
- **Dual-LSTM (D-LSTM):** A two-stage LSTM based network is proposed in [25] by Long *et al*. The proposed method feeds the input trajectory sequence to the first LSTM block to recognize the driver intention as an intermediate step. The second LSTM block receives the recognized driver intention to estimate the future trajectory. Since the code is not available and the model is tested on the same dataset we considered the results provided in the paper.
- **Convolutional Social Pooling-LSTM (CS-LSTM):** This incorporates a maneuver based (Left Lane Change, Right Lane Change or Follow Road) motion model to generate a multi-modal predictive distribution [13].

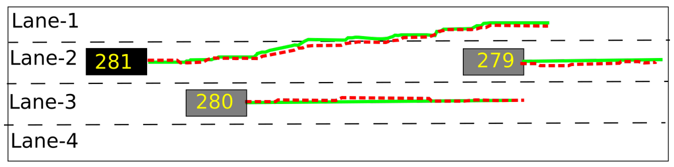
B. Performance Comparison with State-of-the-Art Methods

A comparison of performance of all algorithms over 4000 test sequences randomly selected from the NGSIM dataset [20] is shown in Fig. 7. As a performance metric, we computed the Root Mean Squared Error (RMSE) between the predicted and ground-truth positions for a future horizon of 1 to 10 seconds for all the traffic participants (Cars and HGV). It can be seen that the naive CV model produces the highest prediction error due to the absence of any temporal or interaction information. Using V-LSTM, we also employ the past trajectory, which makes future motion estimation (velocity and acceleration) more accurate and so outperforms the basic CV model. Our proposed model that includes the information on surrounding cars for the whole future prediction horizon outperforms both CV and V-LSTM; this suggests that the surrounding cars' relative motion plays a crucial role in future trajectory prediction.

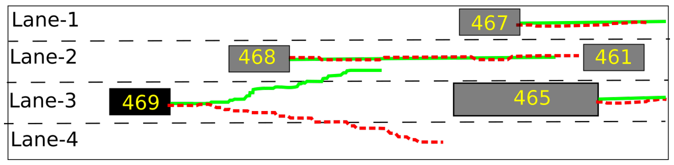
The performance of Dual-LSTM is better than both the CV and V-LSTM models during the short term horizon due to the presence of the intermediate intention recognition step, but it is still not as good as CS-LSTM or our proposed method due to the missing surrounding cars' information. CS-LSTM (with maneuver information) and our proposed technique (without maneuver information) perform almost similarly during the short term future horizon. Achieving similar performance without maneuver class information (Lane Change, Follow Road) does save the effort and necessity of labeling each trajectory sequence during the training process. In addition, since our proposed method also considers the predicted positions of all other surrounding vehicles in the scene to update the OGM frames, this leads to smaller errors during the long term prediction horizon, compared to the state-of-the-art CS-LSTM method. The RMSE of each method at different time horizons is shown in Table I.



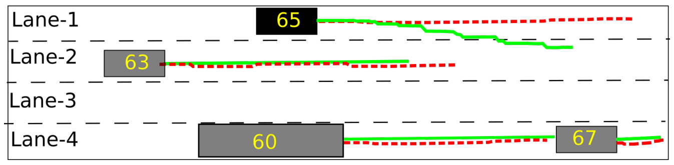
(a) Successful left lane change prediction on NGSIM dataset



(b) Successful left lane change prediction on High-D dataset



(c) Wrongly anticipated right lane change on High-D dataset



(d) Missed pulling back to right lane after overtake on High-D dataset

Fig. 9: Case studies of the prediction results by the proposed method. The predicted and ground truth trajectories are drawn in dashed Red and solid Green lines respectively. Black dashed lines are the lane markings where Lane 1 is the left most lane. For each vehicle, we plot its future 5s trajectory.

TABLE II: RMSE comparison on two different datasets (NGSIM and HighD) with two different activation functions (ReLU and Leaky-ReLU) for the Conv-LSTM layers at 5 different time horizons

		Prediction Horizon				
Dataset	**Activation**	**1 s**	**2 s**	**3 s**	**4 s**	**5 s**
NGSIM	ReLU	0.76	1.66	2.59	3.77	4.92
NGSIM	Leaky-ReLU	0.87	1.76	2.58	3.34	4.14
High-D	ReLU	0.74	1.57	2.44	3.39	4.30
High-D	Leaky ReLU	0.87	1.55	2.20	2.88	3.59

We use two different activation functions, ReLU and Leaky-ReLU, for the first three Conv-LSTM layers to identify the contribution of activation functions to model performance. The comparisons of these two settings on two different datasets, NGSIM and High-D, are shown in Fig. 8(a) and Fig. 8(b), respectively. The performance with the Leaky-ReLU activation function is better than normal ReLU on both datasets. Moreover, due to the better variety and more naturalistic trajectories, the overall performance on the High-D is better than the NGSIM dataset. The RMSEs of each setting on each dataset, at different time horizons, are shown in Table II.

C. Case Studies

The distribution of different maneuvers in most recent datasets is heavily biased towards follow road sequences or in other words an absence of frequent lane changes. To address this problem we further illustrate our results by showing the predicted trajectories (see Fig. 9) in four highly interactive scenes with a potential lane change. Fig. 9(a) shows that target vehicle 239 is preceded by a relatively slow-moving vehicle 237 within the same lane and the right adjacent lane is occupied with HGV 233. In this scenario, ideal driving behaviour would be to perform a left lane change to overtake the slow preceding vehicle provided a sufficient gap is available in the left lane which is indeed the case. Another similar scenario in the High-D dataset with a high chance of lane change is shown in Fig. 9(b) where vehicle 281 is preceded by a slow-moving vehicle 279, the right lane is occupied by another vehicle 280, and the left lane is available. In both these cases the model successfully predicted a left lane change trajectory.

Within the US driving code, the overtake maneuver should only be performed using the left vacant lane; in the case when the left lane is occupied the driver should wait until it is clear. In addition, once the overtake maneuver is completed the vehicle should cut back in to the rightmost vacant lane. Contradictory trajectories, overtaking using the right lane and not pulling back into the right lane after successful overtaking, are shown in Fig. 9(c) and Fig. 9(d) respectively. Fig. 9(c) shows the violation of the first code, where vehicle 469's ego lane and both the left lanes are congested. Due to the availability of a significant gap in the rightmost lane, our model predicted a dangerous accelerated right lane trajectory as opposed to the decelerated left lane change ground truth trajectory. Fig. 9(d) shows the violation of the second code, where vehicle 65's ego lane is empty and there is no need for an immediate lane change in terms of lane congestion. Our model predicted a follow lane trajectory as opposed to the right lane change (ground truth) trajectory, pulling back into the rightmost vacant lane. We think the reason behind these violations is missing traffic code information in the model. In future, we will consider adding rule-based layers in our model to handle these types of case.

VI. Conclusion and Future Work

In this paper, we propose a novel Conv-LSTM based architecture and an interactive feedback based prediction scheme to forecast the future trajectory of traffic participants from the current scene. Several experiments show that our proposed method can achieve comparable prediction accuracy with the state-of-the-art models during the long term prediction horizon, even without using maneuver information. Further work will be carried out to remove that underlying accumulated error to make the prediction more accurate and investigate how to benefit from driving code rules and road topology.

VII. Acknowledgement

This work was supported by Heriot-Watt University in conjunction with Jaguar Land Rover and UK-EPSRC grant EP/N012402/1 as part of the jointly funded Towards Autonomy: Smart and Connected Control (TASCC) program.

References

[1] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[2] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behaviour analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.

[3] T. Luettel, M. Himmelsbach, and H. J. Wuensche, "Autonomous ground vehicles - concepts and a path to the future," *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1831–1839, 2012.

[4] S. Chen, "Kalman filter for robot vision: a survey," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4409–4420, 2011.

[5] 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, 2013, pp. 797–802.

[6] P. Ratsamee, Y. Mae, K. Ohara, T. Takubo, and T. Arai, "Human–robot collision avoidance using a modified social force model with body pose and face orientation," *International Journal of Humanoid Robotics*, vol. 10, no. 01, p. 1350008, 2013.

[7] J. V. D. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[8] H. M. Mandalia and M. D. D. Salvucci, "Using support vector machines for lane-change detection," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 49, no. 22. SAGE Publications Sage CA: Los Angeles, CA, 2005, pp. 1965–1969.

[9] D. Vasquez, T. Fraichard, and C. Laugier, "Growing hidden Markov models: an incremental tool for learning and predicting human and vehicle motion," *The International Journal of Robotics Research*, vol. 28, no. 11-12, pp. 1486–1506, 2009.

[10] Y. Xing, C. Lv, H. Wang, H. Wang, Y. Ai, D. Cao, E. Velenis, and F. Y. Wang, "Driver lane change intention inference for intelligent vehicles: framework, survey, and challenges," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 4377–4390, 2019.

[11] A. Zyner, S. Worrall, and E. Nebot, "Naturalistic driver intention and path prediction using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[12] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1672–1678.

[13] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1468–1476.

[14] N. Mohajerin and M. Rohani, "Multi-step prediction of occupancy grid maps with recurrent neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 600–10 608.

[15] H. S. Jeon, D. S. Kum, and W. Y. Jeong, "Traffic scene prediction via deep learning: introduction of multi-channel occupancy grid map as a scene representation," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1496–1501.

[16] S. Xingjian, Z. Chen, H. Wang, D. Y. Yeungn, W. K. Wong, and W. C. Woo, "Convolutional lstm network: a machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[17] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[18] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[19] S. Dai, L. Li, and Z. Li, "Modeling vehicle interactions via modified lstm models for trajectory prediction," *IEEE Access*, vol. 7, pp. 38 287–38 296, 2019.

[20] J. Colyar and J. Halkias, "US highway 101 dataset," in *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[21] ——, "US highway I-80 dataset," in *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.

[22] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The High-D dataset: a drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2118–2125.

[23] T. Tieleman and G. Hinton, "Rmsprop: divide the gradient by a running average of its recent magnitude. coursera: neural networks for machine learning," *Tech. Rep., Technical report*, p. 31, 2012.

[24] N. Ketkar, "Introduction to Keras," in *Deep Learning with Python*. Springer, 2017, pp. 97–111.

[25] L. Xin, P. Wang, C. Y. Chan, J. Chen, S. E. Li, and B. Cheng, "Intention-aware long horizon trajectory prediction of surrounding vehicles using dual lstm networks," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 1441–1446.