Spatio-Temporal Look-Ahead Trajectory Prediction using Memory Neural Network

Nishanth Rao
Dept. of Aerospace Engineering
Indian Institute of Science
Bangalore, India
nishanthrao@iisc.ac.in

Suresh Sundaram
Dept. of Aerospace Engineering
Indian Institute of Science
Bangalore, India
vssuresh@iisc.ac.in

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9534209

***Abstract*—Prognostication of vehicle trajectories in unknown environments is intrinsically a challenging and difficult problem to solve. The behavior of such vehicles is highly influenced by surrounding traffic, road conditions, and rogue participants present in the environment. Moreover, the presence of pedestrians, traffic lights, stop signs, etc., makes it much harder to infer the behavior of various traffic agents. This paper attempts to solve the problem of spatio-temporal look-ahead trajectory prediction using a novel recurrent neural network called the Memory Neuron Network. The Memory Neuron Network (MNN) attempts to capture the input-output relationship between the past positions and the future positions of the traffic agents. The proposed prediction model is computationally less intensive and has a simple architecture as compared to other deep learning models that utilize LSTMs and GRUs. It is then evaluated on the publicly available NGSIM dataset and its performance is compared with several state-of-art algorithms. Additionally, the performance is also evaluated on a custom synthetic dataset generated from the CARLA simulator. It is seen that the proposed model outperforms the existing state-of-art algorithms. Finally, the model is integrated with the CARLA simulator to test its robustness in real-time traffic scenarios.**

I. Introduction

Research in autonomous vehicles has attracted a lot of interest from researchers around the world. With the rise of electric vehicles over the past few years, autonomous navigation and path planning have become an inherent feature of these vehicles. In presence of traffic, these vehicles should reach their destination and also follow traffic rules, prevent accidents, detect various traffic signs, handle reckless drivers and rogue vehicles. To be able to perform the aforementioned tasks, the autonomous vehicle must have the ability to predict the motion of it's surrounding vehicles. This will enable the vehicle to make necessary decisions at the right time. Anticipating traffic scenarios is thus a major functionality of autonomous vehicles in order to navigate safely amidst their human counterparts.

This is a very challenging problem due to the unpredictable nature of traffic agents. Their behaviour is often determined by multiple latent variables that cannot be estimated beforehand in new and unknown environments, such as the mental state and driving experiences of human drivers, road and weather conditions, destination of each vehicle in the traffic, reckless behaviour of traffic agents that involve overtaking, abrupt lane changing without indication, etc.

Many recent state-of-art deep learning models have utilized *Long-Short Term Memory* (LSTM) networks [1] and *Gated Recurrent Units* (GRUs) [2] for the trajectory prediction problem. One technique that is utilized by many approaches is that of an encoder-decoder architecture. In these approaches, the spatio-temporal context from the vehicle trajectories is extracted and then a recurrent neural network (RNN) based decoder is used to predict the future trajectories. While they have been successful in regressing the future trajectories of traffic agents over a certain time horizon, they are heavily dependent on computational resources due to their complex architecture and require a lot of training time.

This paper will attempt to address all the aforementioned problems by adapting a unique recurrent neural network called the Memory Neuron Network [3]. The Memory Neuron Network is an extension of the traditional neural network with addition of memory elements to each neuron in the network, that are capable of storing temporal information. This network has a simple architecture, and requires less computational resources as compared to the currently available state-of-art deep learning methods. The performance of the proposed model is evaluated on the publicly available NGSIM US-101 dataset. Although, NGSIM dataset provide comprehensive data of real traffic agents, it does not contain sufficient data for reckless and rogue traffic agents. To address this situation, a synthetic dataset is generated using the CARLA simulator [4] that contains the trajectories of multiple heterogeneous rogue traffic agents. As the proposed model is computationally less intensive, it allows for the deployment onto all the rogue vehicles present in the real-time traffic simulation with additional 80 normal cars. To summarize, our main contributions are as follows:

- A novel prediction model is proposed that uses a recurrent neural network - the Memory Neuron Network for the problem of spatio-temporal look-ahead trajectory prediction.
- The proposed prediction model is evaluated on publicly available US-101 dataset, and the RMSE is reported along with several state-of-art methods.
- To evaluate the performance of our model with respect to

978-0-7381-3366-9/21/$31.00 ©2021 IEEE

reckless drivers, rogue vehicles are simulated on CARLA simulator and their trajectories are recorded. The model is then implemented in real-time simulation on each rogue vehicle with a look-ahead horizon of $5s$, demonstrating the robustness and the computational efficiency of the proposed model.

II. Related Work

This section sets out to explore some of the various methods currently present in the literature to address the motion prediction problem. The existing literature can be broadly classified into three parts which are discussed below.

A. Mechanics-based methods

In these approaches, vehicles are mathematically modelled using Newtonian laws of translation and rotation. Once the model is formed, an Unscented Kalman Filter (UKF) is used to estimate the states of the vehicles. [5] propose an Interactive Multiple Model Trajectory Prediction (IMMTP) which combines physics-based and manoeuvre-based predictive models. [6] use a deterministic sampling approach in the UKF process for a robust estimate of target trajectories. These models work really well in certain scenarios and short time prediction horizon. However, these approaches tend to linearize the obtained models and hence, are unable to capture the inherent non-linear characteristics in a generic traffic scenario. Another issue with these approaches is that the parameters of the mathematical model such as the dimensions of the vehicle, its braking coefficients, steering torque etc., must be set and tuned in real-time, as soon as a vehicle is detected in the vicinity. This may not be feasible when the other agent's model is unknown. A detailed study on these methods can be found in [7].

B. Human behavior-based models

These techniques attempt to build a mathematical formulation of the human behavior and utilize these as a model for the driving process. [8] apply the *theory of planned behavior* to model the driver behavior, and develop a driver model that accounts for various human aspects such as driving experiences, emotions, age, gender etc. [9] and [10] apply control theory and Markov Decision Process (MDP) to model human behaviors specifically for the navigation process in a single lane. To extend the analysis to multi-lane junctions, Hidden Markov Models are proposed to model human behaviors in [11]. Statistical models have been proposed in [12], [13] and [14] to predict driving manoeuvres and behaviors. These methods work best when the knowledge of the human behaviors and their analysis are known beforehand. However, in the case of new and unknown environments these models fail to provide reliable predictions.

C. Deep learning methods

These methods use a spatial encoder to process the raw trajectory data, and then use recurrent neural networks to estimate the future trajectories. To extract the spatial context from the trajectories, [15], [16] and [17] use a sequential point-based representation. Occupancy grid-base is another popular representation for the spatial context. These approaches model trajectories as a $2D$ sequence, which can be unstructured at times due to the missing temporal information. Extraction of the temporal context is normally done by using RNNs. [18] propose a Bayesian fuzzy model to accurately estimate the temporal dependencies. [19] and [20] also use Convolutional Neural Networks to encode the temporal context. To unify the spatial and temporal contexts, [21] follows a simple and effective approach, where both the contexts are encoded together, using a Multi-Layer Perceptron, which drastically improves the prediction performance. [22] use a RNN based encoder-decoder along with [23] to model the spatio-temporal context. For the process of predicting future trajectories different variants of RNNs have been used. [24] use a standard LSTM network for trajectory prediction on highways. [25] and [26] use Imitation Learning along with Generative Adversarial Networks to predict future trajectories. [27] use LSTMs along with Convolutional Neural Networks with social pooling layers and generate a multi-modal Gaussian model for trajectory prediction. While these approaches have helped in improving the performance, they require heavy computational resources. This can make them quite hard to be implemented in real-time scenarios.

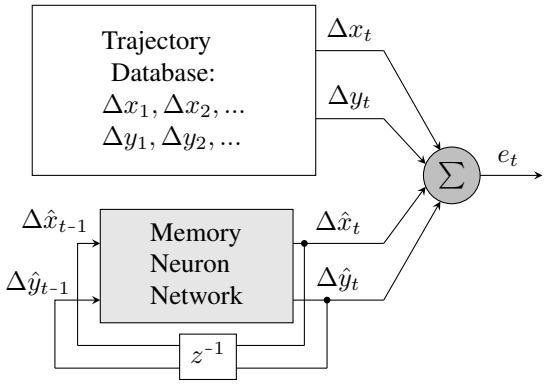

Fig. 1: Spatio-temporal lookahead model

III. Trajectory Prediction Framework

Fig. 1 shows the proposed model for trajectory prediction. The figure consists of a *trajectory database*, that consists of all the change in trajectory samples for multiple vehicles, present in the dataset, and the Memory Neuron Network which is shown as a black box. At every time instant t, the trajectory database provides the change in the (x, y) coordinates for a particular vehicle, and the network estimates the next change in position of the vehicle. The initial values provided by the trajectory database is fed to the network multiple times sequentially, so that the predicted values reach a steady-state. Once the steady-state is achieved, the network then receives consecutive input values from the trajectory database.

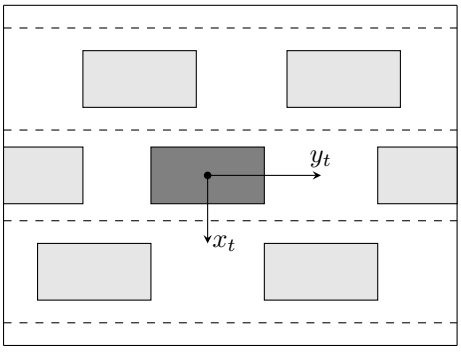

Fig. 2: The coordinate system is shown for a particular ego vehicle in a multi-lane traffic environment. The y-axis is along the longitudinal direction and the x-axis is perpendicular to it.

A. Problem Formulation

The coordinate system used for formulating the trajectory prediction problem is shown in Fig. 2. It shows the ego vehicle (filled rectangle) and the non-ego vehicles surrounding it (hollow rectangles). The location of the vehicle is measured at its centre of mass in the local coordinate frame instead of the global coordinate frame (GPS data). The ego vehicle is assumed to be equipped with sensors that can measure the position and velocity of the surrounding non-ego vehicles in the local coordinate frame . In this manner, it is possible to obtain the track histories of the non-ego vehicles present in the vicinity of the ego vehicle.

The inherent uncertainties of the sensors only provide an approximate estimate of the position and velocities of the surrounding vehicles. As a result, it is challenging to predict the future trajectories of these vehicles using simple kinematic equations. Thus, as followed in [28], a *data - driven* model is developed that can relate the past track histories of the vehicles to their future trajectories. As the values of the trajectory data can change drastically when driving from one point to another over long periods of time, the difference between consecutive (x, y) coordinates are taken:

$$\Delta \mathbf{x}_t = \mathbf{x}_t - \mathbf{x}_{t-1} \tag{1}$$

where $\mathbf{x}_t = (x_t, y_t)$ are the local coordinates of a vehicle at time instant t. As the datasets are generated through sampling data points uniformly, the difference in the trajectory samples will be bounded within certain limit, ensuring network stability and improved performance. The trajectory prediction problem is then, posed as a *system identification* problem, with the state of the system given by $\Delta \mathbf{x}_t$. Assuming this system is *observable*, from [29] the state of the system can be formulated as:

$$\Delta \mathbf{x}_t = F(\Delta \mathbf{x}_{t-1}, \Delta \mathbf{x}_{t-2}, ..) \tag{2}$$

where $F(.)$ is an unknown nonlinear function of the previous states. The goal of the network is to predict the next change in coordinates $(\Delta \hat{\mathbf{x}}_t)$ of the vehicle at time t such that the cost function J is minimized at every time step. Here J is given by

$$J = \left\| \Delta \mathbf{x}_t - \Delta \hat{\mathbf{x}}_t \right\|_2 \tag{3}$$

where $\|.\|_2$ represents the L^2 norm.

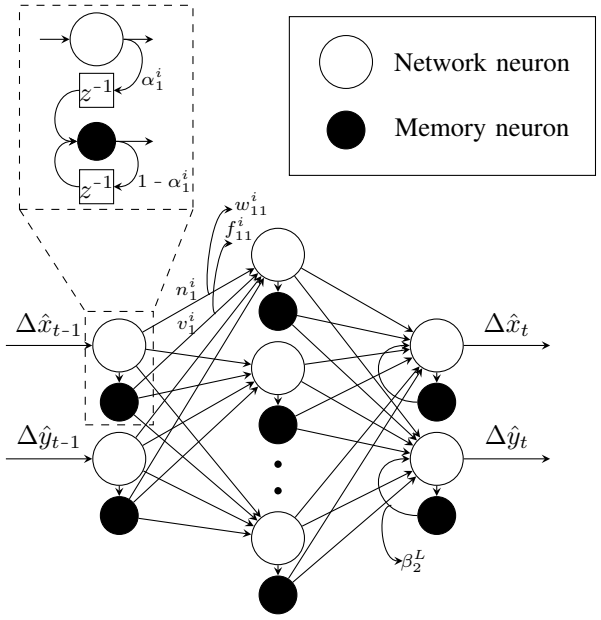

Fig. 3: The memory neuron network is fully connected with 6 hidden neurons. Every neuron has a memory neuron associated with it. Initially, the network is trained with zero inputs so that the weights stabilize to some equilibrium point, before providing the actual data.

B. Network Architecture

The network architecture is shown in Fig. 3. The figure shows some of the network parameters that provides clarity on understanding the functioning of the network. The Memory Neuron Network consists of fully connected *network neurons* (large open circles) and its associated *memory neurons* (small filled circles). There are weights associated with both the connections of network neurons and memory neurons. Both these weights are updated during backpropagation.

To describe the functioning of the network, let $\Delta \mathbf{x}_{t-1} = (\Delta \hat{x}_{t-1}, \Delta \hat{y}_{t-1})$ be the inputs to the network. The net output $n_j^h(t)$ of the j^{th} network neuron in the hidden layer h can be calculated as:

$$m_j^h(t) = \sum_{k=1}^{2} w_{kj}^i n_k^i(t) + \sum_{k=1}^{2} f_{kj}^i v_k^i(t) \tag{4}$$

$$n_j^h(t) = g^h\left(m_j^h(t)\right), \quad 1 \le j \le 6 \tag{5}$$

where,

- w_{kj}^i is the weight of the connection from k^{th} network neuron in the input layer i to j^{th} network neuron of the hidden layer h.
- $n_k^i(t)$ is the output of the k^{th} network neuron in the input layer i. In our case, $n_1^i(t) = \Delta \hat{x}_{t-1}$ and $n_2^i(t) = \Delta \hat{y}_{t-1}$.

- f_{kj}^{i} is the weight of the connection from the memory neuron corresponding to the k^{th} network neuron in the input layer i to j^{th} network neuron of the hidden layer h.
- $v_k^i(t)$ is the output of the memory neuron of the k^{th} network neuron in the input layer i.
- $g^h(.) = tanh(.)$ is the activation function of the network neurons present in the hidden layer.

The output of the memory neuron corresponding to the j^{th} network neuron in the layer l is given by:

$$v_j^l(t) = \alpha_j^l n_j^l(t\text{ - }1) + (1\text{ - }\alpha_j^l)v_j^l(t\text{ - }1),\ l \in \{i, h, L\} \quad (6)$$

where α_j^l is the weight of the connection from j^{th} network neuron in the input layer l to its corresponding memory neuron. The net output $n_j^L(t)$ of the j^{th} network neuron in the last layer L is calculated as:

$$m_j^L(t) = \sum_{k=1}^{6} w_{kj}^h n_k^h(t) + \sum_{k=1}^{6} f_{kj}^h v_k^h(t) + \beta_j^L v_j^L(t) \quad (7)$$

$$n_j^L(t) = g^L\left(m_j^L(t)\right), \quad 1 \le j \le 2 \quad (8)$$

where,

- β_j^L is the weight of the connection from the memory neuron to its corresponding j^{th} network neuron in the last layer L.
- $v_j^L(t)$ is the output of the memory neuron corresponding to the j^{th} network neuron in the last layer L.
- $g^L(.)$ is a linear activation function with unit slope for the network neurons in the output layer L.
- $n_j^L(t)$ is the output of the j^{th} network neuron in the last layer L. In our case, $n_1^L(t) = \Delta\hat{x}_t$ and $n_2^L(t) = \Delta\hat{y}_t$.

To ensure the stability of the network dynamics, the following condition is imposed: $0 \le \alpha_j^l, \beta_j^L \le 1$.

The backpropagation algorithm is used to update all the weights of the network corresponding to both the network neurons as well as the memory neurons. The following squared error function is used for backpropagation:

$$e(t) = \sum_{j=1}^{2} (n_j^L(t) - d_j(t))^2 \quad (9)$$

where $d_j(t)$ is the desired teaching signal that is derived from the trajectory database. In our case, $d_1(t) = \Delta x_t$ and $d_2(t) = \Delta y_t$.

At the time of updation $t = \tau$, the weights are updated by using the following rule:

$$w_{kj}^l(\tau + 1) = w_{kj}^l(\tau) - \eta e_j^{l+1}(\tau) n_i^l(\tau),\ \ l \in \{i, h\} \quad (10)$$

$$f_{kj}^l(\tau + 1) = f_{kj}^l(\tau) - \eta e_j^{l+1}(\tau) v_i^l(\tau),\ \ l \in \{i, h\} \quad (11)$$

where η is the learning rate for the weights of the network, and

$$e_j^L(\tau) = \left(n_j^L(\tau) - d_j(\tau)\right), \quad 1 \le j \le 2 \quad (12)$$

$$e_j^h(\tau) = \left(g^h(m_j^i(\tau))\right)' \sum_{p=1}^{2} e_p^L(\tau) w_{jp}^h(\tau), \quad 1 \le j \le 6 \quad (13)$$

The various memory coefficients are updated using the following equations:

$$\alpha_j^l(\tau + 1) = \alpha_j^l(\tau) - \eta' \frac{\partial e}{\partial v_j^l}(\tau) \frac{\partial v_j^l}{\partial \alpha_j^l}(\tau) \quad (14)$$

$$\beta_j^L(\tau + 1) = \beta_j^L(\tau) - \eta' e_j^L(\tau) v_j^L(\tau) \quad (15)$$

where η' is the learning rate for updating the memory coefficients, and

$$\frac{\partial e}{\partial v_j^h}(\tau) = \sum_{s=1}^{N_{l+1}} f_{js}^h(\tau) e_s^L(\tau) \quad (16)$$

$$\frac{\partial v_j^l}{\partial \alpha_j^l}(\tau) = n_j^l(\tau\text{ - }1)\text{ - }v_j^l(\tau\text{ - }1) \quad (17)$$

where N_{l+1} is the number of network neurons in the layer next to l. The memory coefficients are hard-limited to $[0, 1]$ if they happen to fall outside the range. For a detailed discussion on the functioning of the network and additional details, please refer to [3].

A crucial requirement in system identification problems is to determine how many previous inputs and outputs are to be fed back to the model to capture the generic nonlinear input-output mapping of the model. The presence of the memory neurons ensures that this requirements is optimally learnt during the learning process. Note that the output of the network depends on the previous inputs as well as its own outputs due to the presence of memory neurons in the output layer. Thus, the estimated next state of the system $\Delta\hat{\mathbf{x}}_k$ is given by:

$$\Delta\hat{\mathbf{x}}_t = \hat{F}(\Delta\hat{\mathbf{x}}_{t\text{-}1}, \Delta\hat{\mathbf{x}}_{t\text{-}2}, ...) \quad (18)$$

where $\hat{F}(.)$ is the nonlinear transformation represented by the Memory Neuron Network. The predicted samples $\Delta\hat{\mathbf{x}}_t$ depends on the previous inputs due to the presence of memory neurons in the input and hidden layers, and it depends on its own previous outputs due to the presence of memory neurons in the output layer. Thus, the spatio-temporal look-ahead model represented by Fig. 1 is known as parallel identification model [30].

C. Training and Implementation Details

The Trajectory database consists of differences between consecutive trajectory samples, as given by equation (1). During the learning process, at every time step t the network receives the previous state information $\Delta\mathbf{x}_{t\text{-}1}$, and predicts the estimated next state $\Delta\hat{\mathbf{x}}_t$. The actual state of the system $\Delta\mathbf{x}_t$ is then used as a *teaching signal*, to backpropagate the squared error $\|\Delta\mathbf{x}_t\text{ - }\Delta\hat{\mathbf{x}}_t\|_2^2$ and update both the weights associated with the network neurons and the memory neurons. The network consists of six neurons in the hidden layer, with *tanh(.)* as its activation function, and *linear* activation function in the output layer. The number of hidden neurons were chosen based on experimentation, and no significant improvement was observed when the number of hidden neurons were increased. It was also noted that there was no significant improvement,

TABLE I: Root Mean Square Error (RMSE) values (in meters) are reported over a prediction horizon of $5s$ for the NGSIM dataset.

Time	CV	CV-GMM [31]	GAIL-GRU [32]	LSTM	MATF [33]	CS-LSTM [27]	S-LSTM [34]	UST [21]	UST-180 [21]	MNN
$1s$	0.73	0.66	0.69	0.68	0.67	0.61	0.65	0.58	0.56	**0.36**
$2s$	1.78	1.56	1.56	1.65	1.51	1.27	1.31	1.20	1.15	**0.85**
$3s$	3.13	2.75	2.75	2.91	2.51	2.09	2.16	1.96	1.82	**1.38**
$4s$	4.78	4.24	4.24	4.46	3.71	3.10	3.25	2.92	2.58	**1.92**
$5s$	6.68	5.99	5.99	6.27	5.12	4.37	4.55	4.12	3.45	**2.74**

when the number of memory neurons associated with the network neurons in the output layer were increased. The range of the activation function is adjusted according to the range of the state values of the system, to avoid clipping during the prediction phase. It's slope is also adjusted to provide a linear relationship with unit slope, about the origin.

The entire trajectory data is taken for a vehicle, and the difference between consecutive trajectory samples is calculated and stored in the trajectory database for every vehicle. They will be referred as *differential trajectory samples*. Each sample is then presented to the network sequentially and is trained using backpropagation. One epoch is said to be completed when the last sample in the set of differential trajectories samples is presented and learnt. This procedure is repeated for 100,000 epochs, for multiple vehicle trajectories. The learning rates for both type of weights is chosen to be 4×10^{-6}. Algorithm 1 summarizes the training procedure. The entire model is implemented in `Python` using `NumPy` library [35].

Algorithm 1: Training pseudocode

Input : A list $\mathcal{D} = [d_i]$, $i = 1, 2, \cdots, n$, where each element is a set of differential trajectory data $d_i = \left\{\Delta \mathbf{x}_t^{(i)}\right\} = \left\{(\Delta x_t^{(i)}, \Delta y_t^{(i)})\right\}_{t=1}^{T}$ for vehicle i, learning rates η, η', *epochs*;

Output : Trained memory neuron model for trajectory prediction;

Initialize: Initialize the weights of the network arbitrarily, except the memory coefficients which are initialized to zero.;

```
foreach d_i ∈ D do
    for e ← 0 to epochs do
        foreach Δx_t ∈ d_i do
            Compute output of the network using
              feedforward equations (4) - (8);
            Compute error for backpropagation using
              equation (9)
            Update all the weights and the memory
              coefficients using equations (10) - (17);
        end foreach
    end for
end foreach
```

IV. Performance Evaluation

In this section, the proposed model is evaluated on two datasets, and the performance is compared quantitatively with several state-of-art techniques by employing the RMSE metric.

A. Datasets

For evaluating the performance of the proposed model, the following datasets are used:

(a) *NGSIM US-101 [36]*: The Next Generation Simulation (NGSIM) US-101 dataset consists of trajectory data sampled at 10Hz, over a span of 45 minutes. The trajectory data is reported in both global as well as local coordinate frames. These trajectories are recorded from a fixed bird's eye view, and consists of varying traffic conditions. A similar experimental setup is followed as in [27], where $3s$ of trajectory history is chosen to predict the estimated trajectories over the horizon of next $5s$ during the testing phase.

(b) *Synthetic Dataset:* In order to predict trajectories of rogue vehicles, the trajectories for 20 different rogue vehicles is generated by using the `CARLA` simulator. The rogue vehicles are made to skip traffic lights randomly and move in a zig-zag fashion within the lane, while traveling at a dangerously high velocity. They can also change lanes abruptly without any indication. The trajectory data is sampled at 20Hz over a 1-minute duration. In order to capture abrupt changes in the trajectories of rogue vehicles, they are sampled at a higher rate of 20Hz. The same procedure of choosing $3s$ of trajectory history and predicting the estimated trajectories over the horizon of next $5s$ during the testing phase is followed.

B. Evaluation metric

During the prediction phase, the differential trajectory samples from the trajectory database is provided for a duration of $3s$ to the network and for the next $5s$, the input to the network is it's previous outputs. The predicted values of the network are summed up with the starting actual trajectory values of each vehicle over the duration of $5s$ to generate the predicted actual trajectory of the vehicle. In order to compare the results of the proposed model quantitatively, the root mean squared

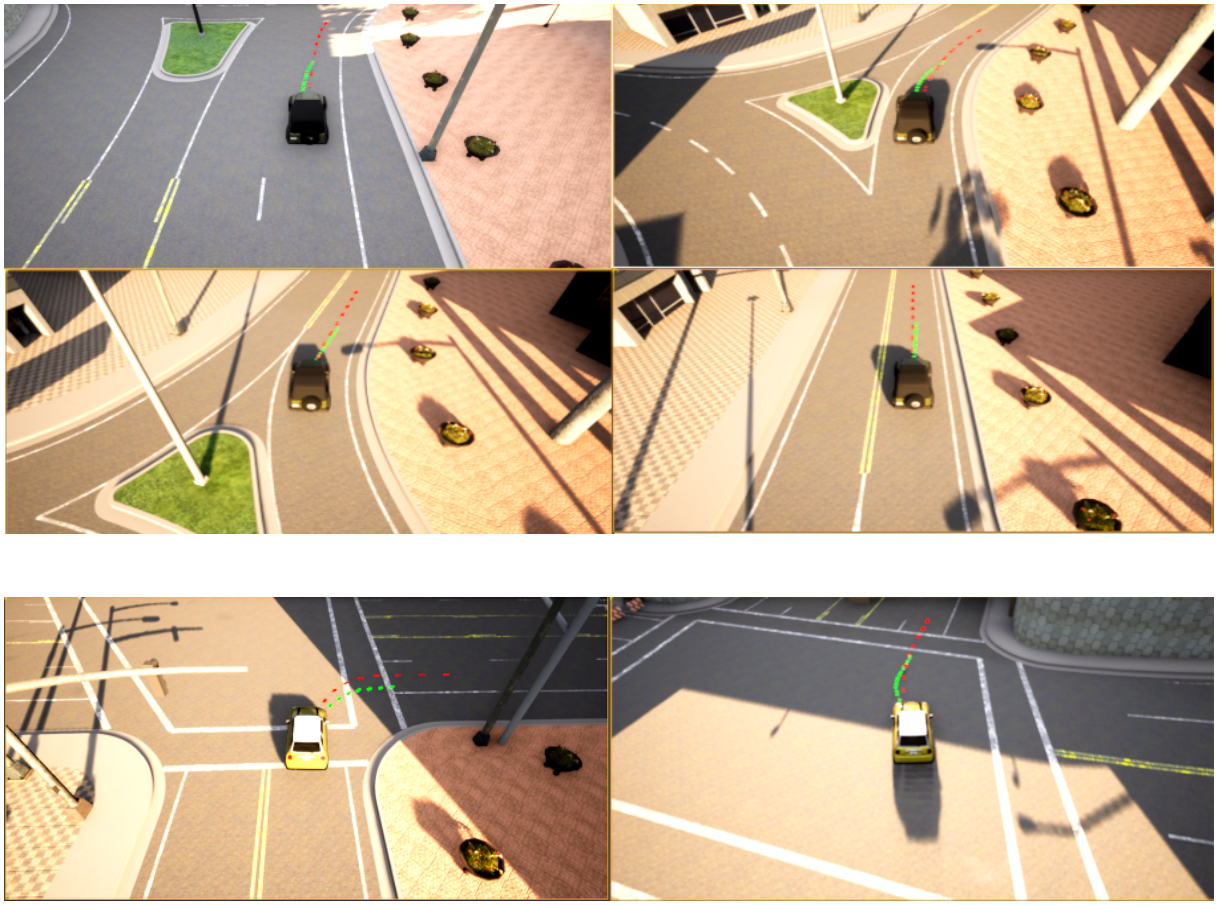

Fig. 4: Simulating trajectory prediction on CARLA for two rogue vehicles. The trained model is deployed on each of the rogue vehicle present in the simulation, so that the other vehicles present in the traffic get a '$5s$' look-ahead of every rogue vehicle. This way, they can plan some protective measures to avoid any collision with them. The predicted trajectories are shown frame-by-frame in green dotted lines for future $5s$, and the actual trajectories given by the planner are shown for $10s$ in red dotted lines. Figure on top shows a car traveling at a roundabout. The bottom figure shows the trajectory prediction at a junction.

error (RMSE) metric is used over all future time steps T_H and number of vehicles N:

$$\text{RMSE} = \frac{\sum_{n=1}^{N} \sqrt{\frac{\sum_{t=1}^{T_H} \left\| \mathbf{x}_t^{(n)} - \hat{\mathbf{x}}_t^{(n)} \right\|^2}{T_H}}}{N} \tag{19}$$

C. Results

The performance of the Memory Neuron Network is reported along with several state-of-the-art algorithms tested on the NGSIM US-101 dataset in Table I. The table consists of the RMSE for a look-ahead duration of $1s$ to $5s$ for 9 algorithms, which has been reproduced from [21]. It is evident that the Memory Neuron Network outperforms all the other algorithms. Our results have improved by 35% for $1s$ prediction horizon, and about 20% for $5s$ prediction horizon when compared to [21]. Further, the rise in the RMSE values from $1s$ horizon to $5s$ horizon is far less for our proposed model, as compared to other algorithms. From this analysis, it can also be concluded that the proposed model is relatively more stable, than the current existing algorithms.

This superior performance can be attributed to the fact that the memory neurons not only remember their own past values, but the past values of all the other memory neurons in it's preceding layers as well. This makes the Memory Neuron Network *globally recurrent*, as compared to the LSTM networks which are locally recurrent.

To test it's robustness, the trained model is deployed in real-time simulation, with 100 cars.[1] The simulation is carried out using C++ APIs provided by CARLA's unreal environment. Only the feedforward part of trained network is implemented in each of the rogue vehicle's trajectory planner. About 20% of them are rogue vehicles. The simulation consists of mixed vehicles, ranging from small cars to heavy trucks. The future trajectories of all the rogue vehicles are predicted, based on their current location and their $3s$ past track histories. The prediction of the trajectories are shown for two different rogue vehicles as frame-by-frame snapshots in Fig. 4.

[1] A detailed video demonstration on the same can be found here.

It can be observed from Fig. 4 that there is minimal error between the predicted trajectories and the actual future trajectories, when the vehicle is travelling in a near-straight path. The bottom left figure shows the predicted trajectories at the beginning of a left-turn manoeuvre. It is evident that there is a relatively higher error in this scenario, as the model cannot anticipate the radius of curvature of the turning due to the fact that it has no prior knowledge about the map and the dimensions of the roads and junctions present in the map. This shouldn't be concerning, as the predicted trajectory has the same structure of the actual future trajectory, and thus it can be inferred that the vehicle is still going to take a left-turn. This also holds true with erratic movements like the zigzagging of the rogue vehicles. While the predicted trajectory may not be *exact* as the trajectory that the rogue vehicle takes in the course of next $5s$, the overall behavior of the rogue vehicle can still be inferred correctly.

V. Conclusions and future works

This paper presents a trajectory prediction model, which uses a novel recurrent neural network as its base model. The trajectory prediction problem is posed as a system identification problem, where the Memory Neuron Network learns the input-output relationship between the past trajectory samples and the future predicted trajectory samples. It is clear that the proposed model outperformed all the state-of-art algorithms currently available, and is also very efficient in the sense that it requires less resources when training, computationally faster due to it's less complicated architecture. The proposed model has a RMSE that is about 20% lesser than the RMSE reported by the current state-of-art algorithms, for a $5s$ look-ahead prediction . The robustness of the proposed model is also verified by deploying it in the `CARLA` simulator, for each rogue vehicle. While the model performs very well in relatively straighter paths, it fails to predict the trajectories accurately at a junction as it is not aware of the structure of the map. The proposed model will be improved in this regards by adding some features related to the roads and junctions present in the map during the training process, in one of our future works.

VI. Acknowledgments

The authors would like to thank Dr. Shirin Dora and Dr. Chandan Gautam for their valuable suggestions and comments, and would also like to acknowledge the Wipro-IISc Research Innovation Network (WIRIN) fund for supporting this research.

References

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

[2] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[3] P. Sastry, G. Santharam, and K. Unnikrishnan, "Memory neuron networks for identification and control of dynamical systems," *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 306–319, 1994.

[4] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[5] G. Xie, H. Gao, L. Qian, B. Huang, K. Li, and J. Wang, "Vehicle trajectory prediction by integrating physics-and maneuver-based approaches using interactive multiple models," *IEEE Transactions on Industrial Electronics*, vol. 65, no. 7, pp. 5999–6008, 2017.

[6] H. Veeraraghavan, N. Papanikolopoulos, and P. Schrater, "Deterministic sampling-based switching kalman filtering for vehicle tracking," in *2006 IEEE Intelligent Transportation Systems Conference*. IEEE, 2006, pp. 1340–1345.

[7] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *2008 11th international conference on information fusion*. IEEE, 2008, pp. 1–6.

[8] L. Li, Y. Liu, J. Wang, W. Deng, and H. Oh, "Human dynamics based driver model for autonomous car," *IET Intelligent Transport Systems*, vol. 10, no. 8, pp. 545–554, 2016.

[9] A. L. Ferreira, G. F. B. Piccinini, S. Rôla, and A. Simões, "Gender and age-related differences in the perception of in-vehicle mobile phone usage among portuguese drivers," *IET Intelligent Transport Systems*, vol. 7, no. 2, pp. 223–229, 2013.

[10] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[11] X. Zou and D. M. Levinson, "Modeling pipeline driving behaviors: Hidden markov model approach," *Transportation research record*, vol. 1980, no. 1, pp. 16–23, 2006.

[12] P. Boyraz, A. Sathyanarayana, J. H. Hansen, and E. Jonsson, "Driver behavior modeling using hybrid dynamic systems for 'driver-aware'active vehicle safety," *Proceedings of the Enhanced Safety of Vehicles*, pp. 1–8, 2009.

[13] N. Dapzol, "Driver's behaviour modelling using the hidden markov model formalism," in *ECTRI Young researchers seminar, The Hague, the Netherlands*, vol. 2, no. 2.2, 2005, pp. 2–1.

[14] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, "Human behavior modeling with maximum entropy inverse optimal control." in *AAAI Spring Symposium: Human Behavior Modeling*, vol. 92, 2009.

[15] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.

[16] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[18] S. Subhrajit, P. Mahardhika, and S. Sundaram, "Bayesian neuro-fuzzy inference system (banfis) for temporal dependency estimation," *IEEE Transactions on Fuzzy Systems*, 2020.

[19] X. Li, X. Ying, and M. C. Chuah, "Grip: Graph-based interaction-aware trajectory prediction," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3960–3966.

[20] N. Nikhil and B. Tran Morris, "Convolutional neural network for trajectory prediction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 0–0.

[21] H. He, H. Dai, and N. Wang, "Ust: Unifying spatio-temporal context for trajectory prediction in autonomous driving," *arXiv preprint arXiv:2005.02790*, 2020.

[22] K. Messaoud, I. Yahiaoui, A. Verroust, and F. Nashashibi, "Attention based vehicle trajectory prediction," *IEEE Transactions on Intelligent Vehicles*, 2020.

[23] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[24] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 353–359.

[25] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1534–1539.

[26] W. Si, T. Wei, and C. Liu, "Agen: Adaptable generative prediction networks for autonomous driving," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 281–286.
[27] 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.
[28] M. V. Kumar, S. Omkar, R. Ganguli, P. Sampath, and S. Suresh, "Identification of helicopter dynamics using recurrent neural networks and flight data," *Journal of the American Helicopter Society*, vol. 51, no. 2, pp. 164–174, 2006.
[29] I. Leontaritis and S. A. Billings, "Input-output parametric models for non-linear systems part i: deterministic non-linear systems," *International journal of control*, vol. 41, no. 2, pp. 303–328, 1985.
[30] K. Narendra and K. Parthasarathy, "Identification and control of dynamical systems containing neural networks," *IEEE Trans. on Neural networks.*, vol. 2, no. 2, pp. 252–262, 1991.
[31] N. Deo, A. Rangesh, and M. M. Trivedi, "How would surround vehicles move? a unified framework for maneuver classification and motion prediction," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 2, pp. 129–140, 2018.
[32] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 204–211.
[33] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, "Multi-agent tensor fusion for contextual trajectory prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 126–12 134.
[34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.
[35] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. [Online]. Available: https://doi.org/10.1038/s41586-020-2649-2
[36] J. Colyar and J. Halkias, "Us highway 101 dataset," *Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-07-030*, 2007.
[37] I. Ajzen *et al.*, "The theory of planned behavior," *Organizational behavior and human decision processes*, vol. 50, no. 2, pp. 179–211, 1991.
[38] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 137–146.
[39] S. Samanta, S. Ghosh, and S. Sundaram, "A meta-cognitive recurrent fuzzy inference system with memory neurons (mcrfis-mn) and its fast learning algorithm for time series forecasting," in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2018, pp. 2231–2237.
[40] S. Samanta, M. Pratama, S. Sundaram, and N. Srikanth, "Learning elastic memory online for fast time series forecasting," *Neurocomputing*, vol. 390, pp. 315–326, 2020.