Probabilistic Modeling of Vehicle Acceleration and State Propagation With Long Short-Term Memory Neural Networks

Ian Jones[1] and Kyungtae Han[2]

***Abstract*— The success of Intelligent Driver Assistance (IDA) depends on the system's ability to accurately model the state of traffic surrounding the ego vehicle and predict driving behavior of the surrounding vehicles in order to help the ego driver make the best informed decisions in real-time. The ability to predict acceleration behavior is crucial as a first step towards modeling traffic patterns. In this paper, we show that Long Short-Term Memory (LSTM) neural networks are capable of producing acceleration distributions from which accurate future acceleration values can be sampled. Furthermore, state values calculated from these acceleration predictions are used as input for future predictions, showing that these networks are capable of generating realistic simulated vehicle trajectories over short prediction horizons.**

I. INTRODUCTION

Intelligent Driver Assistance (IDA) systems must be able to accurately predict the future state of traffic in order to successfully improve the driving behavior of the ego driver in real-time. Although modeling the current state of traffic is trivial, in order to model and predict the future state of traffic, the system must be capable of learning the multitude of interactions and connections between the various elements of the current traffic state.

Despite the complexity of these calculations, humans are capable of relatively high success in this area. Human drivers create models of other drivers' behavior and base their decisions off of these models. Take, for example, a driver trying to merge onto a highway onramp. The driver must look at the lane they are trying to merge into, record the current (and constantly changing) state of traffic in that lane and quickly create models of the future behavior of each vehicle in that lane; based off these models, they must then determine the best time to execute their lane change in order to minimize the risk of collision. This involves complex calculations of the probabilities that other drivers will slow down or speed up in order to let the driver merge in front of or behind them. All of the calculations are done almost instantaneously, and allow for the efficient flow of traffic. In order for IDA to be successful and helpful, the system must be able to accurately model and predict the behavior of the surrounding vehicles and give the driver information in real time that will help them make the best driving decisions, e.g. notifying the driver of which vehicle in the neighboring lane has the highest chance of yielding to allow the driver to merge in front of them.

Many approaches have been taken to predict the behavior of the surrounding vehicles, whether it be with the goal of fully autonomous driving or driver assistance. Recently, there has been much success using recurrent neural networks (RNNs) to predict driver behavior [14]. RNNs have been successfully utilized in many different areas, from recognition tasks such as reading handwriting to generation tasks such as generating novel music and predicting the output of computer programs [5], [7], [9], [15]. In particular, long short-term memory (LSTM) neural networks, a variant of RNNs, have achieved success with a wide range of applications involving sequential data, such as music generation. LSTM neural networks consist of "memory cells" that are capable of keeping track of past inputs and learn to remember past information that is integral to making accurate predictions [6].

This paper compares the performance of LSTM neural networks that make acceleration predictions based off of varying complexities of state information, and evaluates the accuracy of future states generated from these predictions and determines how realistic the behavior generated from the networks is. The goal of this research is to determine which state information is necessary to allow for the best informed decisions by IDA systems.

II. REVIEW OF TWO VEHICLE CAR-FOLLOWING MODELS

The question of modeling vehicle behavior has been approached from many angles. The accuracy of each proposed solution depends on the quality of the traffic-flow model at its core. The two critical components of any vehicle behavior model are the car-following model (the vehicle's acceleration behavior) and the lane-changing model (the vehicle's lane changing decisions) [13]. In our work, our primary contribution is in augmenting and improving the car-following model.

In IDA systems, not only is modeling car-following important to notify the driver of potentially dangerous situations (collisions with the vehicle directly in front of or behind the driver), but also modeling the acceleration behavior of neighboring vehicles can help to notify the driver of when is the most optimal time to execute a lane switch and more importantly, when a dangerous situation is probable. There are trade-offs for each type of model. However this research proceeds with a neural driving model for reasons explained

*This work was supported by Toyota InfoTechnology Center, U.S.A., Inc.

[1]Ian Jones received a BS in Symbolic Systems from Stanford University in 2018 and an MS in Computer Science from Stanford University in 2019. ianjones@alumni.stanford.edu

[2]Kyungtae Han is a senior researcher at Toyota InfoTechnology Center U.S.A., Inc. kthan@us.toyota-itc.com

later. The following is a brief overview of other types of vehicle behavior models, their strengths, and weaknesses.

A. Traditional Fixed-Form Models

Much attention has been paid to the interaction of two vehicles in a single lane (the ego vehicle and the preceding, or leader vehicle) on the microscopic level of traffic simulation. Intuitively, the behavior of the ego vehicle is related to that of the preceding vehicle, and responds to changes in the leader's behavior in order to avoid collision and drive optimally.

Fixed-form car following models can be classified five different groups:

- The Gazis-Hermann-Rothery (GHR) model, formulated in the early sixties, is as follows:

$$a_n(t) = cv_n^m(t)\frac{\Delta v(t-T)}{\Delta x^l(t-T)} \tag{1}$$

  where a_n is the acceleration of vehicle n at time t, v is the speed of vehicle n, x and v are the relative headway distance and speed respectively between vehicle n and vehicle $n-1$ (vehicle immediately preceding vehicle n), and T is the driver reaction time. l, and m are constants. This model originated from the initial intuition that a driver's acceleration is proportional to the relative speed difference between the driver and the preceding vehicle. However, through repeated experimentation the GHR model has been found not to be robust because of its simplicity and thus has been all but abandoned [18].
- Collision Avoidance (CA) models attempt to define a "safe following distance", which is the distance within which unpredictable behavior by the preceding vehicle would result in an unavoidable collision. The original formulation of this model is as follows:

$$\Delta x(t-T) = av_{n-1}^2(t-T)+\beta_1 v_n^2(t)+\beta v_n(t)+b_0 \tag{2}$$

  however, changes that consider braking preferences of the driver have been made to the model, and it continues to be developed.
- Linear models, generally attributed to Helly [19], include terms that take into account braking of the preceding vehicle and the vehicle two in front of the ego vehicle:

$$a_n(t) = C_1\Delta v(t-T) + C_2\{\Delta x(t-T) - D_n(t)\} \tag{3}$$

$$D_n(t) = \alpha + \beta v(t-T) + \gamma a_n(t-T) \tag{4}$$

  where $D(t)$ is the desired following distance, and α, β, γ, C_1, and C_2 are calibration constants. Linear models tend to perform well at low accelerations, however produce significant errors when acceleration values begin to fluctuate greatly.
- Psychophysical models are based off the assumption that drivers will react once various thresholds are met. These are defined as a threshold based on relative speed, a threshold based on headway distance between the ego vehicle and the preceding vehicle (the "action point"), and a threshold defined after experiments in which participants were required to decide whether car-following gaps were growing or shrinking [13]. These models are difficult to calibrate because of their dependence on observations of individual drivers' perception, and are thus limited.
- Fuzzy logic-based models describe how well certain variables fit the description of selected terms. These models divide the variables into "fuzzy sets", whose membership is determined by how well the variable fits into a given set; these sets can then be combined with logical operators to create rule-based car-following behavior. Fuzzy set rules are typically defined in natural language, and they typically encounter difficulties with defining membership functions.

B. Deterministic Car-Following Models

- The Optimal Velocity Model (OVM), proposed by Newell and extended by Bando *et al.*, is collision-free because of its dependence on velocity and headway distance [21], [22]:

$$a_n(t) = \frac{1}{\tau}(V(\Delta x_n(t) - v_n(t)) \tag{5}$$

  where τ is the velocity relaxation time, Δx_n is the headway distance, and $V(\Delta x_n(t))$ is an optimal velocity function. Although the model is collision-free, because of its simplistic nature it results in unrealistically high maximum deceleration values in order to avoid collisions.
- The Intelligent Driver Model (IDM) was proposed as a way to deal with the problems encountered by the Optimal Velocity models. It relies on eight model parameters, each of which are easily interpretable and empirically measurable [20]. The IDM states that the vehicle will accelerate at time t according to [12]:

$$a_n(t) = a_{max}[1 - (\frac{v_n(t)}{v_0})^\delta - (\frac{s^*(t)}{s_n(t)})^2] \tag{6}$$

  where s_n is the actual headway distance of the nth vehicle, s^* is the minimum desired headway distance, v_0 is the vehicle desired velocity, a_{max} is the vehicle maximum acceleration, and δ is the acceleration exponent. Like the OVM, the IDM is collision-free. However unlike the OVM, it also accounts for vehicle behavior in the free-flow traffic scenario and only produces realistic acceleration values.

III. Review of Neural Driving Models

Various neural network models have been created to predict vehicle trajectories. The main two categories that have had success are convolutional neural networks that take as input video driving data [24] and LSTM neural networks [14], [23], [25]. For the sake of this article, we will briefly discuss some recent work in LSTM neural networks for predicting vehicle trajectories. In both [23] and [25], a coordinate system is used to predict future trajectory data. Although they achieve relative success (the implemented

LSTM neural networks are particularly good at predicting future positions and velocities based on these inputs), using a coordinate system has its shortcomings, particularly in translating these systems to "real world" systems in which a global coordinate system cannot be defined in real-time. We avoid these issues by using absolute velocity and acceleration of the ego vehicle, as well as relative distances and velocities between the ego and neighboring vehicles (that could conceivably be gathered in real-time), and do not explicitly define a grid system as in [25].

In [14], an LSTM is implemented that takes as input values that are similar to the parameters of the car-following models described early: headway distance, relative speed difference between the leader and ego vehicle, ego vehicle speed, and ego vehicle acceleration. The deep learning model is able to remember past states and accurately predict future accelerations based off of memory and current input. In this research, we start with a similar model to that in [14] as a baseline.

IV. Problem Statement

Our goal is to predict vehicle one-dimensional trajectories given observed vehicle traffic data. From these predicted trajectories, an IDA system can help inform drivers of other vehicles' behavior in order for drivers to make more informed decisions while driving. Networks produce distributions of future timestep acceleration values, from which sample acceleration values are drawn in order to propagate future trajectories.

V. Methodology

A. Dataset

In this work, we use the Next Generation Simulation (NGSIM) dataset [16], which contains detailed vehicle trajectory data collected by synchronized digital video cameras on the eastbound I-80 in Emeryville, California in three 15 minute periods in the afternoon. The area of the highway recorded is 500 meters in length and consists of six freeway lanes, including a high-occupancy lane and an onramp. We use the NGSIM I-80 reconstructed dataset from 4:00 p.m. to 4:15 p.m., because it corrects a multitude of errors that exist in the original dataset, particularly inconsistent distributions of acceleration, maximum speed, and minimum inter-vehicle spacing. Trajectories are smoothed to realistic curves that better mimic reality [2]-[4]. Precise location, velocity, and acceleration data for more than 2000 unique vehicles is recorded at 10 Hz.

B. Data preparation

Since the aim of this research is to predict vehicle behavior based on the surrounding vehicles, we first defined the eight neighboring vehicles, detailed in Fig. 1. Neighboring vehicles are defined as follows: *leader* vehicle is the vehicle directly ahead of the ego vehicle; *follower* vehicle is the vehicle directly behind the ego vehicle; *left* and *right* vehicles are the two vehicles closest to the ego vehicle in the lane directly left and right of the ego vehicle, respectively; *front-left* and *front-right* vehicles are the vehicles directly ahead of the *left* and *right* vehicles, respectively; and *back-left* and *back-right* vehicles are the vehicles directly behind the *left* and *right* vehicles, respectively. All neighboring vehicles are at most 200 feet away from the ego vehicle. If a given neighboring slot is not occupied, its input features are represented as 0 in the tensor input to the networks.

Inputs to the networks were discretized into 12-second segments (each of which consists of 120 timesteps), belonging to a single ego vehicle. Throughout each 12-second segment, it is possible that the IDs of the neighboring vehicles change (due to a lane change or acceleration/deceleration that causes a given vehicle to no longer be within range of the ego vehicle), and this is taken into account in the pre-processing step.

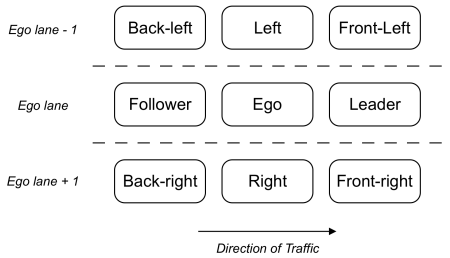

Fig. 1. Neighboring vehicles and lanes surrounding the ego vehicle. Traffic is flowing from left to right.

C. Cross-validation

Ten-fold cross-validation was performed by randomly dividing the reconstructed dataset into ten partitions, each of which was held out as a validation set once while the other nine were used as training sets. The results of each performance metric were then averaged together to report the overall performance of each of the three models. This resulted in a less biased estimation of the success of each model, and allowed us to assess how the models generalize to unseen data.

D. Experiments

All models were trained and evaluated on their ability to predict acceleration distributions and produce realistic trajectories. For each of the models, the first 2 seconds of input data of each 12-second trajectory were used to initialize the internal state of the LSTM network. Following learning, the learned models were used to produce simulated trajectories as follows: first, for each 12-second trajectory, the initial 2 seconds of true data were supplied to the network, then the network output distributions for each timestep for the remaining 10 seconds of the trajectory based off the initial 2 seconds of input. At each timestep, 50 samples were taken from the resulting acceleration distribution, and the input state for the next timestep was determined based off of the following equations:

$$\begin{aligned} v(t+\Delta t) &= v(t) + a(t+\Delta t)\Delta t \\ s(t+\Delta t) &= v(t) + s(t+\Delta t)\Delta t \end{aligned} \tag{7}$$

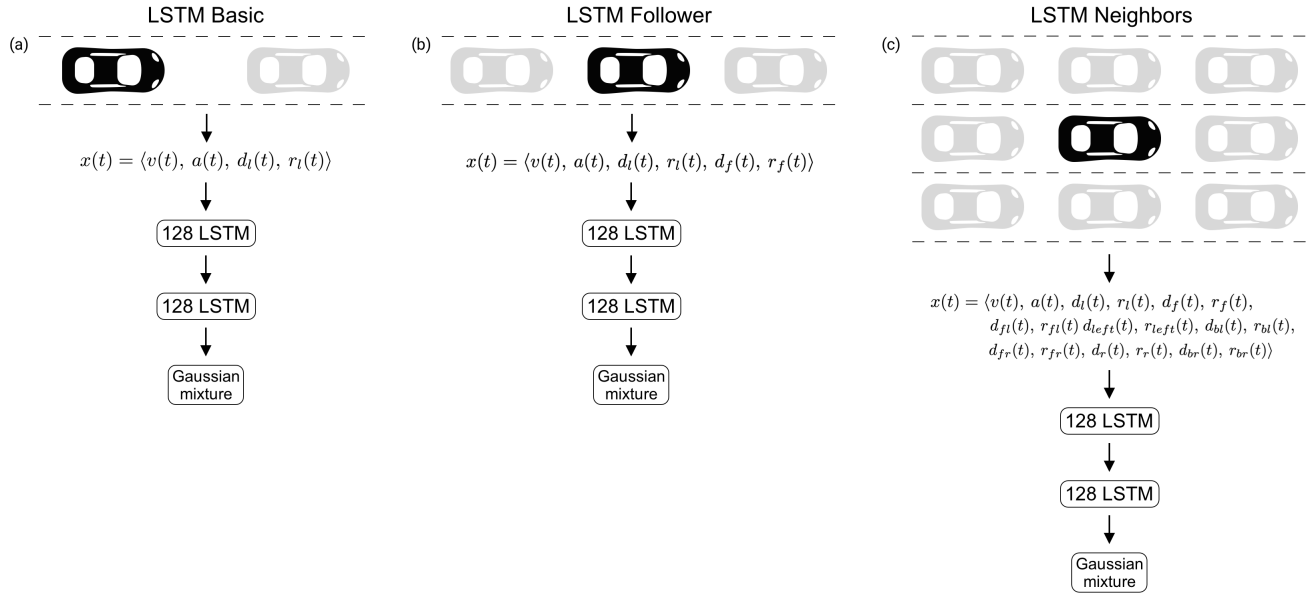

Fig. 2. Network structures from input to output. Ego vehicle appears in black, while respective relevant neighboring vehicles appear in grey.

where s, v, and a are the position, velocity, and acceleration, respectively, of a given vehicle.

This resulted in 50 simulated trajectories for each real trajectory input into the network, on which analysis was performed as described later.

VI. Models

Three LSTM neural network models were implemented. All three models take traffic state information as input and must learn a hidden state representation in order to produce distributions over vehicle accelerations over the next time-step. They each output a Gaussian mixture that models a distribution of future acceleration values, based off of a mixture density network as follows:

$$p(x) = \sum_{i=1}^{n} w_i, \mathcal{N}(x \mid \mu_i, \sigma_i^2) \tag{8}$$

where w_i, μ_i, and σ_i are the weight, mean, and standard deviation for the ith mixture component. Gaussian mixture models have been shown to have success learning multi-valued mappings from a set of example data [1], [16]. Networks were implemented in Torch7 [8] and were based on Karpathys *char-rnn* package [5] and a similar LSTM network implemented in [14].

A. LSTM Basic

The first model (LSTM Basic, Fig. 2(a)) was used as a baseline. It took as input current time-step state information about the ego vehicle and the vehicle directly in front of the ego vehicle (leader vehicle). The state information for the current timestep t is the following:

$$x(t) = \langle v(t),\ a(t),\ d_l(t),\ r_l(t)\rangle \tag{9}$$

where $v(t)$ is the ego velocity at time t, $a(t)$ is the ego acceleration at time t, $d_l(t)$ is the headway distance between the ego vehicle and the leader vehicle at time t, and $r_l(t)$ is the relative speed difference between the ego and the leader vehicle at time t.

B. LSTM Follower

The second model (LSTM Follower, Fig. 2(b)) adds information about the vehicle directly behind the ego vehicle (follower vehicle) to the state information used in LSTM Basic. The state information for the current timestep t is the following:

$$x(t) = \langle v(t),\ a(t),\ d_l(t),\ r_l(t),\ d_f(t),\ r_f(t)\rangle \tag{10}$$

where the first four elements of the current state are the same as in LSTM Basic, $d_f(t)$ is the distance headway between the ego vehicle and the follower vehicle at time t, and $r_f(t)$ is the relative speed difference between the ego and the follower vehicle at time t.

C. LSTM Neighbors

The final model (LSTM Neighbors, Fig. 2(c)) defines a group of "neighboring" vehicles that surround the ego vehicle (see diagram) and adds information about the neighboring vehicles to the state information in LSTM Follower. The state for the current timestep t includes the elements of the LSTM Follower state information, combined with distance headway and relative speed difference information for each of the 6 defined neighboring vehicles, i.e. information for the front-left, left, back-left, front-right, right, and back-right vehicles. This amounts to an input state of 18 elements:

$$\begin{aligned} x(t) = &\langle v(t),\ a(t),\ d_l(t),\ r_l(t),\ d_f(t),\ r_f(t) \\ &d_{fl}(t),\ r_{fl}(t)\ d_{left}(t),\ r_{left}(t)\ d_{bl}(t),\ r_{bl}(t) \\ &d_{fr}(t),\ r_{fr}(t)\ d_r(t),\ r_r(t)\ d_{br}(t),\ r_{br}(t)\rangle \end{aligned} \tag{11}$$

D. Hyperparameter details

The models were trained with a learning rate of 4 x 10^{-3}, which decayed at a rate of 0.97 starting after the third epoch of training. LSTM layers had 128 neurons, and models were trained for 10 epochs or until learning stabilized.

Overfitting to the training data is a common problem in deep neural networks, due to units co-adapting and developing interdependencies. Because of this, a dropout rate of 0.25 was utilized, meaning that 25% of units were turned on during training. Dropout effectively counters the problem of overfitting by randomly dropping neural units and their connections during training. This technique has been shown to improve network performance in tasks ranging from speech recognition and document classification to vision and computational biology [10].

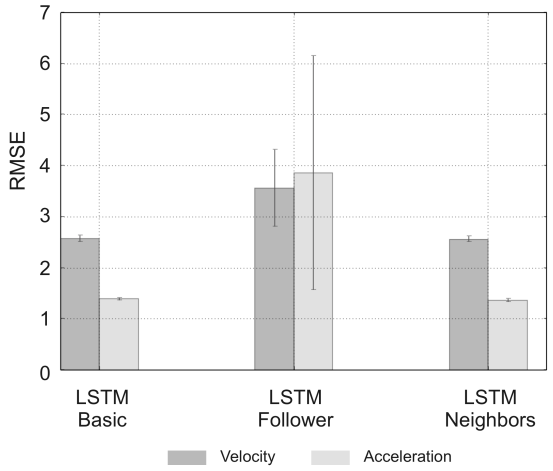

Fig. 3. Root mean squared error for predictions in the first 5 seconds of prediction horizons for each model. Error bars indicate standard error across 10 folds. LSTM Basic and LSTM Neighbors outperform LSTM Follower in acceleration and velocity predictions.

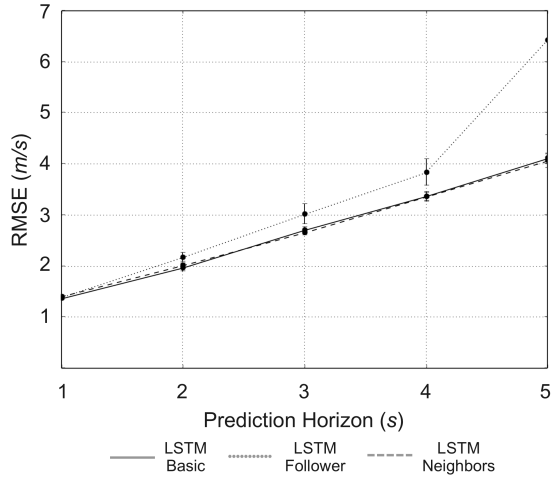

Fig. 4. Root mean squared error of velocity predictions over different prediction horizons. Error bars indicate standard error across 10 folds.

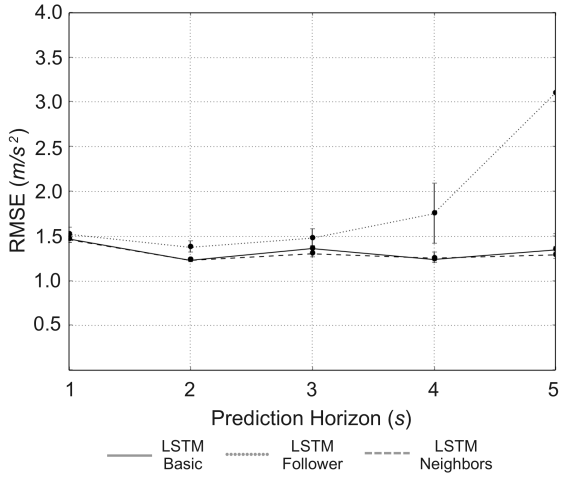

Fig. 5. Root mean squared error of acceleration predictions over different prediction horizons. Error bars indicate standard error across 10 folds.

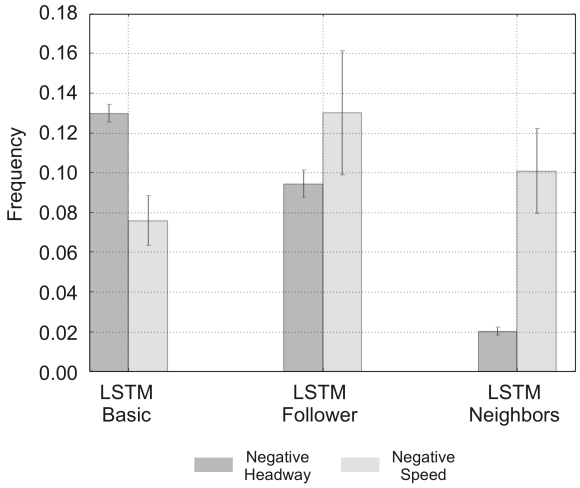

Fig. 6. Frequency of negative state value predictions for distance headway and speed, indicating unrealistic trajectories. Error bars indicate standard error across 10 folds. Negative headway distance corresponds to a collision, while negative speed corresponds to driving in reverse.

VII. Results

Several metrics were utilized to evaluate the networks' success. Root Mean Squared Error was used to evaluate prediction accuracy, while presence of negative speed and headway distance state values were used to evaluate how realistic simulated trajectories were.

A. Root Mean Squared Error

Networks were initially evaluated by computing the Root Mean Squared Error (RMSE) between each sample acceleration trajectory drawn from the network's predicted acceleration distribution and the true acceleration values for each given trajectory:

$$RMSE = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}(v_H^{(i)} - \hat{v}_H^{(i,j)})^2} \qquad (12)$$

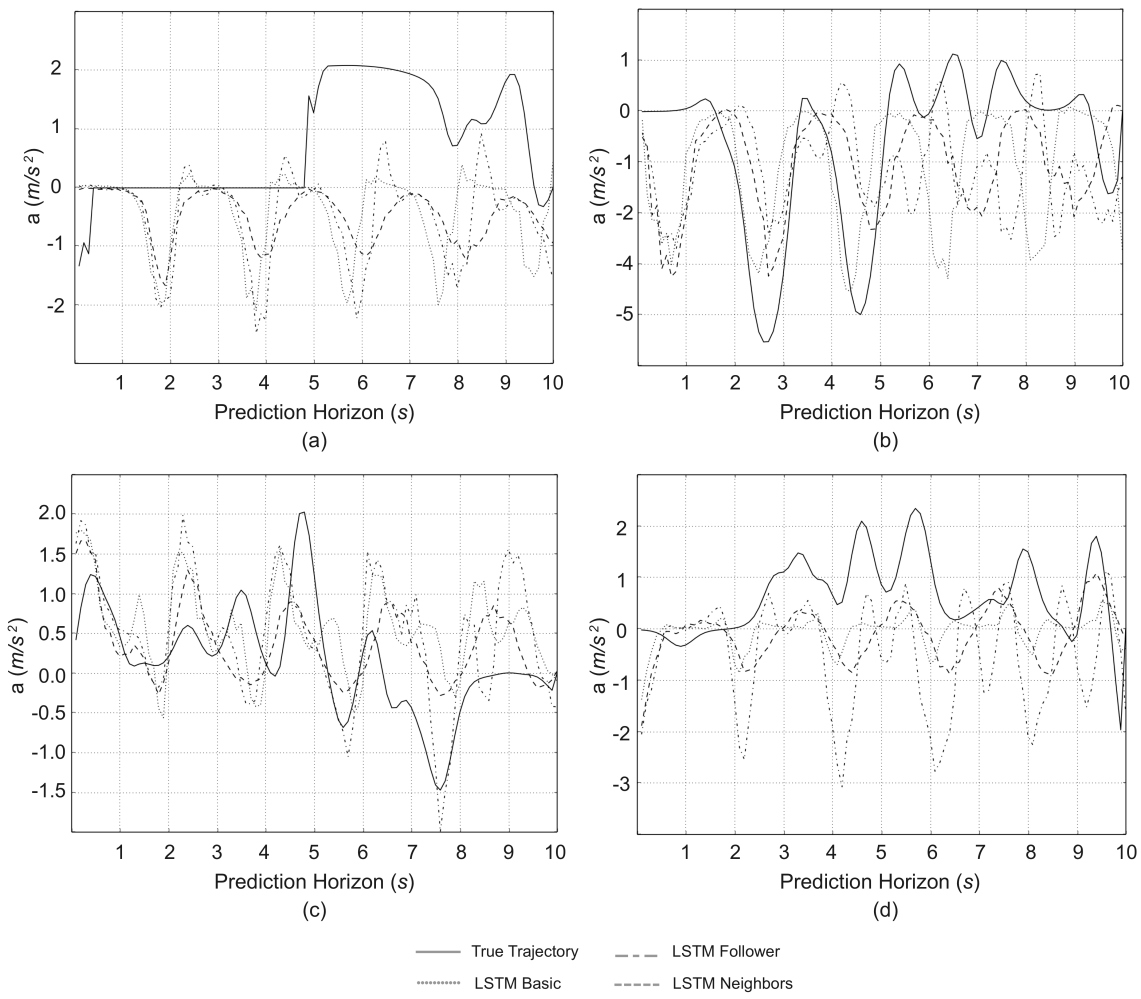

Fig. 7. Sample simulated trajectories produced by each LSTM network, compared to the true trajectories. Each graph depicts an individual 10 second trajectory for an individual ego vehicle.

where m is the number of true trajectories, $n = 50$ is the number of simulated trajectories per true trajectory, $v_H^{(i)}$ is the true value in the ith trajectory at time horizon H, and $\hat{v}_H^{(i,j)}$ is the simulated value for sample j in the ith trajectory at time horizon H. LSTM Basic and LSTM Neighbors had nearly identical RMSE evaluated across 10 second prediction horizons (see Fig. 3 for RMSE of acceleration and velocity for the first 5 seconds of predictions for all three models). As can be seen in Fig. 4 and Fig. 5, LSTM Basic and LSTM Neighbors outperform LSTM Follower when evaluated at prediction horizons 1 through 5. This is interesting, because LSTM Follower's input state contains all 4 input variables that LSTM Basic does, however perhaps the addition of follower vehicle information without neighboring vehicle context causes a decline in prediction accuracy.

B. Negative State Values

Furthermore, simulated trajectories were evaluated on how "realistic" they were. The frequencies of negative speed and headway distance values predicted were used as a metric for this realism. Since traffic data was recorded on the I80 freeway, negative speed values are impossible as they signify vehicles traveling in reverse (because simulated trajectories are propagated from acceleration distributions, it is possible that negative speeds are produced however they should be minimized). Similarly, negative headway distances signify a collision, which is also undesirable trajectory behavior. Interestingly, as can be seen in Fig. 6, both LSTM Follower and LSTM Neighbors outperform LSTM Basic in frequency of negative headway distance, however they predict higher frequencies of negative speed.

C. Analysis of Network Behavior

As can be seen in Fig. 7(a), all three networks do poorly at predicting trajectories that include long periods of acceleration values that are 0 m/s^2 or very close to 0 m/s^2. The reason for this is that these types of trajectories are extremely rare in the training data. Out of nearly 8000 real 10-second trajectories from the overall dataset, only 3 percent of them contain a stretch of 3 seconds or longer during which the ego vehicle remains at an acceleration of 0 m/s^2 (we defined any acceleration value between -0.005 m/s^2 and -0.005 m/s^2 as

an acceleration value of 0 m/s^2). Evidently, because of the scarcity of the specific types of examples to learn from in the training data, all three networks struggle to make acceleration predictions that accurately reflect true trajectories containing long periods of acceleration values of 0 m/s^2. It seems that the networks begin to predict extreme acceleration values following long periods of neutral acceleration. Perhaps this is because the networks learned patterns that after multiple timesteps of neutral acceleration values, the driver is likely to either speed up or in the case of the trajectory in Fig. 7(a), slow down rapidly in order to avoid a crash.

Fig. 7(b) and 7(c) depict relatively accurate predictions by all three networks. Interestingly, trajectories generated by LSTM Followers tend to produce extreme acceleration values, especially in the negative direction, as can be seen in Fig. 7(d). LSTM Basic seems to produce realistic trajectories, however at certain points it seems to lose context and predict randomly, as seen in the later prediction horizons in Fig. 7(b). LSTM Neighbors, on the other hand, generates trajectories that seem to follow the true trajectories more reliably and rarely produces extreme acceleration values.

VIII. CONCLUSIONS

In this article, we proposed three different LSTM networks and compared their ability to produce accurate acceleration distributions and propagate realistic simulated vehicle trajectories. We found that LSTM Neighbors performed best overall, followed by LSTM Basic and then LSTM Followers, showing that LSTM networks are capable of complex prediction and simulation in one dimension. This work can be extended to include predictions in two dimensions by predicting lane changing behavior of vehicles using the same NGSIM dataset. Our experiments showed that information required by IDA systems can be provided by reasonably accurate predictions from LSTM networks.

Looking toward real-world implementations and use cases, we see that the proposed models will be able to perform well by collecting input data in real-time: all input values necessary for these predictions would be readily available to the ego vehicle (absolute speed, acceleration) or easily gathered from Radar or Lidar readings (distance and relative speed and acceleration differences between ego and neighboring vehicles). Most importantly, these predictions will be able to be utilized in order to help ensure higher safety for the driver.

REFERENCES

[1] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 3844-3848, 2014.

[2] M. Montanino and V. Punzo, "Making NGSIM data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction," Transportation Research Part J, vol. 1, no. 2390, pp. 99-111, 2013.

[3] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," Transportation Research Part B 80, pp. 82-106, 2015.

[4] V. Punzo, M.T. Borzacchiello, and B. Ciuffo, "On the assessment of vehicle trajectory data accuracy and application to the Next Generation SIMulation (NGSIM) program data," Transportation Research Part C, vol. 19, no. 6, pp. 1243-1262, 2011.

[5] A. Karpathy, J. Johnson, and F. Li, "Visualizing and understanding recurrent networks," CoRR, vol. abs/1506.02078. [Online]. Available: http://arxiv.org/abs/1506.02078

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

[7] A. Graves, M. Liwicki, S. Fernndez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 5, pp. 855-868, May 2009.

[8] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like environment for machine learning," presented at the BigLearn, NIPS Workshop, 2011.

[9] W. Zaremba and I. Sutskever, "Learning to execute," CoRR, vol. abs/1410.4615. [Online]. Available: http://arxiv.org/abs/1410.4615

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overtting," J. Mach. Learn. Res., vol. 15, pp. 1929-1958, 2014.

[11] J. Colyar and J. Halkias, "US highway 80 dataset," Federal Highway Admin. (FHWA), Washington, DC, USA, Tech. Rep. FHWA-HRT-06137, Dec. 2006.

[12] F. E. Gunawan, "Two-vehicle dynamics of the car-following models on realistic driving condition," J. Transp. Syst. Eng. Inf. Technol., vol. 12, no. 2, pp. 67-75, 2012.

[13] S. Panwai and H. Dia, "Comparative evaluation of microscopic car-Following behavior," IEEE Trans. Intell. Transp. Syst., vol. 6, no. 3, pp. 314-325, Sep. 2005.

[14] J. Morton, T. Wheeler, and M. Kochenderfer, "Analysis of recurrent neural networks for probabilistic modeling of driver behavior," IEEE Transactions on Intelligent Transportation Systems, vol. 18, iss. 5, pp. 1289-1298, May 2017.

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

[16] C. M. Bishop, "Mixture density networks," Aston Univ., Birmingham, U.K., Tech. Rep. NCRG/4288, 1994.

[17] M. Brackstone and M. McDonald, "Car-following: A historical review," Transportation Part Research F, Traffic Psychology and Behavior, vol. 2, no. 4, pp. 181-196, 1999.

[18] T. H. Rockwell, and J. Treiterer, "Sensing and communication between vehicles," The Ohio State University, Systems Research Group, Final Report No. EFS 227-2. Columbus, Ohio, 1966.

[19] W. Helly, "Simulation of bottlenecks in single lane traffic flow," in Symp. Theory Traffic Flow, Research Laboratories, General Motors, New York, 1959, pp. 207-238.

[20] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 62, no. 2, pp. 1805-1824, 2000.

[21] G.F. Newell, "Nonlinear effects in the dynamics of car-following," Operations Research, 1961, 9(2): 209-229.

[22] M. Bando, K. Hasabe, A. Nakayama, A. Shibata, Y. Sugiyama, "Dynamical model of traffic congestion and numerical simulation," Physical Review E, 51(2):1035-1042, 1995.

[23] B. Kim, C. Kang, J. Kim, S. Lee, C. C. Chung, and J. W. Choi, "Probabilistic vehicle trajectory prediction over occupancy grid map via recurrent neural network," in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), 2017, vol. abs/1704.07049 [Online]. Available: http://http://arxiv.org/abs/1704.07049

[24] L. Chi and Y. Mu, "Deep steering: Learning end-to-end driving model from spatial and temporal visual cues," in Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities, VSCC 2017, vol. abs/1708.03798 [Online]. Available: http://arxiv.org/abs/1708.03798

[25] F. Altche, and A. De La Fortelle, "An LSTM network for highway trajectory prediction," in 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC): Workshop. IEEE, 2017.