# Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior

Jeremy Morton, Tim A. Wheeler, and Mykel J. Kochenderfer

***Abstract*—The validity of any traffic simulation model depends on its ability to generate representative driver acceleration profiles. This paper studies the effectiveness of recurrent neural networks in predicting the acceleration distributions for car following on highways. The long short-term memory recurrent networks are trained and used to propagate the simulated vehicle trajectories over 10-s horizons. On the basis of several performance metrics, the recurrent networks are shown to generally match or outperform baseline methods in replicating driver behavior, including smoothness and oscillatory characteristics present in real trajectories. This paper reveals that the strong performance is due to the ability of the recurrent network to identify recent trends in the ego-vehicle's state, and recurrent networks are shown to perform as, well as feedforward networks with longer histories as inputs.**

***Index Terms*—Recurrent neural networks, car-following models, prediction methods, autonomous vehicles, deep learning.**

## I. Introduction

COMPREHENSIVE risk assessments are required for automotive safety systems before their release. Conducting such studies often requires real-world driving tests, which are expensive, time consuming, and subject to safety constraints. Simulation allows for testing a wide range of scenarios in a fraction of the time, at marginal cost, and at no risk of injury, but must employ accurate behavior models for traffic participants in order to produce useful evaluation metrics. It is critical that the simulated behavior be as representative of actual driving as possible; otherwise, the risk associated with a safety system could be significantly over- or underestimated.

Many methods have been proposed for learning microscopic driving models from real-world data. A large body of research exists for car-following models using fixed-form distributions [1]–[4] that rely on specific response equations. In particular, Bonsall, Liu, and Young highlighted the deficiencies in these models, which they attributed to safety-related assumptions, and argued that parameters ought to be learned from real-world data [5].

Manuscript received March 3, 2016; revised June 2, 2016 and August 9, 2016; accepted August 21, 2016. This work was supported by the National Science Foundation Graduate Research Fellowship Program under Grant DGE-114747. The Associate Editor for this paper was L. M. Bergasa.

The authors are with the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA (e-mail: jmorton2@stanford.edu; wheelert@stanford.edu; mykel@stanford.edu).

Digital Object Identifier 10.1109/TITS.2016.2603007

Recent work has sought to automate the construction of general driver models from data using less restrictive probabilistic models. Agamennoni, Nieto, and Nebot developed a softmax classifier over contextual features to identify a context class with an associated Gaussian acceleration distribution [6]. Gindele, Brechtel, and Dillmann constructed a Gaussian distribution over acceleration and turn-rate using random forests over contextual features [7]. Wheeler, Robbel, and Kochenderfer used Bayesian networks to generate predictions over acceleration and turn-rate for free-flow, following, and lane-change context classes [8]. Damerow and Eggert planned using a probabilistic risk map generated using the foresighted driver model [9], [10]. Bahram, Hubmann, Lawitzky, *et al.* combined spatio-temporal cost maps to predict intentions with Bayesian networks for classifying candidate maneuvers with intention-unaware local features [11]. These methods produce distributions over future actions, which can be sampled to propagate driving scenes in simulation, but rely on hand-selected features that are limited in their ability to capture nuanced temporal and spatial characteristics.

Deep neural networks have recently gained widespread popularity as universal function approximators, capable of learning robust hierarchical features from complicated inputs [12], [13]. Deep neural networks have outperformed traditional state-of-the-art methods in fields as diverse as image classification [14] and natural language processing [15]. Their efficiency, effectiveness, and flexibility make deep neural networks highly attractive. Prior applications of neural networks to automotive behavior modeling include maximum likelihood prediction in car-following contexts [16]–[19], lateral position prediction [20], and maneuver classification to provide inputs to a hidden Markov model [21].

This paper describes neural car-following models developed based on naturalistic driving data and outlines a general methodology for constructing such models. The human driving models produce distributions over actions rather than maximum likelihood predictions, allowing for stochastic predictions and the evaluation of statistical risk. Long short-term memory (LSTM) recurrent neural networks [22] are compared to feedforward networks and traditional baselines. The LSTM architecture can automatically learn relevant spatial and temporal features, reducing the need to record and input long sequences. The resulting networks form generative distributions over driver accelerations and are used to propagate simulated trajectories. Trajectories generated from the resulting models are compared using a wide range of metrics that quantify their modeling performance and oscillatory characteristics. Finally, the spatial and temporal features learned by the neural network models are investigated, and it is found that LSTM networks rely largely on very recent information. It is shown that, if the input to a feedforward network is expanded to incorporate state information from multiple time steps, it is able to match the performance of LSTM networks.

## II. Review of Car-Following Models

Car-following models capture the longitudinal interaction between a vehicle and the vehicle(s) before it. They are critical components of traffic simulation models, and model variations can significantly impact the evaluation of system performance.

Car-following models fall into two categories, fixed-form models with a small number of parameters and generalizable models with a large number of parameters. The former were originally developed for the analysis of emergent macroscopic traffic patterns, while the latter are used for learning vehicle-specific microscopic behavior.

### A. *Fixed-Form Car-Following Models*

Brackstone and McDonald review the five primary categories of traditional fixed-form behavior models: the Gazis-Herman Rothery model, the collision avoidance model, linear models, psychophysical models, and fuzzy logic-based models [23]. The Gazis-Herman Rothery [1] class of models predicts the vehicle acceleration based on the vehicle's current speed $s(t)$, the relative speed $r(t)$, the distance headway $d(t)$, and includes an estimated driver reaction time $T$:

$$a(t) = \alpha_1 s^{\alpha_2}(t) \frac{r(t-T)}{d^{\alpha_3}(t-T)} \tag{1}$$

where $t$ is the time at which predictions are made. The driver reaction time is typically on the order of 0.8 to 2 s [5], and the three parameters $\alpha_{1:3}$ are constants that must be calibrated against a dataset.The values for $\alpha_2$ and $\alpha_3$ can vary significantly depending on the dataset [23].

The collision avoidance model predicts the safe following distance required to avoid frontal collisions using similar features and several calibration constants [24]–[26]. The linear model of Helly extends the Gazis-Herman Rothery model with past accelerations and has five calibration constants, $\beta_{1:5}$:

$$a(t)=\beta_1 r(t-T)+\beta_2[d(t-T)+\beta_3+\beta_4 s(t-T)+\beta_5 a(t-T)]. \tag{2}$$

These first three classes assume very specific forms for the driver response behavior. They are limited in their ability to generalize across contexts and datasets without recalibration.

Concerns about generalization motivated the construction of the last two categories, the psychophysical and fuzzy-logic-based models. Psychophysical car-following models assume that drivers will react once a certain threshold is met [28]. These thresholds are often defined in terms of relative distances or headways, and once exceeded, cause the modeled driver to decelerate or accelerate until the relative speed meets the desired zero-difference. The inclusion of several thresholds allows for more expressive models, but the model is still limited by the assumed form of the resulting response.

Fuzzy-based-logic models, based on fuzzy-set theory, have been applied to the Gazis-Herman Rothery model [29]. These techniques allow one of several parameterizations of the model to be identified based on natural language-based driving rules. Fuzzy inference produces a fuzzy output set, which parameterizes the model in a probabilistic fashion. The primary difficulty with fuzzy logic models is the selection of membership functions [2].

The intelligent driver model (IDM) is a more recent fixed-form car-following model that produces collision-free trajectories [30]. The IDM generates acceleration predictions based on the idea that each driver balances the desire of a certain speed with the desire to maintain a safe distance to the lead vehicle. The relevant equations are:

$$d_{\text{des}} = d_{\text{min}} + \tau \cdot s(t) - \frac{s(t) \cdot r(t)}{2 \cdot \sqrt{a_{\text{max}} \cdot b_{\text{pref}}}} \tag{3}$$

$$a(t) = a_{\text{max}} \left[ 1 - \left( \frac{s(t)}{s_{\text{max}}} \right)^4 - \left( \frac{d_{\text{des}}}{d(t)} \right)^2 \right] \tag{4}$$

where $d_{\text{des}}$ is the desired distance to the lead vehicle, $d_{\text{min}}$ is the minimum distance to the lead vehicle that the driver will tolerate, $\tau$ is the desired time headway, $a_{\text{max}}$ is the ego-vehicle's maximum acceleration ability, $b_{\text{pref}}$ is the preferred deceleration, and $s_{\text{max}}$ is the driver's desired free-flow speed.

### B. *Generalizable Car-Following Models*

Traditional fixed-form behavior models make assumptions about the driver responses. Recent research has sought to overcome these restrictions using more general models. Angkititrakul, Ryuta, Wakita, *et al.* evaluate the performance of Gaussian mixture regression and piecewise autoregressive exogenous models for car-following, but they do not compare against traditional fixed-form models [31]. Panwai and Dia developed car-following models using two-layer feedforward neural networks accepting relative speed, relative distance, desired speed, and current speed as inputs [17]. Their models were shown to outperform a wide variety of traditional fixed-form models in micro- and macroscopic evaluation, but since the targeted application was macroscopic traffic simulation and not prediction, they did not provide performance results in the absence of measurements over the prediction period. Both Gaussian mixture regression and neural network models were further investigated by Lefèvre, Sun, Bajcsy, *et al.*, who implemented the models as maximum-likelihood acceleration predictors and compared against traditional baselines [19]. The Gaussian mixture regression and feedforward neural network models were found to produce comparable results. None of these models used recurrent neural networks.

## III. Problem Definition

We seek to produce probability distributions over accelerations in one-dimensional driving scenes. These models only

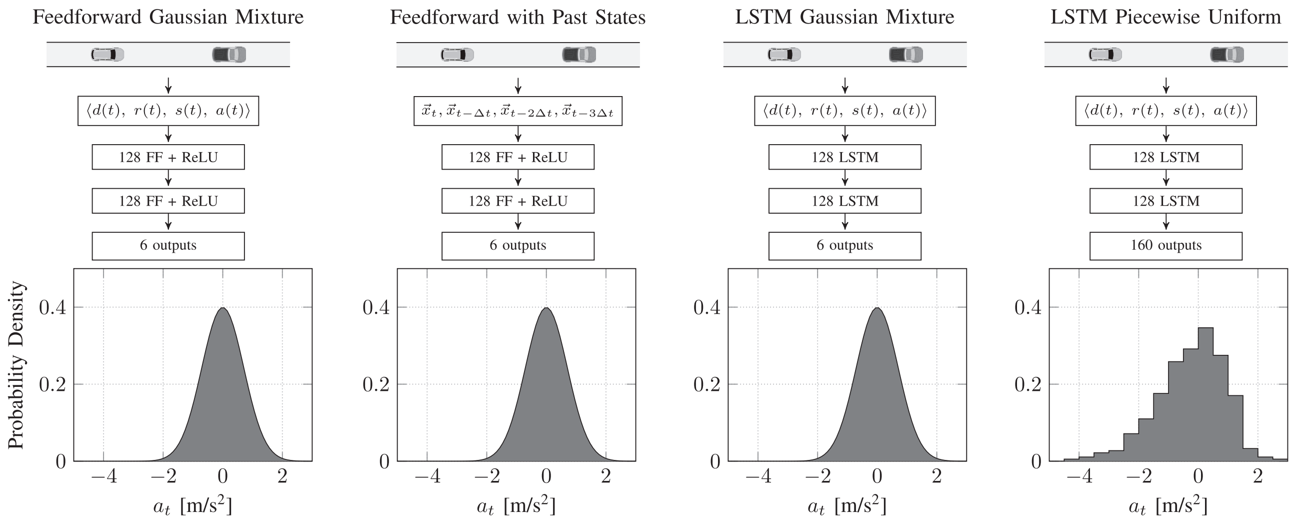
Feedforward Gaussian Mixture
Feedforward with Past States
LSTM Gaussian Mixture
LSTM Piecewise Uniform
$\langle d(t),\ r(t),\ s(t),\ a(t)\rangle$
$\vec{x}_t, \vec{x}_{t-\Delta t}, \vec{x}_{t-2\Delta t}, \vec{x}_{t-3\Delta t}$
128 FF + ReLU
128 LSTM
6 outputs
160 outputs
Probability Density
$a_t$ [m/s$^2$]

Fig. 1. Network structures and resulting distribution families.

consider the ego-vehicle, whose actions are being predicted, and the lead vehicle. For a given ego-lead vehicle configuration, the system state $\vec{x}(t)$ at time $t$ is represented by:

$$\vec{x}(t) = \langle d(t),\ r(t),\ s(t),\ a(t)\rangle \tag{5}$$

where $d(t)$ is the headway distance between the vehicles, $r(t)$ is their relative speed difference, $s(t)$ is the ego-vehicle speed, and $a(t)$ is the ego-vehicle acceleration.A human behavior model maps a state history to a distribution over the acceleration in the next time-step, from which samples can be drawn to propagate simulated trajectories. Given a sampled ego acceleration at the next time step, $a(t + \Delta t)$, the ego position and speed can be updated according to:

$$\begin{aligned} s(t + \Delta t) &= s(t) + a(t + \Delta t)\Delta t \\ y(t + \Delta t) &= y(t) + s(t + \Delta t)\Delta t \end{aligned} \tag{6}$$

where $y$ is the absolute longitudinal position of the ego-vehicle.The ego state can then be updated by propagating the lead vehicle according to its true trajectory. The underlying model implementation can take many forms. In this work we use neural networks, as described in the next section.

## IV. Neural Driving Models

Neural networks are universal function approximators that traditionally consist of an input layer, multiple hidden layers, and an output layer. Vector-valued inputs are fed into the input layer and are manipulated by a set of linear transforms and nonlinear activations as they traverse the hidden layers to the output layer. Here, the network outputs parameterize probability distributions over the acceleration in the next time-step.

Neural networks are typically trained with stochastic gradient-descent to optimize a scalar loss function [32]. In the context of driver modeling, this loss function is the predicted log-likelihood of the true vehicle behavior. Several hyperparameters, including learning rate and regularization strength, must be chosen to produce high-quality networks.

Neural networks are very efficient to use once trained. The resulting models permit fast-time Monte Carlo simulation but are sophisticated enough to produce physically realistic dynamics. Recurrent neural networks extend traditional feed-forward networks with internal cycles. This provides recurrent neural networks with an internal state that is capable of tracking sequences of information and learning relevant temporal features. The long short-term memory (LSTM) neural network architecture is widely used due to its resistance to the vanishing gradient problem [22], [33].

All networks were trained to predict parameters for distributions over future acceleration values given the vehicle state, $\vec{x}(t)$. Ten-fold cross-validation was performed and extracted metrics were averaged across folds. The training data was divided into twelve-second trajectory segments, the first two seconds of which were used to initialize the internal state of the LSTM networks. All networks were implemented in Torch7 [34] based on Karpathy's *char-rnn* package [35] and Zaremba's LSTM implementation [36].

Two forms for the parameterized distribution family were considered: Gaussian mixture and piecewise uniform. The network structures and parameterized distribution families are shown in Fig. 1.

### A. Gaussian Mixture

The first parameterized acceleration distribution family is the Gaussian mixture:

$$p(x) = \sum_{i=1}^{n} w_i, \mathcal{N}(x \mid \mu_i, \sigma_i^2) \tag{7}$$

where $w_i$, $\mu_i$, and $\sigma_i$ are the weight, mean, and standard deviation for the $i$th mixture component. The neural network's output layer produces the component weights, means, and standard deviations. The weight parameters are obtained through softmax activation to ensure they sum to one, and exponential activations on the standard deviations ensure they are positive. The outputs for the component means are not passed through any activation because they can theoretically take on any value. This approach was initially proposed by Bishop [37], and has been successfully applied in fields such as speech synthesis [38].

### B. Piecewise Uniform

The second form for the parameterized acceleration distribution is the piecewise uniform distribution. The domain over observed acceleration values is discretized into a finite number of bins. In this case, the network's output layer represents probabilities that the next acceleration value will fall within each bin. A probability distribution is produced by performing a softmax activation over the output layer:

$$P(\text{bin}_i \mid x) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{8}$$

where $z_i$ is the output corresponding to the $i$th bin and $n$ is the number of bins. The loss function is the Negative Log-Likelihood criterion, evaluated according to the predicted probability density associated with the true acceleration value:

$$p(a \mid x) = p(a \mid \text{bin})P(\text{bin} \mid x) \tag{9}$$

where the probability density is uniform within each bin. To sample from the piecewise uniform distribution, a bin can be selected based on the generated probabilities, and an exact acceleration value can be sampled from a uniform distribution within the bin.

## V. Models

Four neural models and one baseline fixed-form model were implemented. All models take states as inputs and produce distributions over vehicle accelerations over the next time-step.

The feedforward Gaussian mixture network (FF) maps the current state $\vec{x}(t)$ to a Gaussian mixture. A second feedforward network $(\text{FF} \times 4)$ also produces a Gaussian mixture, but it takes the four most recent states as input, and thus has more information available with which to make predictions. Two LSTM models were implemented, one that outputs a Gaussian mixture (LSTM GM) and one that outputs a piecewise uniform distribution (LSTM PU). Both take only the current state $\vec{x}(t)$ and must learn a hidden state representation.

The expressiveness of neural networks typically scales with their depth and number of hidden units. This enhanced expressivity can lead to overfitting and is often associated with a large computational cost [39]. These competing considerations must be weighed when selecting a network architecture. In this work, each network was given the same architecture, consisting of two hidden layers with 128 units. In the recurrent networks, the hidden units are LSTM cells, while in the feedforward networks the hidden units apply affine transforms followed by ReLU activations [40]. The experimental section further discusses the hyperparameters used for each network.

The intelligent driver model (IDM) was included to illustrate the difference between fixed-form models and generalizable models. Lefèvre, Sun, Bajcsy, *et al.* showed that the IDM outperforms several other low-parameter car-following models in predicting vehicle speeds over various time horizons [19]. The Levenberg-Marquardt algorithm [42] was used to learn the remaining IDM parameters through a nonlinear least-squares fit. These parameters are summarized in Table I. It should be noted that the IDM performance would likely improve if these parameters were learned for individual drivers, but this was not done in this study due to limited data.

TABLE I
Learned Parameters for the IDM

| $d_{min}$ | $T$ | $b_{pref}$ | $s_{max}$ | $a_{max}$ |
|---|---|---|---|---|
| 5.249 m | 0.918 s | 3.811 m/s$^2$ | 17.837 m/s | 0.758 m/s$^2$ |

## VI. Methodology

The five models described above are evaluated to compare their ability to predict distributions over accelerations and produce realistic trajectories. This section describes the dataset and methodologies used in the experiments.

### A. Dataset

Experiments used on the Next Generation Simulation (NGSIM) dataset for Interstate 80, which contains 15 minutes of vehicle trajectory data collected using synchronized digital video cameras providing the vehicle lane positions and velocities over time at 10 Hz [43]. The Interstate 80 dataset covers an area in the San Francisco Bay Area approximately 500 m in length with six mainline lanes, including a high-occupancy vehicle lane and an onramp, as shown in Fig. 2. Traffic density transitions from uncongested to full congestion and exhibits a high degree of vehicle interaction as vehicles merge on and off the highway and must navigate in the nearly-congested flow. The dataset was collected by the Next-Generation Simulation program in 2005 to facilitate automotive research and is publicly available.

The NGSIM dataset includes vehicle positions and velocities in the lane-relative frame. We use the reconstructed NGSIM I-80 dataset from 4:00 to 4:15 due to its corrected kinematic inconsistencies, extreme acceleration values, and switching vehicle IDs [41], [44]–[46]. All training was conducted on zero-centered, normalized data segmented into twelve-second trajectories, two seconds of which were used to initialize the LSTM networks.

### B. Hyperparameter Selection

Neural networks are very sensitive to the training hyperparameters, which include the number and size of the layers, the learning rate, and learning rate decay. For this work, several hyperparameter settings were tested, and selections were made according to the performance achieved by trained models on a validation set. In all networks, a learning rate of $4 \times 10^{-3}$ was used and was cut in half every three training epochs. The Gaussian mixture networks, both LSTM and feedforward, consist of two 128-neuron hidden layers and a two-component Gaussian mixture in the output layer.

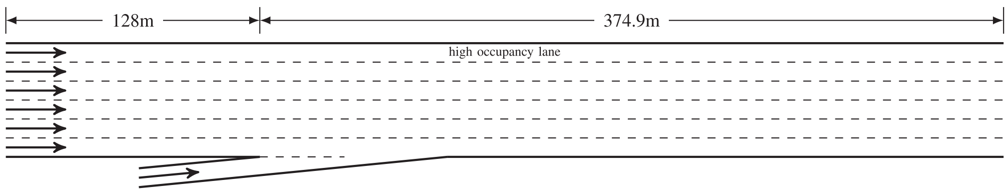

Fig. 2. Data source highway schematic for the NGSIM Interstate 80 dataset.

The piecewise uniform network architecture consists of two hidden layers containing 128 neurons and an output layer containing 160 neurons. The neurons in the output layer correspond to bins in the predicted acceleration distribution with the range of possible acceleration values spanning $-5$ to 3 m/s$^2$. Constant width and constant frequency bin boundaries were averaged to find the bin sizes in the output distribution, which allows for finer discretization near common acceleration values, while maintaining reasonable bin sizes near the extremes. Smaller output layers were tested, but required coarser discretization, leading to unrealistic acceleration jumps when samples were drawn from wider bins. The performance of both LSTM networks was found to improve when a dropout of 25-percent was applied during training. Dropout is a form of regularization that helps prevent overfitting [39].

Once trained, each network was used to generate simulated trajectories. For each trajectory in the training set, the networks were supplied with two seconds of inputs corresponding to true state values, and then ten-seconds worth of simulated trajectories were sequentially sampled. The trajectories were generated through iteratively sampling from the output acceleration distributions and updating the state values using the kinematic relations in Eq. (6). Fifty simulated trajectories were generated for each true trajectory in the following analysis.

### C. Cross-Validation

To assess how the models generalize to unseen data, we used 10-fold cross-validation. The complete set of trajectories was partitioned into 10 folds, of which one fold served as the validation set and the remaining nine folds were used to train each of the models. This process was repeated 10 times, allowing each fold to serve as the validation set. The performance of each method was then computed by averaging scores across all ten folds on the basis of several metrics.

## VII. Experiments

After training, we assess whether our learned models are capable of generating artificial trajectories that exhibit the same characteristics as real trajectories. This similarity can be evaluated through quantitative criteria such as predictive accuracy and emergent behavior, as well as qualitative criteria such as smoothness. This section outlines several metrics to quantify this similarity and discusses the relative performance of each method. All propagation was at 10 Hz.

### A. Root Weighted Square Error

The predictive accuracy of a method can be measured using the discrepancy between the predicted speed values and the true values over various time horizons. The root-weighted square error (RWSE) captures the deviation of a model's probability mass from real-world trajectories [47]. The RWSE for $m$ trajectories for predicted variable $v$ over horizon $H$ is:

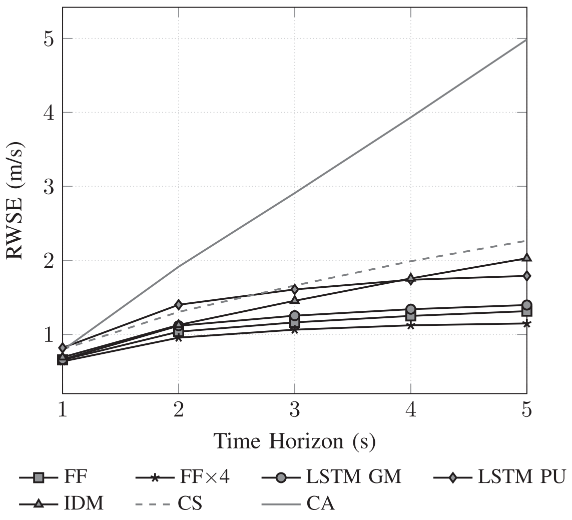

Fig. 3. Mean root-weighted square error in velocity predictions over different horizons. Error bars are omitted for clarity, but the margins are relatively small.

$$\text{RWSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\int_{-\infty}^{\infty} p(v)\cdot\left(v_H^{(i)} - v\right)^2 dv} \tag{10}$$

where $v_H^{(i)}$ is the true value in the $i$th trajectory at time horizon $H$ and $p(v)$ is the modeled density. Because the integral is difficult to evaluate directly, we estimated it with $n = 50$ simulated traces per recorded trajectory:

$$\text{RWSE} = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(v_H^{(i)} - \hat{v}_H^{(i,j)}\right)^2} \tag{11}$$

where $\hat{v}_H^{(i,j)}$ is the simulated variable under sample $j$ for the $i$th trajectory at time horizon $H$. The intelligent driver model produces point estimates, in which case the RWSE reduces to the root mean square error.

Fig. 3 shows the RWSE values for each model over prediction horizons between 1 and 5 seconds. The performance of constant speed (CS) and constant acceleration (CA) prediction methods are included as baselines. The Gaussian mixture methods achieve the best performance according to this metric, outperforming all other methods over all time horizons. The

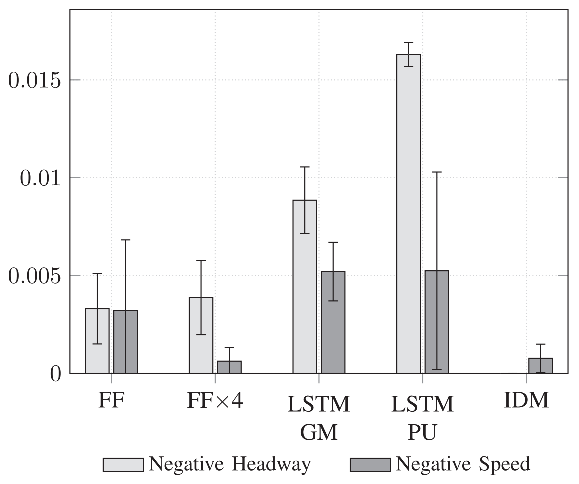

Fig. 4. Frequency of negative state values in generated trajectories. Error bars indicate standard deviation across folds.

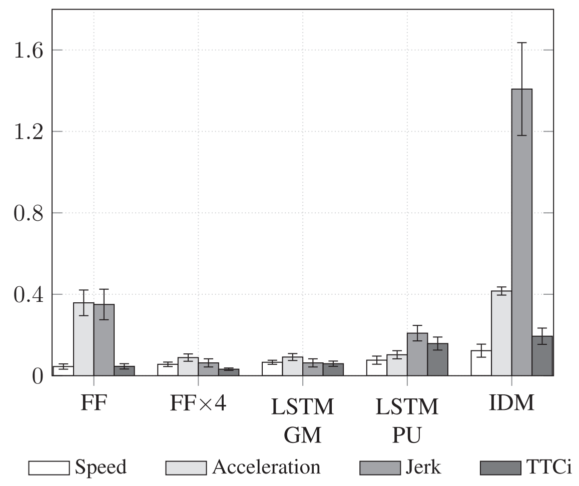

Fig. 5. Summary of KL divergence values for trajectories generated using each model.

feedforward network with multiple-state input slightly outperforms the other methods, indicating that the acceleration distributions it learns to predict are least likely to introduce error into speed estimates. IDM outperforms the piecewise uniform method over time horizons less than four seconds, and accumulates less error than constant speed predictions over all time horizons.

### B. Negative State Values

A second method for assessing model performance is to look for unrealistic behaviors in simulated trajectories. Two such values are negative distance headways and speed values, corresponding to collisions and vehicles in reverse, respectively. While there is a nonzero probability that acceleration values will be sampled from the neural models that lead to such negative state values, these events should be rare.

Fig. 4 summarizes the frequency with which these values are observed in simulated trajectories for each model. The values depicted are the mean values obtained over ten-fold cross-validation, with error bars indicating one standard deviation about the mean. Negative state values are observed in all neural models to an extent, and are most common in the trajectories generated by the piecewise uniform network. Interestingly, the feedforward models have fewer negative state values than both LSTM models. The IDM produces no collisions and few negative speed values in accordance with its design.

### C. Emergent Properties

The similarity of generated trajectories to true trajectories can also be measured by comparing the distributions over emergent properties in real and simulated trajectories. These quantities can be discretized into bins, and the distributions over these values can be approximated as piecewise uniform. From these empirical distributions, the Kullback-Leibler divergence can be evaluated. These values are shown in Fig. 5 for the speed, acceleration, jerk, and inverse time-to-collision (TTCi) [49], with low values indicating similarity. Time-to-collision quantifies the ratio of range to range-rate for vehicles in a car-following context. Its inverse is often used to avoid singularities when the relative speed approaches zero.

According to these metrics, the FF $\times$ 4 and LSTM GM networks are superior to the other models and achieve nearly equal performance. The feedforward network with a single-state input generates trajectories with the least realistic acceleration and jerk profiles among the neural models. In addition, there is a large discrepancy between the IDM empirical acceleration and jerk distributions and the true distributions, largely due to the artificially smooth nature of its trajectories.

The feedforward network with single-state input generates the same output as the FF $\times$ 4 and LSTM GM models and could be expected to share their advantages. However, this is not the case, as its acceleration and jerk distributions are biased toward values that are small in magnitude. In fact, many of its trajectories exhibit extended stretches where the predicted acceleration values stay close to zero. Predicting that a driver's acceleration will remain near zero if it was close to zero at the previous time step is a reasonable strategy if no other information is available, but it illustrates a clear shortcoming of the feedforward network: it cannot observe and follow trends.

Fig. 6 shows examples of real trajectories and corresponding artificial trajectories. The LSTM trajectories are noticeably smoother than the feedforward trajectories, which qualitatively illustrates the advantage of having access to state values from multiple time steps. This reflects what we would expect in realistic driving scenarios where the interactions between vehicles are more complicated than the car-following scenarios studied in this paper. In lane changes, for example, we would expect the acceleration of the ego-vehicle to change gradually to achieve a desired following distance instead of jumping instantaneously. Models without access to information from multiple time steps are less adept at accounting for these more complicated phenomena, and therefore have trouble capturing the smooth, oscillatory nature of the true trajectories.

### D. Jerk Sign Inversions

The similarity between the smoothness and oscillatory behavior of the true and simulated trajectories can be quantified by

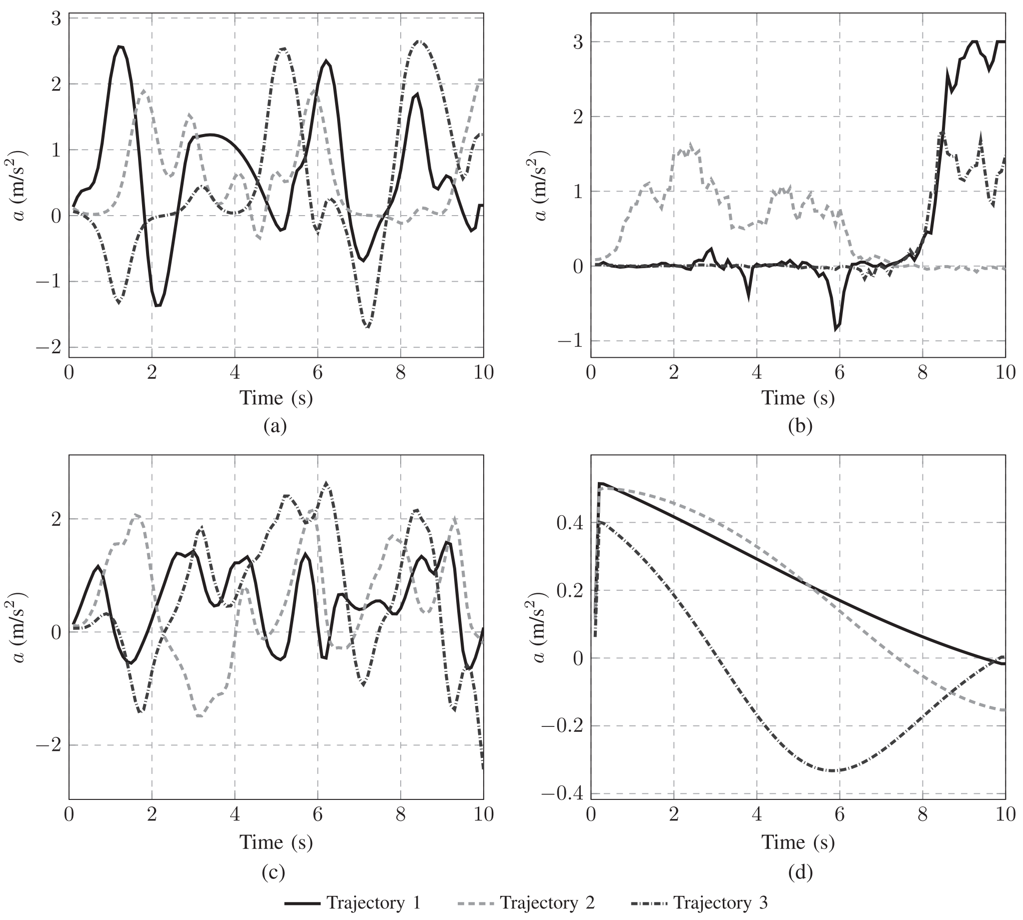

Fig. 6. Visual comparison of simulated and real trajectories. Trajectories created by the piecewise uniform network and the feedforward network with four-state input are qualitatively similar to those generated by the LSTM Gaussian mixture network.

TABLE II
JERK SIGN INVERSIONS PER TRAJECTORY

| True | FF | FF×4 | LSTM GM | LSTM PU | IDM |
|---|---|---|---|---|---|
| 12.51 ± 0.83 | 47.18 ± 0.94 | 14.05 ± 0.95 | 14.02 ± 0.38 | 17.45 ± 0.44 | 1.76 ± 0.06 |

the average number of jerk sign inversions per trajectory. Very smooth trajectories will have few jerk sign inversions, whereas highly oscillating trajectories will have many. These quantities are listed in Table II.

As expected, the number of sign inversions in FF trajectories far exceeds the number of sign inversions in true trajectories. IDM trajectories, on the other hand, exhibit far fewer sign inversions than the real trajectories. The LSTM and FF $\times$ 4 networks produce the most realistic jerk sign inversion counts.

The feedforward networks can be viewed as approximations to recurrent networks, with the ability to remember information over timescales equivalent to the number of states contained in their input. The results above suggest that knowledge of only one previous state is insufficient for the task of generating smooth acceleration profiles, but the expansion of the input to include just four previous states allows the feedforward network to match the performance of the LSTM networks. Hence, recurrent networks learn to track only very recent information in creating their acceleration predictions.

## VIII. UNDERSTANDING NETWORK BEHAVIOR

In addition to evaluating the predictive performance of the LSTM networks, it is also important to understand which spatial and temporal features they track in order to generate predictions. Fig. 7 illustrates some of the reactions that individual network neurons have in response to input state values. These activations exhibit responses to both the state values themselves, and the temporal variation in state values.

Neuron activation is most affected by the input acceleration, and knowing a previous time history of acceleration values is very important in predicting future acceleration values. Presumably, though, the network should also use the other inputs. Fig. 8 illustrates an experiment that was devised in order to determine whether this is the case.

As mentioned previously, before propagating simulated trajectories, the networks are supplied with two seconds of state values from real trajectories. In this experiment, within a subset of trajectories contained in the first validation set, the $d$ and $s$ values were translated independently such that the lowest value over the two seconds preceding propagation corresponds to some varied minimum value. By moving these state values closer to zero, the network is forced to generate accelerations that prevent the state values from going negative.

These experiments were performed using a Gaussian mixture LSTM network and minimum state values varied between

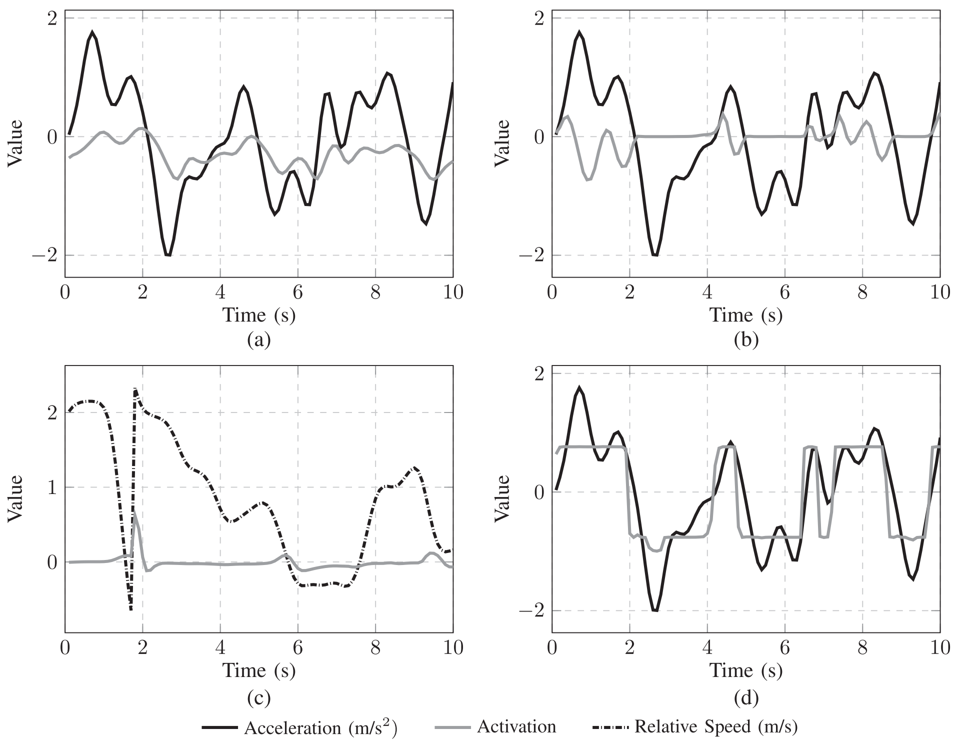

Fig. 7. Visualization of activations. In (a), the variation in activation values tracks the variation in acceleration values, but at a delay, which is evidence that the network remembers previous acceleration values. In (b), the neuron activations follow the jerk values when the acceleration is positive. Because jerk values are not passed into the network, they must be found through the temporal variation in acceleration values. In (c), the neuron strongly activates due to an abrupt change in relative speed. These jumps are due to lane changes by vehicles ahead of the ego-vehicle. In (d), the neuron saturates with a sign consistent with the sign of the acceleration values.

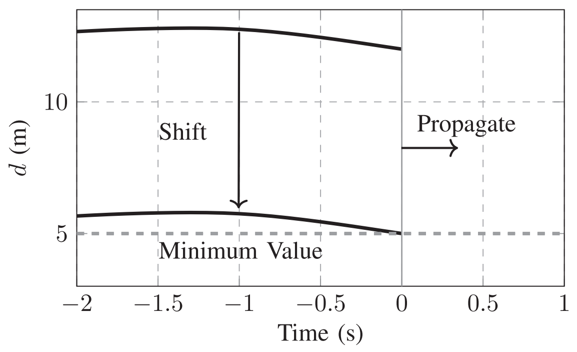

Fig. 8. Illustration of the shift in primed state values. In this case, the distance to lead vehicle is shifted such that the minimum value is 5 m.

0 and 5 in increments of 0.5. The fraction of negative headway distance, $d$, and speed, $s$, values were extracted from the resulting trajectories in order to quantify the effect of the shifted state inputs. These metrics are shown in Fig. 9.

The network does not strongly react to small $d$-values, with a near-monotonic decay in performance as the minimum state value is decreased. While the mean acceleration values output by the network do decrease slightly as the minimum $d$-value is decreased, the response is small in magnitude and insufficient to avoid negative $d$-values in many cases. On the other hand, the network appears to respond strongly to variations in speed. It is able to generate the accelerations needed to avoid negative speeds. In fact, Fig. 9(b) illustrates that the mean acceleration value in propagated trajectories increases nearly linearly as the minimum $s$-value is decreased.

There are a multitude of factors that could contribute to this discrepancy in network responsiveness. One key source likely lies in the data used to train the network—around 24% of the speed values in the dataset are 5 m/s or less, whereas only 1% of the $d$ values are 5 m or less. This means that the network has significantly fewer opportunities to learn how human drivers react to small headway distances. Another contributing factor could be the relative complexity of the relationship between acceleration and distance to the lead vehicle. Acceleration has a direct effect on speed, but its effect on headway distance is more indirect, as $d$ is also influenced by the ego-vehicle speed and the behavior of the lead vehicle.

## IX. Conclusion

This study has demonstrated the ability of Long Short-Term Memory networks to effectively generate predictions for vehicle acceleration distributions. Networks trained to predict Gaussian mixture and piecewise uniform distributions over future acceleration values exhibited performance on par with or superior to baseline methods as captured by evaluation metrics such as the RWSE and Kullback-Leibler divergence. Furthermore, the recurrent neural networks were shown to generate trajectories that replicate much of the qualitative behavior that is present in real vehicle trajectories. While the piecewise uniform network performed reasonably well, the Gaussian mixture network demonstrated an advantage in all respects. Additionally, it was shown that feedforward networks can closely match, and in some cases slightly exceed, the performance of LSTM

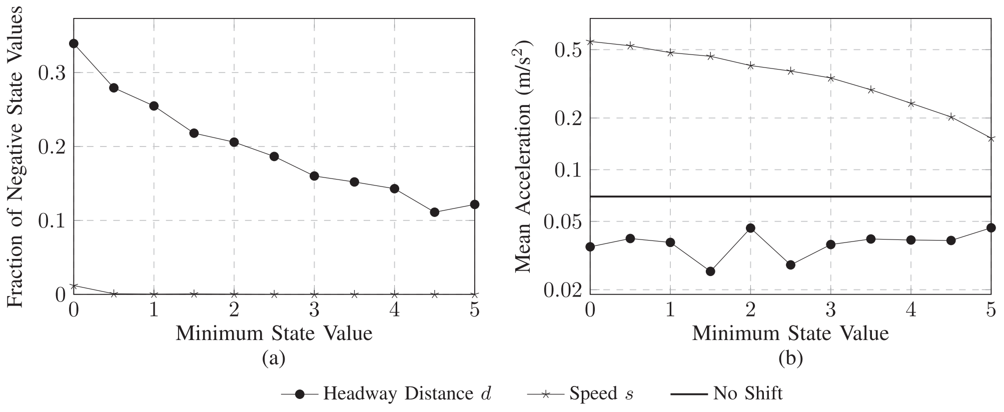

Fig. 9. Experimental results for shifted state values. In (a), the fraction of negative state values are shown as the minimum headway distance and speed are varied. In (b), the mean acceleration value in propagated trajectories is plotted against the minimum state values. The mean acceleration for trajectories, where no shift occurred is represented by the solid line.

networks, provided they are supplied with enough information about previous states. Thus, the LSTM networks seem to rely on recent acceleration trends in generating their predictions. Finally, a study of network behavior revealed that the networks respond to both the current state values and the variation in state values over time.

There are many potential avenues for extending this work. First, it should be noted that the dataset used for this project is not representative of all traffic conditions, so further study is needed to evaluate the neural driving models. The prevalence of negative state values, especially negative distances to the lead vehicle, in the propagated neural network trajectories can be addressed by augmenting the network input with more features or by augmenting the training data with more trajectories that include small speed or distance headway values. In addition, this work can be extended to include a 2-dimensional representation of the local driving scene, which would require the generation of predictions on both acceleration and turn rate. This problem should require the identification of even more complex features, in which case deep recurrent neural networks may provide greater benefits. The code associated with this paper is publicly available, and can be found at https://github.com/sisl/LSTM-acc-predict.

## Acknowledgment

The authors gratefully acknowledge the helpful comments from anonymous reviewers.

## References

[1] R. E. Chandler, R. Herman, and E. W. Montroll, “Traffic dynamics: Studies in car following,” *Oper. Res.*, vol. 6, no. 2, pp. 165–184, 1958.
[2] 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.
[3] 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.
[4] P. Ranjitkar, T. Nakatsuji, and A. Kawamua, “Car-following models: An experiment based benchmarking,” *J. Eastern Asia Soc. Transp. Studies*, vol. 6, pp. 1582–1596, 2005.
[5] P. Bonsall, R. Liu, and W. Young, “Modelling safety-related driving behaviour-impact of parameter values,” *Transp. Res. A, Policy Pract.*, vol. 39, no. 5, pp. 425–444, 2005.
[6] G. Agamennoni, J. Nieto, and E. Nebot, “Estimation of multivehicle dynamics by considering contextual information,” *IEEE Trans. Robot.*, vol. 28, no. 4, pp. 855–870, Aug. 2012.
[7] T. Gindele, S. Brechtel, and R. Dillmann, “Learning context sensitive behavior models from observations for predicting traffic situations,” in *Proc. IEEE Int. ITSC*, 2013, pp. 1764–1771.
[8] T. Wheeler, P. Robbel, and M. J. Kochenderfer, “Traffic propagation models for estimating collision risk,” in *Proc. IEEE Int. ITSC*, 2015, pp. 262–267.
[9] F. Damerow and J. Eggert, “Risk-aversive behavior planning under multiple situations with uncertainty,” in *Proc. IEEE Int. ITSC*, Sep. 2015, pp. 656–663.
[10] J. Eggert, F. Damerow, and S. Klingelschmitt, “The foresighted driver model,” in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2015, pp. 322–329.
[11] M. Bahram, C. Hubmann, A. Lawitzky, M. Aeberhard, and D. Wollherr, “A combined model- and learning-based framework for interaction-aware maneuver prediction,” *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 6, pp. 1538–1550, Jun. 2016.
[12] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proc. ICML*, 2009, pp. 609–616.
[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proc. Adv. NIPS*, 2012, pp. 1–9.
[14] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” ArXiv preprint arXiv:1512.03385.
[15] G. E. Dahl, D. Yu, L. Deng, and A. Acero, “Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition,” *IEEE Trans. Audio, Speech, Language Process.*, vol. 20, no. 1, pp. 30–42, Jan. 2010.
[16] J. Hongfei, J. Zhicai, and N. Anning, “Develop a car-following model using data collected by ‘five-wheel system’,” in *Proc. IEEE Int. ITSC*, 2003, pp. 1–6.
[17] S. Panwai and H. Dia, “Neural agent car-following models,” *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 1, pp. 60–70, Mar. 2007.
[18] A. Khodayari, A. Ghaffari, R. Kazemi, and R. Braunstingl, “A modified car-following model based on a neural network model of the human driver effects,” *IEEE Trans. Syst., Man Cybern., A, Syst. Humans*, vol. 42, no. 6, pp. 1440–1449, Nov. 2012.
[19] S. Lefèvre, C. Sun, R. Bajcsy, and C. Laugier, “Comparison of parametric and non-parametric approaches for vehicle speed prediction,” in *Proc. ACC*, Jun. 2014, pp. 3494–3499.
[20] Q. Liu, B. Lathrop, and V. Butakov, “Vehicle lateral position prediction: A small step towards a comprehensive risk assessment system,” in *Proc. IEEE Int. ITSC*, Oct. 2014.
[21] P. Boyraz, M. Acar, and D. Kerr, “Signal modelling and hidden Markov models for driving manoeuvre recognition and driver fault diagnosis in an urban road scenario,” in *Proc. IEEE Intell. Veh. Symp.*, 2007, pp. 987–992.
[22] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[23] M. Brackstone and M. McDonald, “Car-following: A historical review,” *Transp. Res. F, Traffic Psychol. Behav.*, vol. 2, no. 4, pp. 181–196, 1999.

[24] E. Kometani and T. Sasaki, "Dynamic behaviour of traffic with a non-linear spacing-speed relationship," in *Proc. Symp. Traffic Flow Theory*, 1958, pp. 105–119.

[25] A. D. May, *Traffic Flow Fundamentals*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1990.

[26] R. F. Benekohal and J. Treiterer, "CARSIM: Car-following model for simulation of traffic in normal and stop-and-go conditions," *Transp. Res. Rec.*, vol. 1194, pp. 99–111, 1988.

[27] W. Helly, "Simulation of bottlenecks in single-lane traffic flow," in *Proc. Symp. Traffic Flow Theory*, 1959, pp. 207–238.

[28] R. Michaels, "Perceptual factors in car following," in *Proc. Int. Symp. Theory Road Traffic Flow*, 1963, pp. 44–59.

[29] S. Kikuchi and P. Chakroborty, "Car-following model based on fuzzy inference system," *Transp. Res. Rec.*, vol. 1365, pp. 82–91, 1992.

[30] 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.

[31] P. Angkititrakul, T. Ryuta, T. Wakita, K. Takeda, C. Miyajima, and T. Suzuki, "Evaluation of driver-behavior models in real-world car-following task," in *Proc. IEEE ICVES*, 2009, pp. 113–118.

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[33] A. Graves, M. Liwicki, S. Fernández, 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.

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

[35] 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

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

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

[38] H. Zen and A. Senior, "Deep mixture density networks for acoustic modeling in statistical parametric speech synthesis," in *Proc. IEEE ICASSP*, 2014, pp. 1–5.

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

[40] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. ICML*, 2010, pp. 1–8.

[41] M. Montanino and V. Punzo, *Reconstructed NGSIM I80-1. COST ACTION TU0903—MULTITUDE*, 2013.

[42] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Numerical Analysis*. Berlin, Germany: Springer-Verlag, 1978, pp. 105–116.

[43] J. Colyar and J. Halkias, "US highway 80 dataset," Federal Highway Admin. (FHWA), Washington, DC, USA, Tech. Rep. FHWA-HRT-06-137, Dec. 2006.

[44] M. Montanino and V. Punzo, "Trajectory data reconstruction and simulation-based validation against macroscopic traffic patterns," *Transp. Res. B, Methodol.*, vol. 80, pp. 82–106, 2015.

[45] M. Montanino and V. Punzo, "Making NGSIM data usable for studies on traffic flow theory: Multistep method for vehicle trajectory reconstruction," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1, no. 2390, pp. 99–111, 2013.

[46] 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," *Transp. Res. C, Emerging Technol.*, vol. 19, no. 6, pp. 1243–1262, 2011.

[47] J. A. Cox and M. J. Kochenderfer, "Probabilistic airport acceptance rate prediction," in *Proc. AIAA Model. Simul. Technol. Conf.*, 2016, pp. 1–9.

[48] R. E. Caflisch, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, vol. 7, pp. 1–49, 1998.

[49] V. E. Balas and M. M. Balas, "Driver Assisting by Inverse Time to Collision," in *Proc. World Autom. Congr.*, 2006, pp. 1–15.

**Jeremy Morton** received the B.S. degree in aerospace engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2014. He is currently working toward the Ph.D. degree in the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA.

He is a member of the Stanford Intelligent Systems Laboratory. His research interests include deep learning and driver behavior modeling.

**Tim A. Wheeler** received the B.S. in aerospace engineering from the University of California, San Diego, La Jolla, CA, USA, in 2013. He is currently working toward the Ph.D. degree in the Department of Aeronautics and Astronautics, Stanford University, Stanford, CA, USA.

He is a member of the Stanford Intelligent Systems Laboratory. His research interests include the validation and optimization of active automotive safety systems.

**Mykel J. Kochenderfer** received the B.S. and M.S. degrees in computer science from Stanford University, Stanford, CA, USA, and the Ph.D. degree from the University of Edinburgh, Edinburgh, U.K.

He is an Assistant Professor of aeronautics and astronautics with Stanford University. He is the Director of the Stanford Intelligent Systems Laboratory, conducting research on advanced algorithms and analytical methods for the design of robust decision-making systems.