CS231n Project Report
Deep Learning of Spatial and Temporal Features for Automotive Prediction

Jeremy Morton and Tim Allan Wheeler
Stanford University
Stanford CA, 94305 USA
{jmorton2, wheelert}@stanford.edu

Abstract

Vehicle behavior models and motion prediction are critical for advanced safety systems and safety system validation. This paper studies the effectiveness of convolutional recurrent neural networks in predicting action profiles for vehicles on highways. Instead of using hand-selected features, the neural network is given an image-like representation of the local scene. Convolutional neural networks and recurrence allow for the automatic identification of robust features based on spatial and temporal relations. Real driving data from the NGSIM dataset is used for the evaluation, and the resulting models are used to propagate simulated vehicle trajectories over ten-second horizons. Prediction models using Long Short Term Memory (LSTM) networks are shown to quantitatively and qualitatively outperform baseline methods in generating realistic vehicle trajectories. Predictions over driver actions are shown to depend heavily on previous action values. Efforts to improve performance through inclusion of information about the local sscene proved unsuccessful, and will be the focus of further study.

1. 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.

This paper describes neural probabilistic action models trained 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. Convolutional recurrent neural networks learn relevant spatial and temporal features directly from an image-like input, removing the need for feature engineering and automatically maintaining an internal state over time. 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.

2. Related Work

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

Recent work has sought to automate the learning of general driver models 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 turnrate using random forests over contextual features [7]. Wheeler, Robbel, and Kochenderfer used Bayesian networks to generate predictions over acceleration and turnrate for free-flow, following, and lane-change context classes [8]. Damerow and Eggert planned using a probabilistic risk map generated based on 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.

The use of contextual features in a driving model comes with several drawbacks. Certain features are useful only in certain contexts, so effective action prediction with a single feature set is difficult. Different models have traditionally been developed for different driving contexts, but this is undesirable. Secondly, hand-specified features often include inconsistencies and corner cases. For instance, driving behavior is highly affected by relations to the lead vehicle, but the definition of lead vehicle begins to break down as another vehicle begins to cross over from another lane, there is a lane split, or an imminent merge. In the case when there is no lead vehicle, the contextual features are either: (1) set to an arbitrary large value, which can lead to fitting issues for common Gaussian models; (2) imputed, requiring a fair degree of computation and producing unreliable results; or (3) specially treated as missing or unknown. Finally, it is difficult to encode the specifics of nuanced driving interactions, abstract spatial relations, and temporal behavior. Driver aggressiveness, small intention-portraying actions prior to a merge, and overtaking require following detailed inter-vehicular relations over several frames with variable duration, and coupled with arbitrary road geometry, are extremely difficult to encode.

An ideal action model produces a realistic action distribution for any possible traffic scenario. Accomplishing this in a single model using hand-coded features is difficult. We instead use deep learning to automatically extract spatial and temporal features.

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 superior 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]–[20], lateral position prediction [21], and maneuver classification to provide inputs to a hidden Markov model [22].

3. Methods

3.1. Problem Statement

Let a scene s_t define the joint configuration of vehicles on a roadway at time t. A microscopic probabilistic action model relates contextual scene information extracted over H time steps to a probability distribution over the vehicle's action over the next time step, $p(a_t^{(i)} \mid s_{t-H:t-1})$, where $a^{(i)}$ is the action taken by the ith vehicle at time t. Sampling from the conditional action distribution for each traffic participant and propagating each vehicle over a small time step using a dynamics model leads to probabilistically valid successor scenes.

3.1.1 Vehicle State and Kinematics

We seek to produce probability distributions over vehicle accelerations and turnrates in highway driving scenes. These models take a general overhead scene representation centered on the ego vehicle, whose actions are being predicted, and the local scene configuration. A scene s_t at time t is an arrangement of n vehicles $\vartheta^{(i)}$, $i = 1, \ldots, m$ on a roadway topology. Vehicles are described by four-tuples $\vartheta^{(i)} = \langle x, y, v, \theta \rangle$ containing two global position coordinates, vehicle speed, and heading angle. Given an acceleration a_t and turnrate ω_t, a vehicle's state can be propagated over one time step Δt according to:

$$\begin{aligned} v_{t+\Delta t} &= v_t + a_t \Delta t \\ x_{t+\Delta t} &= x_t + v \Delta t \cos \theta_t \\ y_{t+\Delta t} &= y_t + v \Delta t \sin \theta_t \\ \theta_{t+\Delta t} &= \theta_t + \omega_t \Delta t \end{aligned} \tag{1}$$

The model used to predict the acceleration and turnrate can take many forms. In this work we use deep convolutional neural networks with LSTM layers.

3.1.2 Probabilistic Action Models

An action model defines a probability distribution over a single vehicle's action space, $p(a_t^{(i)} \mid s_{t-H:t-1})$. The action variables in this work are the acceleration, a, and turnrate ω over the next tenth-second time step, corresponding to the traditional throttle/brake and steering wheel driving inputs. Both the acceleration and turnrate were extracted using finite-difference derivative approximations, for instance $a_{t+1} = {}^{(v_{t+\Delta t} - v_t)}/_{\Delta t}$. The goal of this work is to learn the conditional probability distribution $p(a_t^{(i)} \mid s_{t-H:t-1})$ from data.

3.2. Network Architecture and Training

This work uses convolutional recurrent neural networks to obtain parameters to a probability distribution over a vehicle's action space. We introduce an image-like data input based on an over-head view. Different components in the image correspond to different types of spatial data. We currently use three components, analogous to an RGB image, but more could be used in the future.

The image is extracted in an ego-relative frame and is fed to the neural network. The image is first processed by a convolutional neural network, consisting entirely of convolutions, batch normalization layers, and ReLU activations. This procedure allows for learning spatial features and automatically reduces the image to a smaller feature space. This reduced feature space is processed through fully connected recurrent layers. These layers use long-short term memory units [23] to allow for temporal information to be stored and used. The final output is a vector of probability distribution parameters. We are currently using a multivariate Gaussian mixture model. For n components one needs $6n$ outputs: two for the component means, two for the component variances, one for the component correlation coefficient, and one for the component weight. A depiction of the network structure is given in Fig. 1.

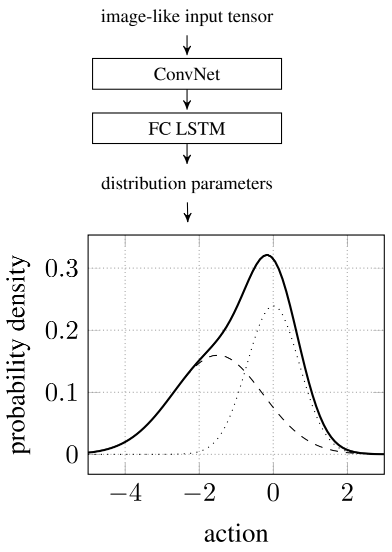


Figure 1: The neural network structure. An image-like input is processed by a convolutional neural network and then by a set of fully connected LSTM layers. The network produces parameters for a probability distribution over the vehicle's action.

3.2.1 Training the Feature Extractor

The image-like inputs used in this project are of size 32×64 with three color channels. If this representation was used to directly learn the output action distribution, the network would need to learn the relevance of all 6,144 input values in generating its predictions. To simplify the learning task of the LSTM, a convolutional neural network is used to extract relevant spatial information from the image-like representation and encode that information in a lower-dimensional feature space.

Hinton and Salakhutdinov [24] showed that it is possible to use a deep neural natework as an autoencoder for data dimensionality reduction. This technique requires the construction of a network with "encoder" and "decoder" sections. The encoder section takes the input data and transforms it into a low-dimensional code. The "decoder" section takes the low-dimensional code and attempts to reconstruct the network input.

In our problem, the image-like scene representation is used as input to the autoencoder network and transformed via a series of convolutions. Strided convolutions are interspersed within the convolutional layers, which serve to reduce the height and width of the data being passed through the network. At its narrowest point the network consists of two 8×4 filters, which are then transformed via upconvolution into an output with the same dimensions as the input. The loss function used to train the network is given by:

$$L = \frac{1}{2}||\mathbf{I} - \hat{\mathbf{I}}||^2 + \frac{1}{2}\lambda||\mathbf{W}||^2 \tag{2}$$

where $\mathbf{I}$ contains the pixel values from the input image, $\hat{\mathbf{I}}$ represents the set of values in the output layer, and λ is a regularization parameter applied to the network weights $\mathbf{W}$.

Once trained, the narrowest point in the network will contain a condensed feature vector $\mathbf{f} \in \mathbb{R}^{(4\times8\times2)}$, which is smaller than the input image by a factor of 96. If the values contained in $\mathbf{f}$ are able to capture much of the information contained in the full input, then they can be used as a surrogate for the full set of values contained in $\mathbf{I}$. Figure 2a shows an example of an image-like input from a validation set, with Fig. 2b showing the reconstructed image produced by a trained autoencoder network. While there are certainly many differences between the two images, the reconstructed image is able to reproduce quite a bit of information from the original image, such as the approximate location of vehicles and lane markers.

3.2.2 Training the LSTM

The compressed feature representation from a trained autoencoder can be used to generate predictions over driver actions by outputting parameters to a multivariate Gaussian mixture distribution. While a simple feedforward architecture could be used for this, such a network structure would only be capable of making predictions conditioned on the current highway scene. Since driver actions depend not only on their current surroundings, but also the evolution of those surroundings over time, we would instead like to condition our predictions on information from multiple time steps. To allow for this, we use LSTM layers, which are capable of tracking temporal variations in their inputs.

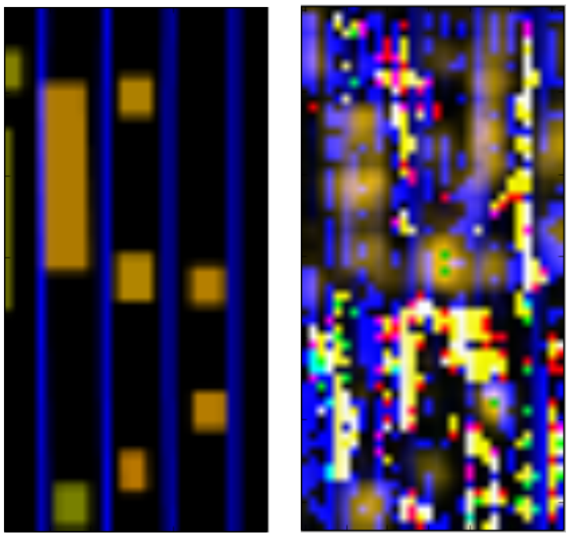

(a) Input (b) Reconstructed Image

Figure 2: Comparison of input image from validation set and image reconstructed by autoencoder. The ego-vehicle is positioned in the top-center of the image.

To train the recurrent layers of the network, an image is run through the autoencoder, and its compressed representation is extracted. Since no direct information about ego-vehicle acceleration and turnrate is contained in the image-like inputs, we concatenate the previous acceleration and turnrate to the feature vector, and feed this 66-component input into the LSTM network. The network output layer contains values corresponding to weights, means, variances, and correlation coefficients. The weights are run through a softmax activation to ensure that they sum to one, the variances are run through an exponential activation to ensure that they are positive, and the correlation coefficients are run through a sigmoid activation to ensure that they take on a value between zero and one. Covariance matrices can then be constructed from the variances and correlations.

The LSTM networks are trained to minimize the negative log-likelihood of the probability density associated with the true acceleration and turnrate at the next time step:

$$L = -\log p\left(a_t^{(i)} \mid s_{t-H:t-1}\right) \tag{3}$$

where $a_t^{(i)}$ is the set of actions taken by vehicle i at time t and $s_{t-H:t-1}$ is a history of scenes over H time steps. To test how much the LSTM network learns to use the compressed feature representation in generating predictions relative to the recent accelerations and turnrates, a separate LSTM network will be trained that receives only the acceleration and turnrates as inputs. By comparing the performance of this network relative to the LSTM network with the full input, we can judge whether spatial information from the input images is helpful in predicting driver actions.

3.3. Baseline Methods

Four models are compared in our results below, two of which use deep learning. The static gaussian model is a simple baseline that predicts a constant Gaussian distribution over the acceleration and turnrate. It is the naive prediction in the case that no other information is given.

The linear Bayesian model is a conditional linear Bayesian network over acceleration, turnrate, and six features: vehicle class, length, width, speed, current acceleration, and current turnrate. Vehicle class has three categories: car, truck, and motorcycle, the labeling of which is provided in the NGSIM dataset. The model learns conditional univariate normal distributions over each model given its parents in the Bayesian network:

$$p\left(x \mid \mathrm{pa}(x)\right) = \begin{cases} \mathcal{N}\left(w_1^T \, \mathrm{pa}_c(x) + b_1, \sigma_1\right) & \text{for } \mathrm{pa}_d^{(1)}(x) \\ \mathcal{N}\left(w_2^T \, \mathrm{pa}_c(x) + b_2, \sigma_2\right) & \text{for } \mathrm{pa}_d^{(2)}(x) \\ & \vdots \end{cases}$$

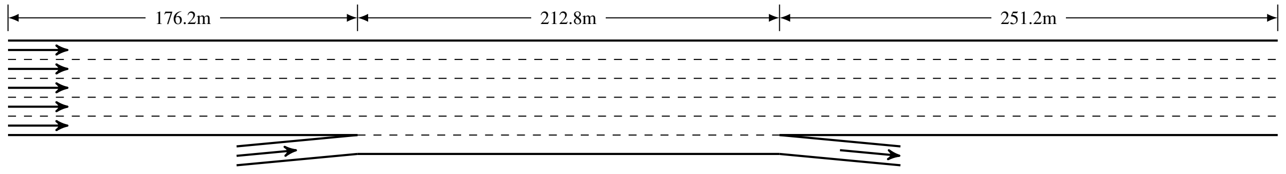
where $\mathrm{pa}_d^{(i)}(x)$ is the ith instantiation of the discrete parents of variable x and $\mathrm{pa}_c(x)$ is the vector of assignments to the continuous parents.

4. Dataset and Features

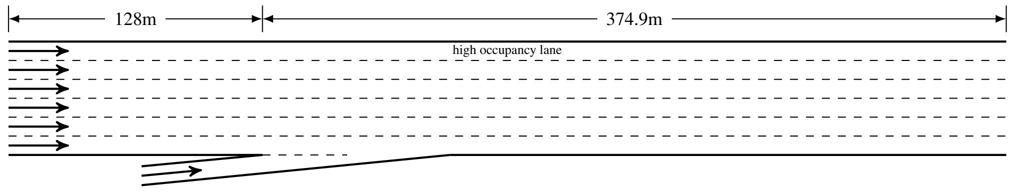
4.1. NGSIM Dataset

This work used real-world driving data obtained from the Next-Generation Simulation (NGSIM) US Highway 101 and Interstate 80 datasets [25], [26]. Each dataset consists of 45 minutes of vehicle trajectory data collected using synchronized digital video cameras providing the vehicle lane positions and velocities over time at 10Hz. The US Highway 101 dataset covers an area in Los Angeles, CA, approximately 640m in length with five mainline lanes and a sixth auxiliary lane providing highway entrance and exit. The Interstate 80 dataset covers an area in the San Francisco Bay Area approximately 500m in length with six mainline lanes, including a high-occupancy vehicle lane and an onramp. Schematics of both locations are given in Fig. 3. These datasets were collected by the Next-Generation Simulation program in 2005 to facilitate automotive research and are freely available.

Traffic density in the datasets 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. This and the datasets' complete scene description make these sources particularly useful for learning vehicle behavior based on image-like general model.



(a) Study Area for Highway 101



(b) Study Area for Interstate 80

Figure 3: Roadway topology for the NGSIM Highway 101 and Interstate 80 datasets.

The NGSIM datasets provide positions and velocities in the global frame, and lane boundaries were extracted from the provided CAD files. Vehicle headings were extracting assuming zero side-slip. Vehicle positions, headings, and velocities were smoothed using exponential moving averages according to the method described by Thiemann *et al.* [27].

4.2. Image-Like Input Representation

We use the 4:00 - 4:15 NGSIM dataset component for I-80, which gives us vehicle positions, estimated bounding boxes, and velocities at 10Hz. Headings assumed zero side-slip and were obtained using finite-differences in position, $\theta \approx \text{atan2}(y_{t+\Delta t} - y_t, x_{t+\Delta t} - x_t)$. Positions, velocities, and headings were smoothed according to the method described by Thiemann *et al.* [27]. Lane markings were extracted from a CAD file included in the NGSIM dataset.

Each image is set in the ego-vehicle's body frame, with the width of the image aligning with the vehicle's forward vector. All image components use the same size and projection parameters, these being the height and width in pixels (32×64), and the fore, rear, and side distance of the image's field of view relative to the ego vehicle (150, 40, & 25m).

Our current input representation uses three components. The first two components are the velocities of the pixel contents relative to the lateral and longitudinal orientation of the ego-vehicle, respectively. Empty pixels have zero velocity, and a maximum and minimum threshold are used to scale the velocity range between 0 and 255. Together these layers convey the relative positions of vehicles and their velocities, theoretically allowing the network to react to and reason about interrelations. The third component contains the roadway markings. This layer allows for arbitrary roadway structures to be used and avoids the need to add complicated pre-processing concerning lane merging and splitting, etc. The layer construction is shown in Fig. 4.

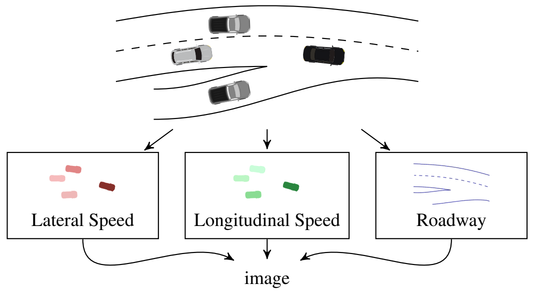


Figure 4: Image-like input tensor construction. Ego-relative image layers are extracted containing the lateral and longitudinal velocities and the road markings.

4.3. Feature Augmentation

While it is theoretically possible to train an action model which only uses the input image, performance was found to greatly improve by augmenting the input to the LSTM network with the current ego-vehicle acceleration and turnrate. These augmented values will generally be known whether the model is being used in simulation or on an autonomous car platform. This augmentation increases the input feature vector to have 66 dimensions.

5. Results and Discussion

The CNN was pretrained using the auto-encoding structure shown in Fig. 5 using Adam with a mini-batch size of 128. The initial learning rate was 0.01, with a decay of 0.95 after every epoch. Training was terminated after five epochs due to settled loss and time constraints.

The filter depth used for all non-bottleneck layers and the bottleneck filter depth were separately tuned. The final model used a general filter depth of 16 and a bottleneck filter depth of 2, as this was found to converge best. Several other values were investigated, as was using a fully-connected

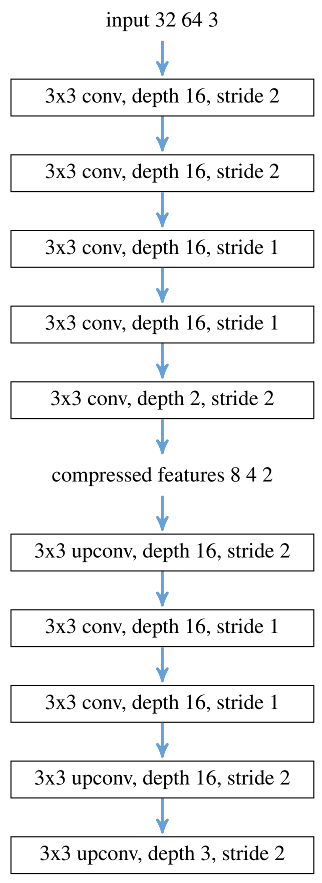

input 32 64 3
3x3 conv, depth 16, stride 2
3x3 conv, depth 16, stride 2
3x3 conv, depth 16, stride 1
3x3 conv, depth 16, stride 1
3x3 conv, depth 2, stride 2
compressed features 8 4 2
3x3 upconv, depth 16, stride 2
3x3 conv, depth 16, stride 1
3x3 conv, depth 16, stride 1
3x3 upconv, depth 16, stride 2
3x3 upconv, depth 3, stride 2

Figure 5: Network structure for the image-like input autoencoder. Batch normalization and ReLU activations were used after each layer.

layer in the bottleneck. Other values either did not converge to reasonable loss values or converged to where only the ego-vehicle's position was preserved in the image, and the other vehicles were left out.

The trained CNN was run over the dataset to pre-extract the 64-dimensional spatial feature vectors. This new dataset was used to train the LSTM networks, consisting of a variable number of fully-connected LSTM layers. Training used Adam with a mini-batch size of 10, as traces consist of 100 frames each. An initial learning rate of 0.005, with a decay of 0.75 after every epoch seemed to provide the best results within a small search over hyperparameter values. The final network used two LSTM layers with 128 cells in each layer.

5.1. Validation Likelihood

The first evaluation metric is the cross-validated likelihood of withheld data given the learned model. The validation likelihood serves as the core performance metric in maximum likelihood model selection [28]; it assesses the model's ability to generalize across datasets. A good model will have a high validation likelihood and will generally also have a high training likelihood.

Table 1: Validation log-likelihood for each model based on 10% of withheld data.

Method	Validation Log Likelihood
Static Gaussian	−31.33
Linear Bayesian	172.74
Pure LSTM	734.33
CNN + LSTM	374.51

The validation likelihoods are given in Table 1. The neural models show considerably higher validation likelihoods, indicating stronger generalization. The Pure LSTM model has the highest likelihood, but was also trained for considerably longer than the CNN + LSTM model due to time constraints.

5.2. Root-Weighted Square Error

The second validation metric, the root-weighted square error (RWSE), captures the deviation of a model's probability mass from real-world trajectories [29]. Where the cross-validated likelihood measures the immediate probability of successor frames, the RWSE measures the expected square deviation of particular variables in successor traces, thereby assessing the model's ability to act over longer time scales and with evolving traffic scenes. The RWSE is a natural extension to the root-mean square error, which is the mean deviation of a predicted trajectory from real-world examples. The models developed in this paper are distributions rather than maximum likelihood predictors, and it is important that the overall probability mass correctly reflect the true distribution over agents' actions. The RWSE for m trajectories for a predicted variable v at horizon H is:

$$RWSE_H = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\int_{-\infty}^{\infty} p(v)\cdot\left(v_H^{(i)} - v\right)^2 dv}, \quad (4)$$

where $v_t^{(i)}$ is the true value in the ith trajectory at time t and $p(v)$ is the modeled probability density. Because the integral is difficult to evaluate directly, we use Monte Carlo integration [30] with $n = 5$ simulated traces per real-world trajectory:

$$RWSE_H = \sqrt{\frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n}\left(v_H^{(i)} - \hat{v}_H^{(i,j)}\right)^2}, \quad (5)$$

where $\hat{v}_H^{(i,j)}$ is the simulated variable under sample j for the ith trajectory at the horizon H.

The root-weighted square error for speed versus horizon is given in Fig. 7. All three advanced models perform equally well for the first one and a half seconds, after which

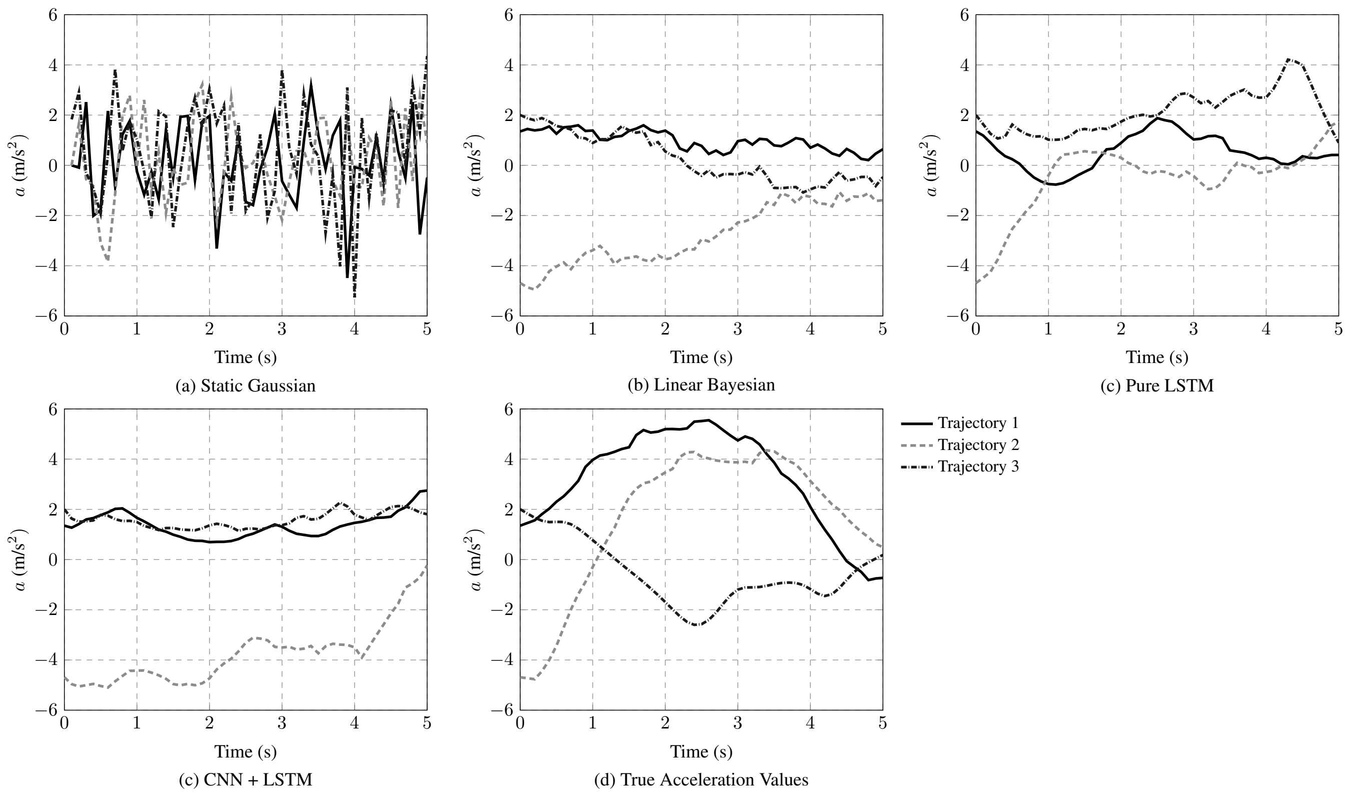


Figure 6: Visual comparison of simulated and real trajectories.

they begin to diverge in comparison to the static Gaussian model. The static Gaussian model, which takes the average action, overtakes the other models for long horizons. This suggests that the more complicated models follow inferred trends that take them farther away from the true trajectory. The Pure LSTM outperforms the CNN + LSTM in this analysis.

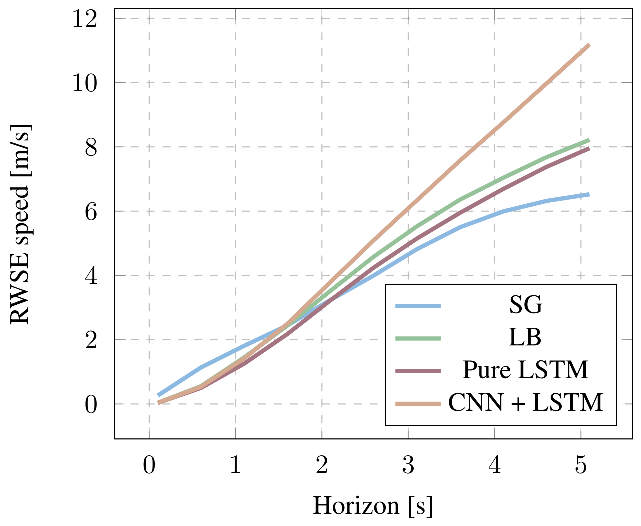


Figure 7: The root-weighted square error for speed vs. horizon. Notice the short-term superior performance for the three advanced models, but the eventual consistency of the static Gaussian.

5.3. Smoothness

The final validation metric concerns the emergent behavior of the vehicles under a particular model through simulation. Likelihood alone is insufficient to capture the performance of a stochastic process such as a probabilistic action model. The model must produce vehicle traces and overall driving behavior that is comparable to real-world driving. An emergent variable is a value extracted from a trace that is not explicitly fitted in the modeling process, such as collision frequencies and traffic flow rates. A distribution over emergent variables can be produced by simulation and compared to the distribution extracted from real world data. Matching distributions suggest model accuracy [31].

One such emergent variable is the comfort and smoothness of resulting trajectories, often described by the derivative of lateral or longitudinal acceleration: $\text{jerk}(t) = j_t = \dot{a}_t$ [32]. Humans tend to drive smoothly, so accurate driving models should produce similarly smooth trajectories. We used a simple smoothness measure in the form of jerk sign inversions. Counting the average number of jerk sign changes across a 10-second trajectory is easy to implement and gives a clear indication of the trajectory smoothness.

Table 2: Jerk sign inversions on simulated trajectories over the validation set.

Method	Jerk Inversion Count
Static Gaussian	64.38
Linear Bayesian	48.59
Pure LSTM	22.44
CNN + LSTM	19.98
True	14.06

Smoothness results over the validation set are given in Table 2. From these values it is clear that the LSTM models generate trajectories with a level of smoothness that most closely matches the data. This advantage can be largely attributed to the internal state of the recurrent networks, as they can observe trends in acceleration and turnrate values that allow them to predict action distributions with a higher level of certainty. This higher certainty translates into more narrow distributions, meaning samples drawn from these distributions are not likely to exhibit large jumps or excessive oscillation.

Figure 6 provides an illustration of the smoothness levels present in trajectories generated by each of the methods. The acceleration profiles generated by the static Gaussian jump around quite a bit, which can be attributed to the fact that the distribution needs to be wide enough to encompass all acceleration values observed within the training set. The Linear Bayesian method generates trajectories that are much smoother than the static Gaussian, but they still exhibit more oscillation than what we see in the real trajectories. The smoothness levels present in the LSTM trajectories seem to most closely match reality.

6. Conclusions

This work formulated a novel approach for learning neural driver models using deep neural networks. Image-like inputs were constructed from naturalistic driving data, and an autoencoder network was trained to extract relevant spatial features. A compressed feature space was then passed to an LSTM network, which used the temporal information in its inputs to generate distributions over driver actions. To our knowledge, this is the first work that uses convolutional and recurrent neural networks to model drivers in a two-dimensional driving context.

On the basis of quantitative metrics such as log-likelihood and qualitative measures such as trajectory smoothness, it was shown that LSTM-based methods generally outperform the baseline methods. Surprisingly, the LSTM network that received only acceleration and turnrate as an input seemed to perform better than the LSTM that also receives features from the image-like input. This may largely be due to the limited amount of time that was available to train the LSTM network with the larger input, since, at the very least, the network should be able to learn to ignore the other inputs if they are not useful. This will require further investigation.

Future work should continue to explore model structures and training methods for the both convolutional and LSTM networks. It would be advantageous to include more non-linearities, perhaps using a residual network [14], and it may be possible to further reduce the bottleneck to as few as 10 features. Future analysis should look at neural activations to identify what sort of relevant information is being extracted, and look at particular contexts such as merging and lane changing. The generative merit of thise models should be proven and emergent behavior, such as maintaining headway distance, should be demonstrated.

References

[1] R. E. Chandler, R. Herman, and E. W. Montroll, “Traffic dynamics: Studies in car following,” *Operations Research*, vol. 6, no. 2, pp. 165–184, 1958.

[2] S. Panwai and H. Dia, “Comparative evaluation of microscopic car-following behavior,” *Intelligent Transportation Systems, IEEE Transactions on*, 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,” *Journal of Transportation Systems Engineering and Information Technology*, vol. 12, no. 2, pp. 67–75, 2012.

[4] P. Ranjitkar, T. Nakatsuji, and A. Kawamua, “Car-following models: An experiment based benchmarking,” *Journal of the Eastern Asia Society for Transportation Studies*, vol. 6, pp. 1582–1596, 2005.

[5] P. Bonsall, R. Liu, and W. Young, “Modelling safety-related driving behaviour–impact of parameter values,” *Transportation Research Part A: Policy and Practice*, 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 Transactions on Robotics*, vol. 28, no. 4, pp. 855–870, Aug. 2012, ISSN: 1552-3098.

[7] T. Gindele, S. Brechtel, and R. Dillmann, “Learning context sensitive behavior models from observations for predicting traffic situations,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 1764–1771.

[8] T. Wheeler, P. Robbel, and M. J. Kochenderfer, “Traffic propagation models for estimating collision risk,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2015.

[9] F. Damerow and J. Eggert, “Risk-aversive behavior planning under multiple situations with uncertainty,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Sep. 2015, pp. 656–663.

[10] J. Eggert, F. Damerow, and S. Klingelschmitt, “The foresighted driver model,” in *IEEE Intelligent Vehicles Symposium*, 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 Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–13, 2016, ISSN: 1524-9050.

[12] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 609–616.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *ArXiv preprint arXiv:1512.03385*, 2015.

[15] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, 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 *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, vol. 1, 2003, pp. 346–351.

[17] S. Panwai and H. Dia, "Neural agent car-following models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 1, pp. 60–70, 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," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 42, no. 6, pp. 1440–1449, 2012.

[19] S. Lefèvre, C. Sun, R. Bajcsy, and C. Laugier, "Comparison of parametric and non-parametric approaches for vehicle speed prediction," *American Control Conference (ACC)*, pp. 3494–3499, Jun. 2014.

[20] J. Morton, T. A. Wheeler, and M. Kochenderfer, "Human driver acceleration predictions using recurrent neural networks, in preparation," *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[21] Q. Liu, B. Lathrop, and V. Butakov, "Vehicle lateral position prediction: A small step towards a comprehensive risk assessment system," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2014, pp. 667–672.

[22] 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 *IEEE Intelligent Vehicles Symposium*, 2007, pp. 987–992.

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

[24] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

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

[26] ——, "US highway 80 dataset," Federal Highway Administration (FHWA), Tech. Rep. FHWA-HRT-06-137, Dec. 2006.

[27] C. Thiemann, M. Treiber, and A. Kesting, "Estimating acceleration and lane-changing dynamics from next generation simulation trajectory data," *Transportation Research Records*, vol. 2088, pp. 90–101, 2008.

[28] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning*. Springer, 2001, vol. 1.

[29] J. A. Cox and M. J. Kochenderfer, "Probabilistic airport acceptance rate prediction," 2016.

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

[31] M. J. Kochenderfer, M. W. Edwards, L. P. Espindle, J. K. Kuchar, and D. J. Griffith, "Airspace encounter models for estimating collision risk," *AIAA Journal on Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 487–499, 2010.

[32] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, IEEE, 2011, pp. 163–168.

[33] S. Kullback and R. A. Leibler, "On information and sufficiency," *Annals of Mathematical Statistics*, vol. 22, pp. 79–86, 1951.