2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Deep Neural Networks for Markovian Interactive Scene Prediction in Highway Scenarios

David Lenz† and Frederik Diehl† and Michael Truong Le† and Alois Knoll‡

***Abstract*— In this paper, we compare different deep neural network approaches for motion prediction within a highway entrance scenario. The focus of our work lies on models that operate on limited history of data in order to fulfill the Markov property[1] and be usable within an integrated prediction and motion planning framework for automated vehicles. We examine different model structures and feature combinations in order to find a model with a good tradeoff between accuracy and computational performance. We evaluate all models with standard metrics like the negative log-likelihood (NLL) and evaluate the performance of each model within a closed-loop simulation. We find a neural network only operating on spatial features of the current state to have the best closed-loop prediction performance, despite the NLL suggesting otherwise.**

I. Introduction

Highly automated or autonomous vehicles are becoming reality: Every year, new demonstrations show the technical feasibility of automatically driving on highways, rural roads or even — a few vehicles — in urban traffic. One drawback of current systems is that the environmental conditions are controlled and the system drives very defensively. In dense traffic, especially during lane merging, this assumption can lead to long waiting times or driver interventions. Therefore, current research for trajectory planning is focusing on how to plan in such interactive and uncertain scenarios.

On the one hand, the field of Machine Learning has gained importance not only for computer vision, but also for prediction and even controlling the vehicle in an End-To-End fashion [1] to mimic human driving behavior. This approach seems promising as no modeling and architecture decomposition has to be done and only a large amount of training data is necessary, but – so far – has only been shown to be successful in very limited lane-following scenarios.

On the other hand, current planning approaches first predict the trajectories of all vehicles and then plan an optimal trajectory under known positions (or distributions) of other vehicles. This prediction can be very sophisticated, but always has the drawback that it must also predict the ego-vehicle with some default behavior. Thus, although capable of incorporating interactions between vehicles, it will not capture the influence of the ego-vehicle's behavior on other vehicles. An integrated prediction and planning approach should therefore be preferred in highly interactive scenarios like lane changing.

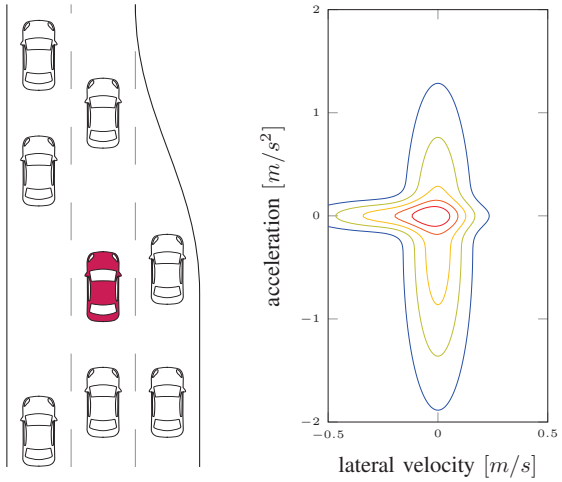


Fig. 1. Example for a traffic scenario with a probability distribution over the next action of the ego-vehicle. The action that the vehicle will keep the lane and keep the velocity has the highest probability, but a chance for a lane change can also be seen.

In this paper, we propose a trained deep neural network for motion prediction in highway (entrance) scenarios. Due to the selected features and network structure, it can be calculated very efficiently and only depends on the current state of all involved vehicles. This makes it applicable to be used within an interactive planning framework as proposed in our previous work in [2] or in planning methods based on Partially Observable Markov Decision Processes (POMDPs) [3]. The model is probabilistic and therefore also includes uncertainty in the prediction. A typical scene and resulting probability distribution over actions can be seen in Figure1.

The main contributions of this paper are:

- A comparison of different deep neural network structures, feature combinations and number of past inputs for probabilistic interactive vehicle prediction
- Techniques to improve the convergence during training and the stability of the closed-loop simulation
- Proposition of a novel model structure with a low-dimensional latent layer

II. Related Work

Predicting the actions of other traffic participants has been extensively researched in the past years as it is a key feature for autonomous driving and especially simulation of traffic scenes. To be able to predict the behavior of other traffic

†David Lenz, Frederik Diehl and Michael Truong Le are with fortiss GmbH, affiliated institute of Technische Universität München, Munich, Germany

‡Alois Knoll is with Robotics and Embedded Systems, Technische Universität München, Munich, Germany

[1]Markov property means that the system has no memory and only depends on the current state.

participants, like cars and trucks, motion models are needed. The challenge within this task lies in the complexity of a traffic scene due to the fact that the behavior of the traffic participants is highly correlated among each other and the model needs to predict the upcoming actions in a feasible time.

A survey by Lefèvre *et al.* [4] divides vehicle motion predictions into three subgroups ordered in their complexity: *physics-based*, *maneuver-based* and *interaction-aware* motion models, which are briefly described in the following:

Physics-based motion models depend only on the laws of physics, usually by assuming constant velocities and/or constant accelerations. They do not react to other traffic participants but can be influenced by external forces, like road inclination or wind velocity. The least complex models are linear motion models such as the constant velocity (CV) and the constant acceleration (CA) model in which the vehicles are only able to move in a straight line [5]. These models are very simple and can be extended to constant turn rate and velocity/acceleration models. More complex methods include vehicle dynamics such as lateral tire forces [6].

Maneuver-based motion models represent each traffic participant as a single entity which can execute maneuvers independently from others. Maneuver-based models use either an estimation of the intention or an estimation of the most likely trajectory. The basic idea of the latter is to cluster all trajectories into a set of motion patterns — the prototype trajectories — and try to predict the intended trajectory the vehicle is going to take by extracting and matching certain features using clustering methods [7]. As an alternative to trajectory prototypes, machine learning algorithms, including Support Vector Machines (SVMs), Relevance Vector Machines (RVMs) and Multi-Layer Perceptrons (MLPs) [8]–[10], are often used to predict the maneuver intention by measuring and evaluating certain features like vehicle state, road topology and the driver behavior.

Interaction-aware motion models expand the previously discussed maneuver-based motion models by interpreting the vehicles as non-independent entities. There exist two methods in literature on which such models are based. The first are again trajectory prototypes and the second are Dynamic Bayesian Networks (DBNs). The first methods select only trajectories where the risks for collisions with other vehicles is minimized by assuming that every driver is acting extremely safely [11]. Brand *et al.* [12] on the other hand use a form of DBNs — so called coupled Hidden Markov Models (HMMs) — to model the pairwise dependencies of multiple entities which can also be moving. This is computationally very expensive as the dependencies grow quadratically with the number of entities. One method to deal with this complexity is to model the dependency so that the ego-vehicle is influenced by the other traffic participants and not vice versa. This is called an asymmetric dependency [13]. Another method proposed by Gindele *et al.* [14] uses factored states to implement the interdependencies of the traffic participants and can therefore reduce the complexity. Another approach is to use an Expectation-Maximization algorithm (EM) to learn models integrated in a DBN [15]. This algorithm only depends on unlabeled observations and can learn different motion models. Bahram and Hubmann [16] implement an interaction-aware motion model by combining a learning- and model-based motion model. The assumption is made that all traffic participants obey the traffic law and drive very carefully so that risk is minimized. Any deviation from this assumption is only due to following the interaction-aware model. A gradient descent algorithm on a cost map is used to plan the next maneuver.

The proposed methods for modeling driver behavior have in common that more complex models need more computational resources. The fastest models are physics-based and maneuver-based motion models, however they do not take the influence of other traffic participants into account. On the other side the more complex models need more time for a prediction and are therefore not suitable for repeated usage. Often these methods can only model one specific driving style with their set of parameters and can only mimic other styles by changing them.

Besides these *classic* methods, new methods were developed based on machine learning algorithms which can overcome the problem of computationally complexity while being more accurate and modeling different driving styles. With the recent success of Deep Neural Networks in the fields of Computer Vision and Natural Language Processing, Morton *et al.* [17] also used a Recurrent Neural Network (RNN) [18] to model driver behaviors. Their work investigated the use of 4 different neural models to predict a distribution of the acceleration of cars in front of the ego-vehicle in a car-following scenario for the next time step. Among these networks they have used a simple feedforward network which takes the current state as input, a simple feed forward network which takes the 4 most recent states as input and a recurrent LSTM network. All these models map the current state to a Gaussian Mixture Model for predicting the longitudinal acceleration distribution. They showed that neural networks are outperforming baseline methods such as CV and CA in predicting the acceleration limited to a car-following scenarios over several seconds. In contrast to our work, only the longitudinal motion and a limited state space (vehicle and leader) is considered. Thus interactions based on the vehicles on neighboring lanes like opening gaps for lane change are not considered.

In this work, we aim to provide an interaction-aware prediction model that in contrast to most related work can be computed with limited resources and with limited number of past states. Our work does not claim to reach the performance of the best interactive scene predictions presented in this section, but is usable for interactive prediction and planning due to fast computation and the limited state space.

III. Motion Models using DNNs

In this section, we first present the problem definition for probabilistic motion prediction. Then we introduce different

features used for training the different deep neural networks in the subsequent subsection.

A. Behavior Prediction

We want to model the probability distribution over all possible actions a_c for vehicle c given the current state of the scene $\mathcal{S}$:

$$p(a_c|\mathcal{S}), \tag{1}$$

where each action can be split into a longitudinal action a_c^{long} and a lateral action a_c^{lat}. As longitudinal action, we use the vehicles' acceleration $\ddot{s}$ and for the lateral action we use the lateral velocity $\dot{d}$. By $\ddot{s}$ as a longitudinal action, we explicitly model the cars' restrictions in velocity change, and restrict our state space. By using $\dot{d}$ as a lateral action, we model the direct reaction of cars to steering. This also reduces integration errors, since the dataset does provide the longitudinal velocity, but only the lateral position.

A scene $\mathcal{S}$ consists of the states $\boldsymbol{x}_c$ of all vehicles, the road, and lane geometry and obstacle information. The state $\boldsymbol{x}_c$ of each vehicle contains the track coordinates (s, d), the velocity $v = \dot{s}$, and the last applied action a_c^{prev}.

We model the distribution as a two-dimensional Gaussian Mixture Model (GMM) of the form

$$p(a_c|\mathcal{S}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{a_c}|\boldsymbol{\mu}_k, \Sigma_k), \tag{2}$$

where the mixture coefficients π_k, the means $\boldsymbol{\mu}_k$ and the covariance matrices Σ_k are all dependent on the current scene state $\mathcal{S}$. This dependency is omitted for clarity. An example distribution from such a GMM with two mixtures is shown in Figure1.

Furthermore we can break down the mean and covariance to

$$\boldsymbol{\mu}_k = \begin{pmatrix} \mu_{\text{lon},k} \\ \mu_{\text{lat},k} \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma^2_{\text{lon},k} & \sigma_{\text{lon},k}\sigma_{\text{lat},k}\rho_k \\ \sigma_{\text{lon},k}\sigma_{\text{lat},k}\delta_k & \sigma^2_{\text{lat},k} \end{pmatrix}. \tag{3}$$

Thus, each of the K mixture component consists of 6 parameters $\{\pi_k, \mu_{\text{lon},k}, \mu_{\text{lat},k}, \sigma_{\text{lon},k}, \sigma_{\text{lat},k}, \rho_k\}$, which are themselves dependent on the scene $\mathcal{S}$. This dependency can be parameterized using any model. In this paper, we aim to model them using deep neural networks.

B. Loss function

In order to train the models, we minimize the negative log-likelihood as loss function:

$$L_{\text{nll}} = \sum_{\text{s}} \left\{ -\log \sum_{k=1}^{K} \pi_k^s \mathcal{N}(\boldsymbol{a}^s|\boldsymbol{\mu}_k^s, \Sigma_k^s) \right\}, \tag{4}$$

where s are the samples from the training set.

This is equal to maximizing the probability that action $\boldsymbol{a}_c$ is predicted given a scene state $\mathcal{S}^s$.

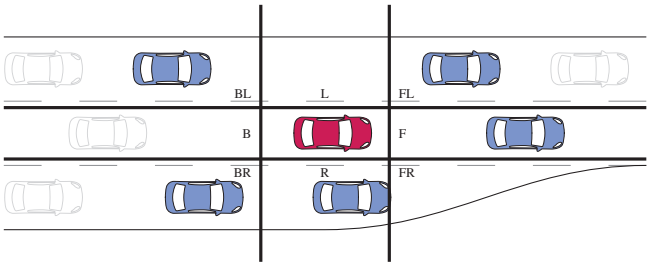


Fig. 2. Representation of the driving scene for feature extraction. A maximum of 7 vehicles will be considered, the closest one in each of the grid cells. All other vehicles (white) will be discarded for the prediction. For each of the blue vehicles a feature vector $f_{v,L}$, $f_{v,FL}$, ... will be added as an input to the neural network.

TABLE I

FEATURES EXTRACTED FOR A DRIVING SCENE

Feature	*Description*
	Ego-vehicle $\boldsymbol{f}_e$
$f_{e,v}$	The current velocity of the ego-vehicle
$f_{e,\boldsymbol{a}}$	Last performed lateral and longitudinal actions
	Surrounding Vehicles $\boldsymbol{f}_v$
$f_{v,e}$	A boolean whether a vehicle is present or not in the grid cell
$f_{v,\boldsymbol{x}_{lon}}$	Position and velocity in longitudinal direction, relative to the ego-vehicle
$f_{v,\boldsymbol{x}_{lat}}$	Lateral position w.r.t the ego lane
$f_{v,\text{TTO}}$	Time until an overlap in longitudinal direction will occur. Overlap is a collision on ego lane, or same height on neighboring lanes.
$f_{v,\text{size}}$	Length and width of the vehicle
$f_{v,\text{type}}$	Vehicle type (motorcycle, car or truck)
	Road geometry $\boldsymbol{f}_r$
$f_{r,e}$	Booleans whether a lane is present to the left or right
$f_{r,l}$	Remaining length of each of the three lanes relative to the ego-vehicle. If the lane does not end, a large constant is used.

C. Features

For each driving scene, we extract three categories of features for each.

1) $\boldsymbol{f}_e$: All features related to the ego-vehicle
2) $\boldsymbol{f}_v$: All features related to the 7 most relevant other vehicles around the ego-vehicle (compare Figure2)
3) $\boldsymbol{f}_r$: All features related to road geometry

An example of a driving scene can be seen in Figure2 and a list of all used features can be found in Table I. An existing lane has a maximum length of 200m. Non-existing lanes have a length of 0m. All features for cars that do not exist are set to 0, except for the $f_{v,type}$ (set to -1), $f_{v,x_{lon}}$ and d (each set to 100) and $f_{v,TTO}$ (which is set to 10).

We found that including the rear vehicle into the features results in a good negative log-likelihood, but leads to worse results in the closed-loop simulation. We believe that, by including the rear vehicle, the model learns to adapt its behavior to the rear vehicle which, in the real data, reacts to the front vehicle. Once this model is exposed to a simulation it will fail to provide useful data since the rear vehicle is governed by the same model instead of real data.

D. Model Structure

To parameterize the GMM, we evaluated several architectures.

Since we prefer our models to fulfill the Markov property, our base model receives an input vector which is constructed out of the features mentioned in Table I and only depends on the current timestep. However, we also compare it to an identical model which receives multiple consecutive past timesteps and a recurrent neural network.

1) Fully Connected: The fully connected model, shown in Figure3, consists of a series of n fully connected layers with N_i neurons each. We also add dropout layers between each fully connected layer to ensure better regularization. A linear combination of the last layer is then used to parameterize the K Gaussian Mixtures. Here, the linear combination for each μ uses an identity activation function. The σs are learned in log-space, and therefore effectively use an exponential activation function. Since the correlation ρ has to be between -1 and 1, we use a *tanh* activation function. Lastly, due to the fact that all πs have to sum up to one, we use an identity activation function followed by a softmax function between all mixture components' πs. This follows Graves [18].

As already mentioned we also implemented a multi-timestep variant of the fully connected network. For this, the features of the last t timesteps are concatenated and the network is trained on them.

Our final network uses four layers of 400 units each, with an exponential linear unit [19] activation function. We drop each unit with a probability of 0.5. We use four mixtures.

In addition, we found several techniques to boost our performance. We regularized the variances of the GMMs' components by introducing an L2-penalty term for covariances deviating from 0.5. This was weighted with 0.01. We also introduce artificial noise in the feature of the last action and last velocity, adding noise according to a Gaussian distribution with zero mean and variance of a tenth of the total variance of that feature.

We also evaluated predicting the action over a longer time-period, and found that using a longer timespan significantly improves performance. We therefore train the network to predict the average acceleration over the next 2 s, and use this predicted value as the action for the next 0.1 s during the simulation. We vary only the training, and use the same evaluation for every prediction length. We also evaluated both shorter (0.1 s, 1 s) and longer (4 s, 8 s) periods but found that 2 s resulted in the best performance. We assume that a longer period of time decreases both the influence of noise and therefore allows for a more consistent prediction.

Note that these techniques did increase the negative log-likelihood, but resulted in a more robust closed-loop simulation.

2) Latent Model: Inspired by Variational Auto-Encoders [20], [21], we introduce a variation of the fully connected model described above. In this model, we assume that the motion of the cars depend on several hidden variables, so-called latents. Accordingly, we call this the Latent Model.

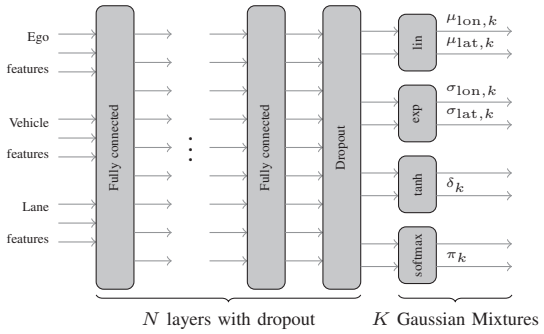


Fig. 3. Feature based Deep Neural Network

We augment the standard feed forward model above with an additional, low-dimensional layer. The output of this layer is a probability distribution. In practice, we use a normal distribution which is parameterized by the outputs of the previous layer:

$$z \sim \mathcal{N}(z|\mu_x, C_x) \tag{5}$$

To reduce computational complexity, we follow [21] in assuming that the latents are independent of each other and therefore use a diagonal covariance matrix C.

In contrast to [20] and [21], we train this Latent Model not on the task of reconstructing samples, but directly on the task of predicting actions. During training, we draw a single latent sample and use this to compute the loss which consists of both the negative log-likelihood (see equation (4)) and a Kullback-Leibler (KL) divergence term forcing the latents to approximate a standard-normal distribution:

$$L = L_{\text{nll}} + L_{\text{KL}} \tag{6}$$

$$L_{\text{KL}} = \frac{1}{n}\sum_n \frac{1}{l}\sum_{l \in \textit{latents}} KL\left[\mathcal{N}(\mu_n, C_n)||\mathcal{N}(0, I)\right] \tag{7}$$

$$= \frac{1}{2n}\sum_n \frac{1}{l}\left[||\mu_n||^2 + \sigma_n - \log\sigma_n - 1\right] \tag{8}$$

Unlike [21], we use the mean of the latents' μ and σ instead of the sum, which we found to result in significantly better performance. Using the original formula, we found the training to minimize the KL divergence loss and therefore predict the same output distribution for every sample. We assume that this occurs because the predicted actions are a) approximately Gaussian-distributed and b) extremely low-dimensional, in contrast to a reconstruction task on images. This makes minimizing the KL divergence loss the best way to reduce the total loss. The training is then unable to escape that local minimum.

We use the same parameters as above, and use twenty latents.

3) RNN: As a comparison, we also implemented a recurrent neural network. In the recurrent model we replaced the n fully connected layers of the feed forward model described above with LSTM layers. The rest of the network is unchanged from the base model. The input of the recurrent

Fig. 4. Example merging scenes from the NGSim Dataset for I-80, shown with the tracked bounding boxes and vehicle ids

network are the last t timesteps. It therefore violates the Markov property. However, it forms a useful baseline against which our Markov-property networks can be compared.

It uses the same parameters as the models before, except we only use two layers of 200 LSTMs each and no dropout.

E. Dataset

In order to train both models, we use data collected within the NGSIM project [22], specifically the dataset for the I-80. The dataset contains trajectory data for all vehicles tracked with a camera system in highway merging scenarios with a time resolution of 0.1 s. Some exemplary scenarios are shown in Figure4.

As Thiemann *et al.* [23] pointed out, the position, acceleration, and velocity data cannot be used directly as they contain unrealistic high values, probably because of noise and numeric differentiation. Therefore, we are using the same exponential smoothing algorithm proposed in their paper to preprocess the dataset, and are clamping the acceleration and velocity values to a maximum of $[-3, 3]$. We found that the proposed acceleration smoothing leads to wrong velocities when integrating, thus we used the time-discrete derivative of the smoothed velocity as acceleration. For all vehicles and each timestep, we computed all the features presented in the previous sections in order to generate a feature dataset with a total of $4.452.827$ samples. The dataset is splitted into a training (80%) and test (20%) set with respect to the individual cars. That means a car appearing in the training set does not appear in the test set. With this approach the correlation of the two sets are held low.

F. Implementation and Learning Procedure

We implemented the networks described in the previous subsections within tensorflow [24]. We used the Adam optimizer [25], for which we used default parameters except for the learning rate, which was set to 10^{-4} to stabilize our training. We trained each model for 50 epochs on the full training set. We found feature standardization to significantly improve our results.

IV. Evaluation

In this section we present the different evaluation techniques we used to assess the model quality.

A. Validation Metrics

Our evaluation procedure uses two of the metrics proposed by Wheeler *et al.* [26]. We first report our results using the likelihood of our targets given the features on an unknown dataset. This is easily available since it corresponds to our loss function in Equation 4. However, it is not easily interpretable.

Accordingly, we also report the Root-Weighted Square Error (RWSE) using a Monte Carlo integration over several trajectories, up to a certain horizon T. Similar to [26], we use $N = 20$ simulated trajectories for each of our M evaluated samples, but compare our results on a fixed horizon of up to $10s$. The RWSE is then defined as

$$\text{RWSE}_T = \sqrt{\frac{1}{MN}\sum_{i=1}^{m}\sum_{j=1}^{n}(v_T^{(i)} - \hat{v}_T^{(i,j)})} \tag{9}$$

where $v_H^{(i)}$ is the true value for sample i at horizon H and $\hat{v}_H^{(i,j)}$ is the prediction from the jth simulated trajectory for the same.

B. Interactive Prediction

We evaluate how well the models perform in a closed-loop simulation. We simulate all vehicles over a time horizon of $T = 10\,\text{s}$ in order to verify that the model is able to generate realistic traffic. In order to do that, we keep $\mathcal{S}$ fixed, while we calculate the features and sample an action from the GMM for each vehicle. Then, we apply all actions for the vehicles simultaneously and repeat until we reach our desired time horizon.

We compare the predicted and sampled trajectory with the true positions from the dataset and calculate the RWSE_T for the longitudinal displacement Δs, the lateral error Δd as well as the velocity error Δv.

C. Reference Models

In order to compare our probabilistic models with some baseline, we consider three different reference models:

1) Constant Velocity: The acceleration and lateral velocity are both considered to be zero. That means that the vehicle will drive with constant velocity and fixed lateral offset within its lane.

2) Constant Gaussian Mixture: We use the same mixture definition as for all different trained models, but remove the dependency on the current features. This makes it a constant GMM independent of the current scenario.

3) Intelligent Driver Model: As a commonly used driver model in the microscopic traffic simulation domain, we implement the Intelligent Driver Model (IDM) [27] for the longitudinal motion of the vehicle. It is deterministic and guaranteed collision-free. We discard the lateral motion of the vehicle for evaluation.

The equations to calculate the next acceleration for the given velocity v, difference velocity Δv and distance s to

the leader is:

$$\ddot{s}_i = a_{\max}\left[1-\left(\frac{v}{v_0}\right)^{\delta}\right] - a_{\max}\left(\frac{s^*(v,\triangle v)}{s}\right)^2,$$
$$s^*(v,\triangle v) = s_0 + \tau v + \frac{v\triangle v}{2\sqrt{a_{\max} b}} \quad (10)$$

with the parameters for desired velocity v_0, a time gap of τ, maximal acceleration $a_{\max}$, comfortable deceleration b, minimal distance to the front vehicle of s_0 and the acceleration exponent δ, a tuning parameter of the model.

We use the parameters from [17] which are calibrated on part of the NGSIM dataset. Thus, we set $v_0 = 17.8\,\mathrm{m/s}$, $\tau = 0.92\,\mathrm{s}$, $a_{\max} = 0.76\,\mathrm{m/s^2}$, $b = 3.81\,\mathrm{m/s^2}$, $s_0 = 5.249\,\mathrm{m}$ and $\delta = 2$ for all cars in the evaluation.

D. Feature Importance

In order to determine which features are important in the trained models, we took the following approach: First we calculate the loss L_{test} for the test set without alterations. Then, for each feature f_i, we swap the value of that feature of each test sample with a randomly chosen other sample. The resulting increase in loss $\Delta L_i = L_{\text{test}} - L_{\text{test}-f_i}$ is an indicator how strongly the network depends on said feature. Now we can sort the features by the loss increase ΔL_i.

In addition to switching single features, we also examined switching all features for a particular other vehicle in order to estimate the importance of that vehicle on the behavior of the ego-vehicle. Similarly, we proceed with the different timesteps for models with multiple timesteps as input.

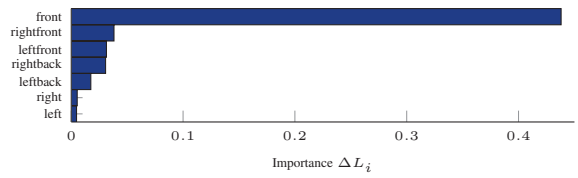
V. Experiments and Results

In this section we will present the results of the evaluation methods presented ealier. First, the importance of different features will be compared to find those features which may affect a neural network and result in bad performance in the closed-loop evaluation. After ensuring that only valid features are taken into account, the results of the closed-loop simulation are discussed w.r.t. RWSE. All quantitative results can be found in Table II including the negative log-likelihood, the RWSE, and the runtime. The presented runtime is an average over each test sample in the test set calculated in parallel in batch sizes of 1k on the GPU[2] and does not include the feature calculation. The runtime is very similar for all feed forward and latent networks, but the RNN has significantly higher computational requirements.

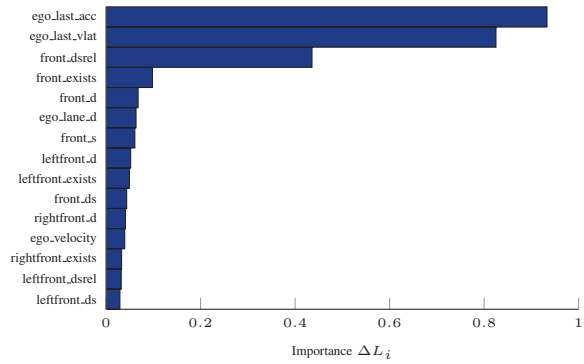
A. Feature Importance

The importance of certain features and their corresponding vehicle positions can be seen in Figure5 for the final trained model of the feed forward network with all vehicles and the last action included. Figure5(a) shows that the front-vehicle is by far the most important vehicle for predicting the next action. Beside the direct front-vehicle the other front-vehicles and the back-vehicles are equally significant. We believe the side vehicles are less important since their possible state-space is far smaller, and replacing them with other cars therefore results in only small variations.

[2]Timing evaluation was performed on an Nvidia GTX 780.



(a) Importance of surrounding vehicles



(b) Feature importance for the 15 most important features

Fig. 5. Feature importance for the Fully Connected network with all except the rear vehicle.

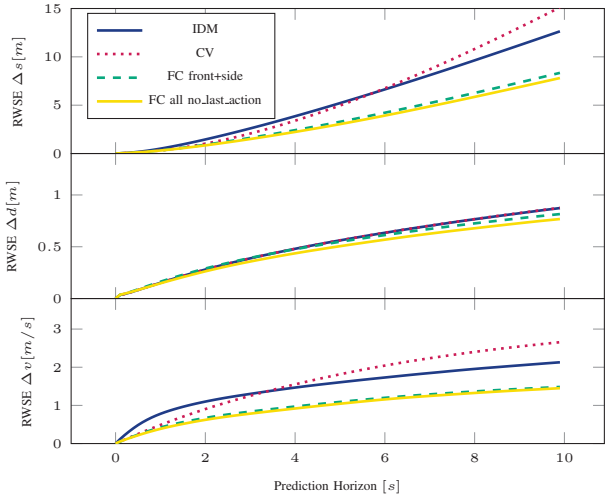


Fig. 6. RWSE results for the closed-loop simulation of two of our networks fully connected networks (ff), compared to baseline models, IDM and CV. From top to bottom, this figure shows the longitudinal, the lateral and the velocity error.

Within the features in Figure5(b), the last actions of the ego-vehicle ($f_{e,a}$) seem to be the most important features. These are followed by features regarding the front-vehicles: relative velocity of ego-vehicle and front-vehicle ($f_{F,\dot{x}_{lon}}$), an indicator if the front-vehicle exists ($f_{F,e}$), the lateral position ($f_{F,x_{lat}}$) and the longitudinal position ($f_{F,x_{lon}}$) of the front-vehicle. Interestingly, the back-vehicles appear to have a small impact on the behavior of the model.

B. Closed-Loop Performance

The performance of the closed-loop simulation for some of the networks can be seen in Table II and in Figure6. It

TABLE II

EVALUATION RESULTS AND CONFIGURATIONS FOR THE PRESENTED MODELS

Model	Vehicles	Last Action Feature	Prior Timesteps	Trained Horizon [s]	NLL	RWSE after $T = 10\,\mathrm{s}$ Δs[m]	Δd[m]	Δv[m/s]	Runtime [μs]
Constant Gaussian	-	-	-	2	-0.32	15.27	0.87	2.68	1.70
Constant Velocity	-	-	-	-	-	15.12	0.87	2.65	-
IDM	F	-	1		-	12.64	0.87	2.13	-
Fully Connected	all	yes	1	2	-1.49	8.24	0.80	1.47	3.95
	F+L+R	yes	1	2	-1.48	8.34	0.81	1.48	3.78
	F+FL+FR	yes	1	2	-1.48	8.36	0.81	1.49	3.59
	all	yes	1	0.1	-3.23[a]	12.23	1.15	2.27	3.82
	all	no	1	2	-0.85	**7.80**	**0.77**	**1.44**	3.32
	all	yes	4	2	**-1.68**	11.88	1.35	2.24	4.33
	all	no	4	2	-1.46	8.97	0.80	1.57	4.36
RNN	all	no	4	2	-1.55	12.27	1.57	2.32	32.94
Latent	all	yes	1	2	-1.42	9.28	0.82	1.67	3.44
	F+L+R	yes	1	2	-1.40	9.26	0.83	1.67	3.36

[a] The NLL cannot be compared to the other entries as the target variable is different from the average over $2s$

can be clearly seen that most of our networks perform better than the other baseline models regarding the longitudinal position error and the velocity error. The differences in the error increases with a longer prediction horizon. The improvement in lateral performance is not as big. However, it is clear that improvement could be achieved regarding the RWSE_T. Although the networks trained on multiple timesteps (including the RNN) produce a better NLL, they do perform worse in the closed-loop simulation.

Surprisingly, the feed forward model using all except the rear vehicle but without the last action performs best in the closed-loop simulation, although both NLL and intuition suggests otherwise. As already indicated in the previous section, we believe that the trained models with the last action included rely heavily on that feature. Therefore, small deviations (due to sampling of the GMM) in one timestep will lead to a bigger deviation in the next timestep of the simulation. By amplifying the error over multiple timestep the closed-loop simulation becomes unstable in some instances which leads to bigger errors. A similar reasoning can be applied to the models with multiple timesteps as input. The model without the last action must rely only on the spatial features around itself in order to determine the next best action. Even if a wrong action is chosen, the integration error for the velocity and position is small in one timestep and can be corrected in the next step. One major advantage of this result is that the last actions do not have to be included in the state vector in order to fulfill the Markov property.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented trained Gaussian Mixture Models parameterized by neural networks that are able to predict the motion of a group of vehicles in highway scenarios while only operating on the current state of the scene. They thus follow the Markov Property. The prediction is stable in a closed-loop fashion and able to generate reasonable motion over a time horizon of about $10\,\mathrm{s}$ in highway entrance scenarios. We found, that the NLL – a standard error measure for such problems – is not a very good indicator for performance in closed-loop simulations. Therefore, we plan to evaluate different error metrics during network training. Furthermore, we found measures like predicting the average acceleration over the next $2\,\mathrm{s}$, regularization of the covariance or removing the trailing vehicle to be successful to generate stable motion.

In future work, multiple models could be trained for the ego-vehicle and for other group of vehicles. These groups can be picked based on the observed driving style, that the car makes or other criterions. Separate models could also be trained for distinct situations, like car-following or lane-changing, with another model choosing which situation applies. This would further improve the accuracy of the trained models. Additionally this work is based on sensor data recorded by a statically mounted top view camera with full knowledge of the whole scene. For autonomous driving the models can be used to train on a set of features that are recorded by sensor events coming from the own ego-vehicle which would be more sophisticated due to problems like occlusions and limited view angle.

Furthermore, our next step is to combine the presented model with the planning framework from our previous work [2]. With the model presented in this paper, the convergence and quality of the generated policies should increase drastically.

REFERENCES

[1] M. Bojarski, D. Del Testa, D. Dworakowski, *et al.*, "End to end learning for self-driving cars", *ArXiv:1604.07316*, 2016.

[2] D. Lenz, T. Kessler, and A. Knoll, "Tactical Cooperative Planning for Autonomous Vehicles using MCTS", in *IEEE Intelligent Vehicles Symposium*, Gothenburg, Sweden, 2016.

[3] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic Decision-Making under Uncertainty for Autonomous Driving using Continuous POMDPs", in *IEEE Intelligent Transport Systems Conference*, 2014.

[4] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles", *Robomech Journal*, 2014.

[5] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking", *11th International Conference on Information Fusion*, pp. 1–6, 2008.

[6] R. Pepy, A. Lambert, and H. Mounier, "Path planning using a dynamic vehicle model", *Information and Communication Technologies*, vol. 1, pp. 781–786, 2006.

[7] D. Vasquez and T. Fraichard, "Motion prediction for moving objects: A statistical approach", *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, pp. 3931–3936, 2004.

[8] P. Kumar, M. Perrollaz, C. Laugier, *et al.*, "Learning-based approach for online lane change intention prediction", *Proc. IEEE Intelligent Vehicles Symposium*, pp. 797–802, 2013.

[9] B. Morris, A. Doshi, and M. M. Trivedi, "Lane change intent prediction for driver assistance: Design and on-road evaluation", *Proc. IEEE Intelligent Vehicles Symposium*, pp. 895–901, 2011.

[10] M. Garcia Ortiz, J. Fritsch, F. Kummert, and A. Gepperth, "Behavior prediction at multiple time-scales in inner-city scenarios", *Proc. IEEE Intelligent Vehicles Symposium*, pp. 1068–1073, 2011.

[11] A. Lawitzky, D. Althoff, C. Passenberg, *et al.*, "Interactive scene prediction for automotive applications", in *IEEE Intelligent Vehicles Symposium*, 2013.

[12] M. Brand, N. Oliver, and A. Pentland, "Coupled hiddenMarkov models for complex action recognition", *International Conference on Computer Vision and Pattern Recognition*, pp. 1–6, 1997.

[13] N. Oliver and A. Pentland, "Graphical models for driver behavior recognition in a SmartCar", *Proc. IEEE Intelligent Vehicles Symposium*, pp. 7–12, 2000.

[14] T. Gindele, S. Brechtel, and R. Dillmann, "A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments", *IEEE Intelligent Transport Systems Conference*, 2010.

[15] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning", *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[16] M. Bahram and C. Hubmann, "A combined model-and learning-based framework for interaction-aware maneuver prediction", *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1538–1550, 2016.

[17] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, "Analysis of Recurrent Neural Networks for Probabilistic Modeling of Driver Behavior", *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2016.

[18] A. Graves, "Generating Sequences With Recurrent Neural Networks", 2013. arXiv: 1308.0850.

[19] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)", 2015. arXiv: 1511.07289.

[20] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes", 2013. arXiv: 1312.6114.

[21] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models", *Proc. of the 31st International Conference on Machine Learning*, vol. 32, pp. 1278–1286, 2014.

[22] Federal Highway Administration (FHWA), "US highway 80 dataset", FHWA-HRT-06-137, Tech. Rep., 2005.

[23] C. Thiemann, M. Treiber, and A. Kesting, "Estimating Acceleration and Lane-Changing Dynamics Based on NGSIM Trajectory Data", *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2088, pp. 90–101, 2008.

[24] M. Abadi, A. Agarwal, P. Barham, *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems", 2015. arXiv: 1603.04467.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization", 2014. arXiv: 1412.6980.

[26] T. A. Wheeler, P. Robbel, and M. J. Kochenderfer, "Analysis of microscopic behavior models for probabilistic modeling of driver behavior", *Intelligent Transportation Systems Conference*, pp. 1604–1609, 2016.

[27] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations", *Physical Review E*, vol. 62, no. 2, 2000.