Data-Driven Online Estimation of Driver Model Parameters for Vehicle Trajectory Prediction

Johannes Fischer[1] and Christoph Stiller[1]

***Abstract*— Fast and accurate trajectory prediction is crucial for the development and validation of automated driving systems. Using driver models for prediction is a promising approach to achieve this since they can produce realistic driving behavior at low computational cost. To produce high-quality predictions, the driver model parameters need to be adapted to the current traffic situation and observed driving behavior online. Our work combines data-driven methods with driver models to obtain realistic short-term trajectory predictions. We propose to train machine learning models to predict the driver model parameters that best capture the observed behavior of other vehicles. We use attention-based architectures to process sequential input data and predict the driver model parameters as a weighted sum of prototypes, thus ensuring that the predicted driving model parameters are realistic. Compared to particle filter-based state-of-the-art methods, our approach profits from the rich representational capabilities of learned models and the high online runtime efficiency of driver models. We show that our approach outperforms state-of-the-art methods for online driver model parameter estimation on a real-world traffic dataset.**

I. Introduction

The introduction of automated vehicles is expected to not only improve road safety but also to increase transportation availability and reduce the carbon footprint of the transportation sector [1]. To successfully implement automated vehicles driving on public roads, it is crucial to ensure that they behave in a safe and predictable manner. To this end, motion planning algorithms for automated vehicles need to take into account how other traffic participants react to the planned ego motion. Furthermore, the driving behavior needs to be validated in recorded real-world traffic situations.

This makes it clear that the development of automated driving functions depends on high-quality prediction models for other traffic participants: Not only are they necessary for closed-loop motion planning, they are also used to simulate other traffic participants in the validation process. This is because when simulating a real-world traffic scene, the ego vehicle's behavior will typically deviate from the recorded vehicle trajectory. Hence, the behavior of other traffic participants needs to be adapted accordingly to the new ego motion by using a realistic prediction model.

While data-driving models have been proposed to predict other traffic participants, they are typically designed to make one prediction for the surrounding traffic per planning step. However, this limits their ability to make predictions for thousands of ego plans within one planning step, which is necessary to plan with algorithms like Monte Carlo Tree Search, as the model would have to be evaluated many times [2]. Furthermore, this also slows down validation simulations, since the prediction model needs to be evaluated again at each step to simulate the other traffic participants.

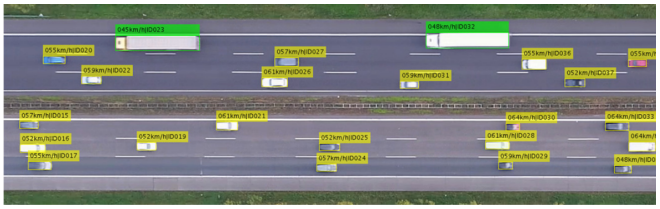


Fig. 1: Our evaluation is based on the highD dataset and predicts car-following model parameters in highway driving scenarios [7].

One solution to this problem is to use driver models for predicting the behavior of other traffic participants. Driver models can be evaluated at very low cost while still producing realistic driving behavior. The parameters defining the resulting behavior can be estimated online to adapt to the current traffic situation [3]–[6]. This allows to make many predictions for different ego behaviors at no additional cost by reusing the estimated driver model parameters. In contrast, trajectory prediction networks would need to be evaluated multiple times to make predictions for multiple ego behaviors.

The state-of-the-art methods for online estimation of driver model parameters are based on particle filters [4], [5] or online optimization [6]. Since these approaches still require significant online runtime, we investigate a data-driven approach to predict driver model parameters. This retains the benefit that the driver model parameters only have to be estimated once and then can be used for many predictions. In addition, it can also learn to focus on the important parts of the history, e.g., if a vehicle behaved aggressively some time steps ago. In contrast, such information might get lost over time with a Bayesian filtering approach or in a fixed-horizon optimization. With the increasing availability of traffic datasets, data-driven approaches are not limited by the availability of training data anymore.

The approach we propose in this work uses deep neural networks to predict driver model parameters. We train our models to either output the driver model parameters directly, or indirectly as a weighted sum of parameter prototypes.

[1]Johannes Fischer and Christoph Stiller are with the Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany. Corresponding author: johannes.fischer@kit.edu

This enables us to retain the benefits of data-driven and model-based approaches: Our models have the generalization capabilities of black-box models, are able to produce non-linear parameter predictions, and can learn which part of the history is important for the prediction. At the same time, by predicting a weighted sum of parameter prototypes, we can ensure that the resulting parameters are reasonable and can be used in a driver model. This results in a realistic trajectory prediction, which is not guaranteed for general black-box trajectory prediction models. The contributions of our work are as follows:

- We present a novel data-driven approach to predict driver model parameters for short-term trajectory prediction.
- We evaluate our approach on real traffic data from the highD dataset and compare different neural network architectures.
- We show that our approach outperforms state-of-the-art methods with respect to prediction accuracy and runtime efficiency.

II. Related Work

Prior work has addressed the problem of fitting driver model parameters to real-world data for both, simulation and prediction purposes. Many works have used offline methods to fit driver model parameters to traffic data [5], [8], [9]. Since only one global parameter set is fitted, the resulting behavior is likely to represent average driving behavior. In [9], this is approach coupled with imitation learning to recover the driving behavior from the dataset, and the driver model is just used to regularize the imitation learning task. Other authors suggest to use multiple parameter sets, since humans exhibit different driving styles [10]. In contrast to these works, our approach does not fit the driver model parameters to the data globally, but estimates them online to adapt to the current traffic situation and the individual vehicle's driving style. However, we adopt the idea of different driving styles when we predict the driver model parameters as a weighed sum of fixed parameter prototypes.

Other works have estimated driver model parameters online before. Monteil *et al.* use an extended Kalman filter for online estimation of driver model parameters [3]. Particle filters are used for the same task within POMDP solvers for decision-making under uncertainty [2] and for traffic scene prediction [4], [5], [11], [12]. While these works share our goal of estimating driver model parameters online for prediction and decision-making, they rely on Bayesian filtering techniques that can be computationally expensive. In contrast, we propose a data-driven approach that can be trained offline and is computationally efficient at runtime.

An optimization-based approach for online parameter estimation is proposed by Kreutz *et al.* Instead of Bayesian filtering, they fit the driver model parameters to the recent observations using an optimization algorithm [6]. Like our approach, they fit the parameters as a weighted sum of parameter prototypes by optimizing the prototype weights. They show that this approach is computationally more efficient than particle filtering. However, it still requires significant online runtime.

III. Fundamentals

Driver models are widely used to capture realistic vehicle behavior in traffic simulations. Given the current traffic state, they predict the future traffic evolution. They are typically parameterized with only few interpretable parameters, which can be used to tune the behavior of the driver model. While macroscopic driver models usually model the traffic flow, microscopic driver models focus on the behavior of individual vehicles. Such microscopic driver models can be used to model car-following behavior, lane-changing behavior, merging behavior and other aspects of individual vehicle behavior.

A. The Intelligent Driver Model

A well-known car-following model is the Intelligent Driver Model (IDM) [13]. It is a microscopic driver model that describes the acceleration of a vehicle based on its velocity v, the distance headway d and the velocity of the leading vehicle v_{lead}. Mathematically, the acceleration a_{IDM} of the target vehicle is given by

$$a_{\text{IDM}} = a_{\max} \left(1 - \left(\frac{v}{v_0} \right)^{\delta} - \left(\frac{d^*(v, v_{\text{lead}})}{d} \right)^2 \right) \quad (1)$$

where the dynamic desired distance is given by

$$d^*(v, v_{\text{lead}}) = d_0 + \max \left(0, vT + \frac{v(v - v_{\text{lead}})}{2\sqrt{a_{\max} b}} \right) \quad (2)$$

The model's parameters $(a_{\max}, v_0, d_0, T, b, \delta)$ consist of the maximum acceleration $a_{\max}$, the desired velocity v_0, the minimum distance d_0, the desired time headway T, the comfortable deceleration b, and the free-driving exponent δ.

The last term in Equation (1) is the interaction term that relates the desired distance to the actual distance headway. With shrinking distance headway, the acceleration decreases and vice-versa. At the same time, the desired distance contains an "intelligent braking strategy" that adapts the desired distance such that the resulting deceleration tends to not exceed the comfortable deceleration b [13]. The other terms in Equation (1) aim to drive with the desired velocity v_0. Overall, the IDM balances driving with the desired velocity and keeping a safe distance to the leading vehicle. Its parameters can be tuned to reproduce realistic driving behavior as well as different driving styles [10].

B. The Online Intelligent Driver Model

The Online Intelligent Driver Model (OIDM) is not a driver model, but a prediction model based on the IDM [6]. The OIDM optimizes the IDM parameters to describe the observed vehicle behavior. To this end, they consider observations of the last L time steps $(s_{t-L}, \ldots, s_{t-1})$. For a given set of IDM parameters, a trajectory of the vehicle can be simulated using these IDM parameters starting from the state s_{t-L}. The IDM parameters are then optimized to minimize the difference between this simulated trajectory and

the observed trajectory. The authors measure the trajectory difference either by the absolute velocity difference objective

$$J_v = \sum_{h=t-L}^{t-1} |v_h - \hat{v}_h| \tag{3}$$

or the absolute acceleration difference

$$J_a = \sum_{h=t-L}^{t-1} |a_h - \hat{a}_h|. \tag{4}$$

The parameters are not optimized directly, but instead they optimize weights $w \in \mathbb{R}^N$ with $w_i \geq 0$ and $\sum_{i=1}^{N} w_i = 1$. These weights are used to obtain the parameters p as a linear combination $p = \sum_{i=1}^{N} \lambda_i p_i$ of parameter prototypes p_i. The authors use the Hooke-Jeeves direct search algorithm for the optimization [14].

IV. Technical approach

The standing assumption of our work is that at each point in time there exists a set of driver model parameters that describe the vehicle's behavior reasonably well. While these parameters can change over time, we assume that they change slowly enough, such that the driver model can be used to predict the vehicle's behavior for a short time horizon.

Hence, our work focuses on estimating the driver model parameters online to use them for short-term predictions. To minimize the time necessary for parameter inference, we propose to use data-driven methods that are trained offline to predict the driver model parameters. In our work, we focus on driver model parameter prediction using deep neural networks. We experiment with different input and output representations that can be combined in various ways and are described in the following sections.

A. Sequential Input Data Processing

The driver model parameters that best describe the current vehicle behavior depend not only on the current traffic state s_t but also on the vehicle's past behavior. If the same vehicle has been driving aggressively in the past, it is likely to continue driving aggressively in the future. For this reason, the history of the vehicle's behavior is crucial for predicting the driver model parameters. Put differently, we have to infer the driver model parameters based on sequential input data. To this end, we propose several neural network architectures that differ in their respective input representations, depicted in Fig. 2.

1) Fixed-length History Feedforward Neural Network: The simplest approach to represent the vehicle's history is to use a fixed-size history window of length L. At time step t, the input to the neural network is composed of the last L traffic observations $(s_{t-L+1}, \ldots, s_{t-1}, s_t)$. This approach is illustrated in Fig. 2a.

2) Recurrent Neural Network: As an alternative, recurrent network architectures can be used to process sequential input data. Recurrent neural networks (RNNs) maintain a hidden state that is updated at each step and influences the network's output. The network receives only the current traffic state s_t as input at each time step t, but it can learn to update the hidden state such that it retains the important information from the past. A recurrent architecture is exemplified in Fig. 2b.

3) Attention-based Neural Network: Attention mechanisms have been proven to be effective in processing sequential data [15]. They allow the network to focus on the most relevant parts of the input sequence. The whole traffic state input sequence $(s_1, \ldots, s_{t-1}, s_t)$ is passed to the network, which then learns to assign different weights to each input element. We use the scaled dot-product attention mechanism that learns to map the input sequence to keys, values, and queries from which the attention weights are computed [16]. The concept is depicted in Fig. 2c. In our work, we use the attention mechanism in two flavors: One that only uses the attention mechanism to process the input sequence, followed by a regular feed-forward network, and one that uses a transformer encoder architecture [16].

B. Driver Model Parameter Prediction

We use neural networks to predict the driver model parameters that best describe the vehicle behavior at the current time step. Hence, the output of the prediction model is a vector of driver model parameters for the given traffic state. We experiment with two different approaches to map the network output to driver model parameters: Direct parameter prediction and prototype-based parameter prediction.

1) Direct Parameter Prediction: In the direct parameter prediction approach, the neural network directly outputs the driver model parameters. To this end, the network outputs y are transformed to the interval $[0, 1]$ using a sigmoid activation function. The sigmoid outputs $\sigma(y)$ are then scaled to the respective parameter range where the minimum $p_{\min}$ and maximum $p_{\max}$ for each parameter are determined beforehand based on expert knowledge. This ensures that the network produces valid parameter values

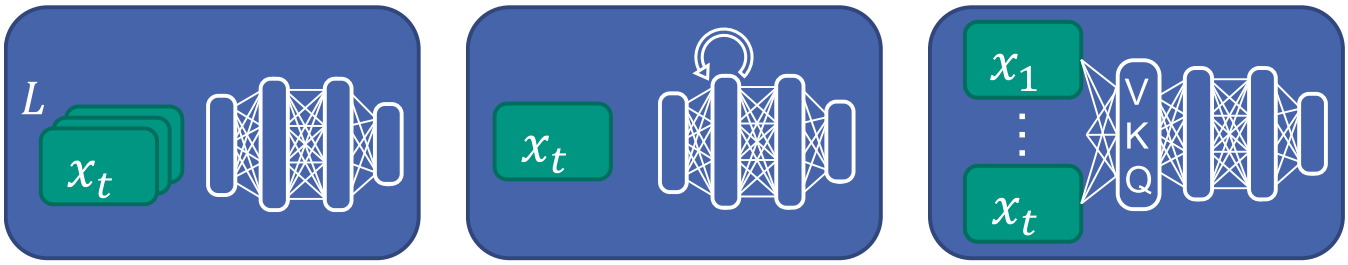
$$p = p_{\min} + \sigma(y) \cdot (p_{\max} - p_{\min}) \in [p_{\min}, p_{\max}] \tag{5}$$

but has great flexibility in predicting the parameters.

2) Prototype-Based Parameter Prediction: In the prototype-based parameter prediction approach, we use a set of prototype parameters $\{p_1, \ldots, p_N\}$ that represent different driver model parameter sets compiled by an expert. The neural network outputs $y \in \mathbb{R}^N$ are mapped to a set of weights $\lambda \in \mathbb{R}^N$ using a softmax activation function, resulting in positive weights that sum up to one. The driver model parameter prediction p is then computed as a weighted sum of the prototype parameters

$$p = \sum_{i=1}^{N} \lambda_i p_i \tag{6}$$

This approach limits the possible parameters to a convex combination of the prototypes and as such is more restrictive than the direct parameter prediction approach. At the same time, this restriction has the advantage that only reasonable parameter sets can be predicted by the model.



(a) Neural network architecture with fixed-size history of length L.

(b) Recurrent neural network architecture with one recurrent layer.

(c) Attention-based neural network architecture using scaled dot-product attention.

Fig. 2: Different neural network architectures for dealing with sequential input data.

C. Training Objective

We want to train our models to predict the driver model parameters that best describe the current vehicle behavior. Given a parameter prediction of our network we use the corresponding driver model to predict the next acceleration value. Since we want to use our model for prediction, we train it to minimize the mean squared error between the next acceleration computed with the predicted driver model parameters and the true acceleration.

V. Experiment Setup

We evaluate our data-driven approach to driver model parameter prediction on the task of predicting car-following behavior. To this end, we use our model to predict the driver model parameters and evaluate the quality of the short-term trajectory predictions in car-following scenarios.

A. Driver Model

We use the Intelligent Driver Model (IDM) [13] in our experiments to model the vehicle's behavior and use our approach to estimate the IDM parameters online. As it is commonly done in the literature, we fix the free-driving exponent to $\delta = 4$, and only estimate the remaining IDM parameters $(a_{\max}, v_0, d_0, T, b)$.

B. Real Vehicle Traffic Dataset

For the evaluation of our approach, we make use of the highD dataset [7]. The highD dataset contains vehicle trajectories from German highways recorded with stationary drones. The dataset is recorded with $25\,\mathrm{Hz}$, but we downsample it by a factor of 3, resulting in a time resolution of $\Delta t = 0.12\,\mathrm{s}$. Since we are interested in car-following scenarios, we filter the dataset to only include trajectories where the target vehicles have a leading vehicle and do not change lanes. Lane changes of the target or leading vehicle would likely result in additional accelerations that can not be explained by a car-following model. We use 80% of the dataset for training and 20% for validation.

C. Observation Modeling

The observation our models receive at each time step is the IDM input state $s = (d, v, v_{\text{lead}})$, consisting of the distance headway d and velocity v of the target vehicle, and the velocity of the leading vehicle v_{lead}. Depending on the particular strategy to process sequential inputs, these observations are stacked to form a fixed-size history input, processed sequentially using a recurrent neural network, or processed as a whole sequence using an attention mechanism.

We note that it is well possible that augmenting the state with more traffic information could lead to better parameters predictions. For example, the state of the vehicle ahead of the leading vehicle or the adjacent lane traffic could provide valuable information for future behavior. However, we restrict our experiments to only using the IDM input state, as this is the same information provided to the baseline methods. Thus, we can compare both approaches fairly.

D. Network Architectures and Training

We evaluate different neural network architectures combining the different input and output representations described in Sections IV-A and IV-B. In the following, we refer to the fixed-size history input representation as DNN and the recurrent input representation as RNN. The attention-based input architecture that only uses one attention layer is referred to as ATTN whereas the full transformer encoder architecture is referred to as TRF. The minimum and maximum values for the parameters as described in Section IV-B.1 are provided in Table I.

For the variants that use the prototype-based output representation, we use $N = 3$ prototype parameter sets, which are provided in Table I. Note that the desired velocity v_0 is relative to the current velocity v of the vehicle. This is necessary since the traffic data contains a wide range of velocities, such that a fixed desired velocity for each prototype would not be appropriate in all scenarios. The network variants that use the prototype-based output representation are referred to as P-DNN, P-RNN, P-ATTN, and P-TRF, respectively.

The recurrent architectures use a Gated Recurrent Unit [17] and the attention-based models use multi-head attention with sinusoidal positional encodings [16]. The neural networks are trained using the Adam optimizer [18] and the hyperparameters used for training are provided in Table II. These parameters were identified based on a preliminary hyperparameter search. All networks are trained for 200 epochs.

	v_0	$a_{\max}$	T	d_0	b
$p_{\min}$	$0\,\mathrm{m/s}$	$0\,\mathrm{m/s^2}$	$0\,\mathrm{s}$	$0\,\mathrm{m}$	$0\,\mathrm{m/s^2}$
$p_{\max}$	$100\,\mathrm{m/s}$	$10\,\mathrm{m/s^2}$	$10\,\mathrm{s}$	$50\,\mathrm{m}$	$10\,\mathrm{m/s^2}$
Defensive	$v - 0.4\,\mathrm{m/s}$	$1.0\,\mathrm{m/s^2}$	$1.8\,\mathrm{s}$	$4.0\,\mathrm{m}$	$1.0\,\mathrm{m/s^2}$
Normal	$v + 3.6\,\mathrm{m/s}$	$1.6\,\mathrm{m/s^2}$	$1.4\,\mathrm{s}$	$2.0\,\mathrm{m}$	$2.0\,\mathrm{m/s^2}$
Aggressive	$v + 7.6\,\mathrm{m/s}$	$2.2\,\mathrm{m/s^2}$	$0.7\,\mathrm{s}$	$1.0\,\mathrm{m}$	$3.5\,\mathrm{m/s^2}$

TABLE I: Parameter range (top) and prototype parameter sets (bottom) for the IDM, based on [6], [10].

Parameter	DNN	P-DNN	RNN	P-RNN
Hidden layers	2	2	2	4
Hidden neurons	32	128	128	128
Learning rate	1e-5	1e-3	1e-3	1e-3
Batch size	256	128	16	16
Parameter	**ATTN**	**P-ATTN**	**TRF**	**P-TRF**
Key dimension	8	16	16	8
Value dimension	32	32	32	32
Number of heads	4	4	4	8
Hidden layers	4	5	5	5
Hidden neurons	128	64	32	32
Learning rate	1e-4	1e-3	1e-3	1e-3
Batch size	16	16	16	16

TABLE II: Hyperparameters of the different neural network architectures.

E. Baselines

Like previous approaches for online driver model parameter estimation, we compare our approach to well-established baseline models for lane-following prediction: A constant velocity (CV) model, a constant acceleration (CA) model, and a combination of both (CACV) [6]. CACV follows a CA model for the first $1.5\,\mathrm{s}$, then the acceleration decreases linearly until reaching zero after $2.5\,\mathrm{s}$ and continuing with a CV model thereafter. Albeit being simple, these models are widely used in the literature due to their runtime efficiency and good prediction performance.

We also compare our approach to the online intelligent driver model (OIDM) [6], described in Section III-B. As in their original work, we consider two objectives for the optimization and refer to the two variants as OIDM(J_v) and OIDM(J_a). For a fair comparison, we use the same history length $L = 5$ in our DNN models as in the OIDM variants. We use the IPOPT algorithm to solve the optimization problem [19].

Prior work has compared Bayes filter-based online parameter estimation for the IDM and has shown only marginal gains over the CV and CA prediction models [4]–[6]. Furthermore, the OIDM approach has been shown to outperform the Bayes filter-based approach with respect to both, performance and runtime efficiency. For this reason, we do not include those methods in our evaluation.

VI. Evaluation and Results

To evaluate our approach, we predict the driver model parameters for every time step of every vehicle trajectory in the dataset using our models as well as the baseline models. Using the predicted parameters for each model, we simulate the vehicle trajectories for a medium-length fixed time horizon of $6\,\mathrm{s}$. To do so, we predict the leading vehicle's trajectory using a CACV model and roll out the driver model with the predicted parameters given this predicted trajectory for the leading vehicle. We then compare the predicted trajectories to the ground truth trajectories of the considered vehicle in terms of the absolute error in the position, velocity, and acceleration profiles.

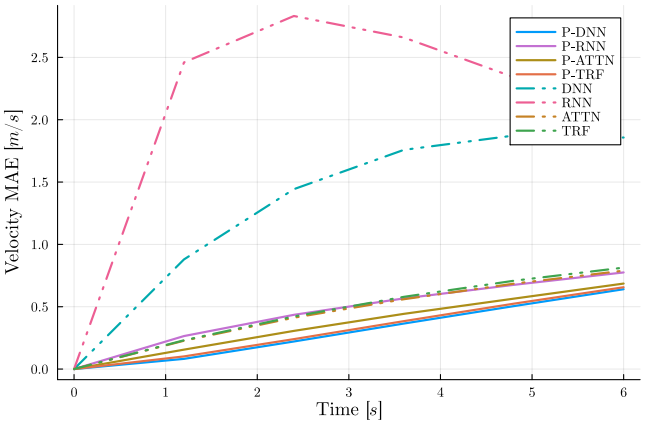


Fig. 3: Mean absolute velocity error of the prediction for different model architectures.

A. Comparison of Network Architectures

We first compare the different network architectures we use with our approach and the direct and prototype-based output representations. To this end, we consider the mean absolute error (MAE) of the predicted velocity, computed for different prediction time horizons, where the mean is taken over all evaluation samples. The results are shown in Fig. 3 for all our models.

We can observe that the DNN and RNN models that directly predict the parameters perform a lot worse than the remaining models. We can also observe that the prototype-based models perform better than the direct parameter prediction models. This is in accordance with the findings of Kreutz *et al.* who also concluded that prototype-based parameter estimation performs better than direct parameter estimation in their optimization-based OIDM prediction model [6]. The performance of the two other direct parameter prediction models, ATTN and TRF, is on par with the prototype-based P-RNN model and not much worse than the other prototype-based models. This can be the result of the attention mechanism, since considering the whole input sequence could have an averaging effect, resulting in less extreme parameter predictions.

Out of the prototype-based models, P-DNN and P-TRF perform the best. For this reason, we will only consider these two models in the following evaluations. A possible explanation why the more complex P-TRF model does not outperform the P-DNN model could be that the IDM is a

simple model that can be well approximated by a simple network and that the P-DNN only receives the most recent observations as input whereas the P-TRF model has to learn which observations are relevant for the prediction. However, with more complex driver models and traffic situations that exhibit more interactive behavior, the transformer-based model might become advantageous.

B. Benchmarking Prediction Accuracy against Baselines

In the following, we consider our two best-performing models, P-DNN and P-TRF and compare their prediction accurary against the baseline models. We also compare the performance with our transformer-based direct parameter prediction model TRF to illustrate the effect of prototype-based parameter prediction. Fig. 4 depicts the MAE of the predicted position, velocity, and acceleration for different prediction time horizons.

While the CV model performs worst, we observe that our trained models perform similar to the CA and CACV models for very short prediction horizons of up to 2.5 s. However, for larger time horzions, our models outperform these simple models.

Compared to the OIDM baseline models, our prototype-based parameter prediction models exhibit superior prediction performance for all time horizons. As Fig. 4c shows, this can be attributed to the large error of the OIDM models in the prediction of the immediate next acceleration. For larger time horizons, all IDM-based prediction models converge to the same acceleration MAE. This can be explained by the fact that the IDM converges to stationary driving with acceleration going to zero, independent of the predicted parameters. For this reason, the velocity prediction performance gap between OIDM and our P-DNN and P-TRF approaches is larger for short time horizons, as can be observed in Fig. 4b. Our direct parameter prediction model TRF performs similar to OIDM.

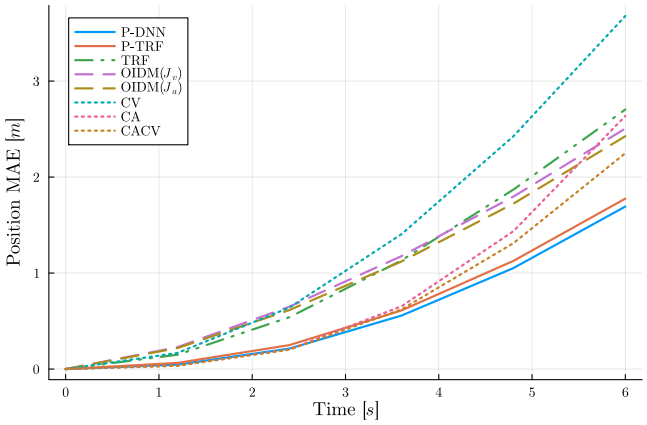
C. Prediction Error Distribution

To evaluate the distribution of the prediction errors, we use box plots of the absolute position and velocity errors at certain prediction horizons. Fig. 5 shows the box plot of the absolute position error for the maximum prediction horizon of 6 s. We can see that our prototype-based parameter prediction models exhibit a lower spread of the prediction errors, while TRF performs again similar to the OIDM models. Moreover, the OIDM models have a higher spread than the CA and CACV models.

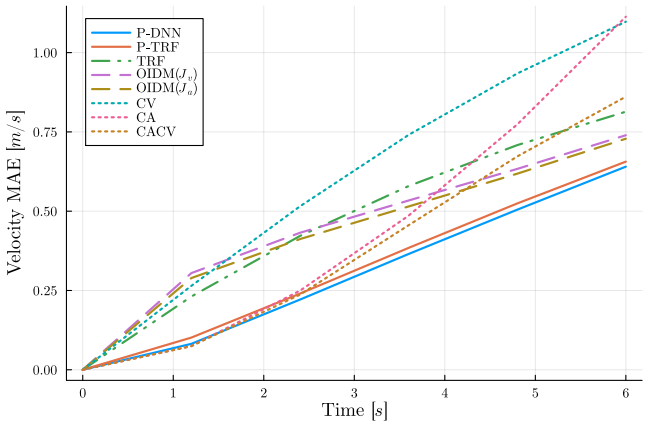
We evaluate the distribution of the absolute velocity errors for a prediction horizon of 1.2 s, since the differences in the velocity MAE are most pronounced at this time horizon. As Fig. 6 illustrates, the TRF, OIDM and CV models have very high spread in the prediction errors, while the CA and CACV models have the lowest spread, closely followed by our prototype-based models P-DNN and P-TRF.

D. Online Runtime Evaluation

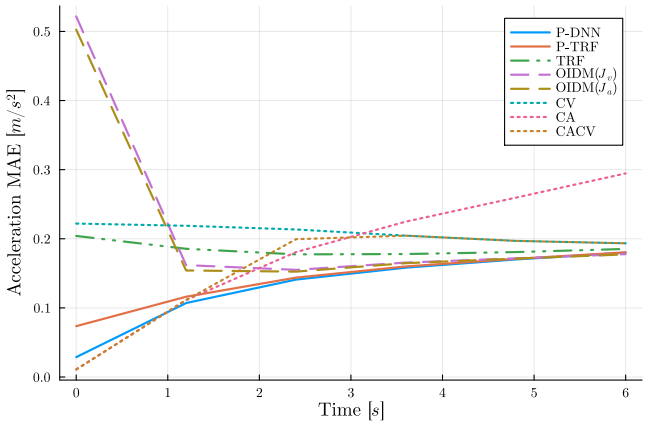
At last, we compare the online prediction time of our trained models with the OIDM models. While prediction



(a) Mean absolute position error of the prediction.



(b) Mean absolute velocity error of the prediction.



(c) Mean absolute acceleration error of the prediction.

Fig. 4: MAE of the prediction at different time horizons for our approach and the baseline models.

with our models requires only a forward pass through the neural network, the OIDM models require solving an optimization problem online. In Fig. 7, we can see that the OIDM models require significantly more time for the prediction than our models. We can also observe that the larger transformer-based architecture requires more time than the history-based P-DNN model.

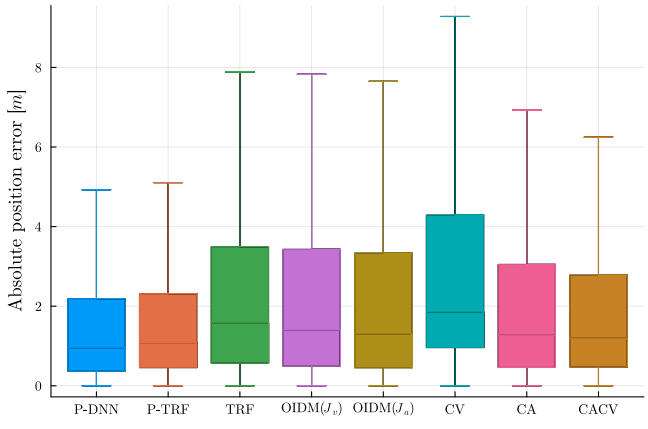


Fig. 5: Boxplot of the absolute position error of the prediction after 6 s.

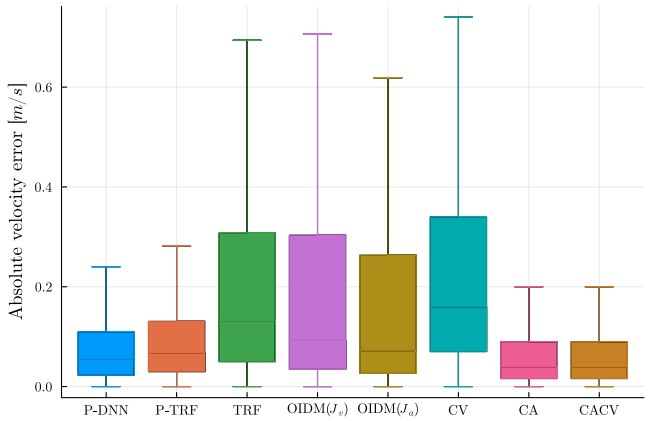


Fig. 6: Boxplot of the absolute velocity error of the prediction after 1.2 s.

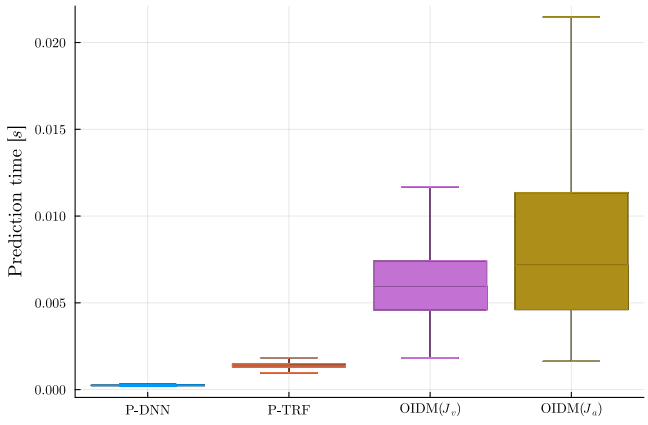


Fig. 7: Boxplot of the online prediction time.

These results have to be taken with care, since the runtime depends not only on the used hardware, but also on the hyperparameters of the optimization algorithm. Thus, they are only indicative and can only be used for a rough comparison. However, they resemble the experience we made with the algorithms during our experiments. In addition, parameter prediction with neural networks can easily be integrated as a separate head into other learned components that might be present in the planning pipeline at barely any additional cost. In contrast, the OIDM models require solving an optimization problem for every parameter prediction.

VII. Conclusions and Future Work

In this work, we have presented a novel approach to predict the parameters of driver models using deep neural networks. Our models predict weights that are used to compute the driver model parameters as a convex combination of fixed prototype parameters. The resulting driver model parameters can be used to predict or simulate the future behavior of the target vehicles.

We have compared our approach to optimization-based state-of-the-art methods and to widely used prediction models. As our extensive evaluation revealed, our method outperforms the state-of-the-art methods in terms of prediction accuracy and computational efficiency.

The computational efficiency of driver models makes it attractive to use them as a rollout policy in planning strategies like Monte Carlo Tree Search, a popular decision-making algorithm. Our approach allows to estimate the driver model parameters online that best describe the behavior of the surrounding traffic participants. Thus, we can predict accurate driver model parameters once per planning step and use them to generate many realistic simulations by propagating the driver model for the other traffic participants.

References

[1] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, Jul. 2015, ISSN: 0965-8564.

[2] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The Value of Inferring the Internal State of Traffic Participants for Autonomous Freeway Driving," in *American Control Conference (ACC)*, Seattle, 2017.

[3] J. Monteil, N. O'Hara, V. Cahill, and M. Bouroche, "Real-Time Estimation of Drivers' Behaviour," in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 2046–2052.

[4] S. Hoermann, D. Stumper, and K. Dietmayer, "Probabilistic long-term prediction for autonomous vehicles," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 237–243.

[5] R. P. Bhattacharyya, R. Senanayake, K. Brown, and M. J. Kochenderfer, "Online Parameter Estimation for Human Driver Behavior Prediction," in *2020 American Control Conference (ACC)*, Jul. 2020, pp. 301–306.

[6] K. Kreutz and J. Eggert, "Fast online parameter estimation of the Intelligent Driver Model for trajectory prediction," in *2022 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2022, pp. 758–765.

[7] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI: IEEE, Nov. 2018, pp. 2118–2125, ISBN: 978-1-72810-321-1 978-1-72810-323-5.

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

[9] Z. Mo, X. Di, and R. Shi, "A Physics-Informed Deep Learning Paradigm for Car-Following Models," *Transportation Research Part C: Emerging Technologies*, vol. 130, p. 103 240, Sep. 2021, ISSN: 0968090X. arXiv: 2012.13376 [cs, eess].

[10] A. Kesting, M. Treiber, and D. Helbing, "Agents for traffic simulation," *Multi-agent systems: Simulation and applications*, vol. 5, 2009.

[11] J. Buyer, D. Waldenmayer, N. Sußmann, R. Zöllner, and J. M. Zöllner, "Interaction-Aware Approach for Online Parameter Estimation of a Multi-lane Intelligent Driver Model," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Oct. 2019, pp. 3967–3973.

[12] J. Buyer, D. Waldenmayer, R. Zöllner, and J. M. Zöllner, "Data-Driven Merging of Car-Following Models for Interaction-Aware Vehicle Speed Prediction," in *2021 IEEE 24th International Conference on Information Fusion (FUSION)*, Nov. 2021, pp. 1–8.

[13] M. Treiber, A. Hennecke, and D. Helbing, "Congested Traffic States in Empirical Observations and Microscopic Simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, Aug. 2000, ISSN: 1063-651X, 1095-3787.

[14] R. Hooke and T. A. Jeeves, ""Direct Search" Solution of Numerical and Statistical Problems," *Journal of the ACM*, vol. 8, no. 2, pp. 212–229, Apr. 1961, ISSN: 0004-5411, 1557-735X.

[15] D. Bahdanau, K. Cho, and Y. Bengio, *Neural Machine Translation by Jointly Learning to Align and Translate*, May 2016. arXiv: 1409.0473 [cs, stat].

[16] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[17] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[19] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, Mar. 2006, ISSN: 1436-4646.