Performance and Efficiency Analysis of a Linear Learning-Based Prediction Model used for Unintended Lane-Departure Detection

John Dahl, Gabriel Rodrigues de Campos and Jonas Fredriksson

***Abstract*—Advanced driver assistance systems have been an active research topic for decades, for which many approaches have been developed not only to reduce the number of traffic accidents but also to increase the driver's comfort. Among the many different solutions proposed, learning-based prediction approaches have gained considerable attention in recent years. Within this scope, this work focuses on the implementation aspects of a linear learning-based regression model for detecting unintended lane-departures, where the goal is to achieve a prediction model with good predictive performance while keeping the computational complexity as low as possible. Aspects under consideration include input signal selection and down-sampling. The linear prediction model is analyzed using a real world data set, and benchmarked against a kinematic constant velocity model and a non-linear regression model. The results show that the linear regression model has a significantly higher prediction performance when compared to a kinematic model. It is also shown that the predictive performance remains comparable to the more complex nonlinear regression model, even though the computational complexity of the linear model is significantly lower.**

***Index Terms*—Threat-assessment algorithms, decision-making methods, intelligent vehicles, active safety systems.**

I. Introduction

Every year, over 1.2 million fatalities are caused by traffic accidents around the world. While one could think that the majority of these causalities is due to faulty roads or malfunctioning vehicles, an analysis on traffic accidents in the US indicates that the human error is associated with up to 99% of the traffic accidents [1]. In this context, the human error accounts for errors made by the drivers such as inattentiveness, reckless driving and drowsiness.

To cope with the human error problems, different Advanced Driver Assistance System (ADAS) have been proposed in recent years. However, designing a well-functioning ADAS is a non-trivial task since it has to deal with various situations, road conditions and driving styles [2]. Moreover, a vehicle in motion obeys to the laws of dynamics, implying that a warning or an automated assistance maneuver must be activated before the traffic situation has reached the point of no return. Hence, predicting the future driving situation, often referred to as Threat Assessment (TA), is vital in the context of Collision Avoidance Systems (CAS). A well functioning TA method must therefore be robust to variations in the scenario, and particularly with respect to different drivers' behavior. To deal with such problems, many different TA methods have been proposed and evaluated over the years, e.g., methods based on kinematics, optimization or probabilistic methods. A thorough literature review on threat assessment techniques for CAS is found in [3].

Fuelled by an increasing general interest in artificial intelligence techniques, a lot of attention has been given to machine-learning approaches that leverage large quantities of real-world data. Some works are based on end-to-end learning, see e.g. [4], where frames from a video camera stream and in-vehicle measurements are used to train a Long Short-Term Memory (LSTM) network, in combination with a convolutional network, to predict the future driver actions. A similar approach, using only camera frames as an input, is found in [5], where a combination of a convolutional Long Short-Term Memory (conv-LSTM) network and a spatio-temporal network is used to predict the future steering wheel angle. Several other works focus on the threat assessment problem by using time series data. For example, in [6] the motion of surrounding vehicles are predicted, with the corresponding uncertainty, using a deep ensemble of Recurrent Neural Networks (RNN), where the paths are expressed as third order polynomials. A similar approach is used in [7], where the motion of surrounding vehicles are predicted using a hybrid architecture based on a Feed-Forward Neural Network (FFNN) and an LSTM network. In [8], a Deep Kinematic Model (DKM), based on an LSTM network architecture, is used to predict kinematically feasible motion profiles of surrounding vehicles. The feasibility is guaranteed by replacing the output-layer with a kinematic output-layer, which restricts the future states of the path to obey a kinematic vehicle model. A similar problem is addressed in [9], where the ego-vehicle's path is predicted using a CNN-LSTM based network architecture. The benchmark results indicate a superior predictive performance, but the prediction model is conditioned on the presence of a fixed number of surrounding vehicles, which is a fundamental limitation. An Ensemble of Bi-directional Recurrent Neural Networks (EBiRNN) is used in [10] to predict intended lane changes of the ego-vehicle. Despite the robustness of an ensemble based network architecture, the EBiRNN is just slightly better performing than a standard, fully connected Feed-Forward Neural Network (FFNN). On a complementary problem, a Multilayer Perceptron (MLP) is used in [11] to predict unintended lane departures, where it is shown that an MLP has superior performance when compared to a kinematic prediction model, at the cost of increased computational demands.

While accurate predictions are a straightforward require-

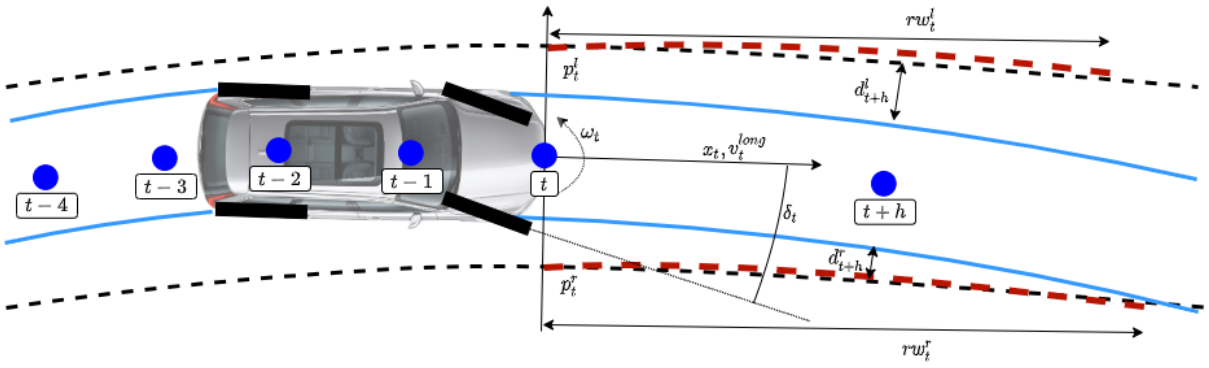

Fig. 1. System overview and available signals in the data-set. The lane markers are represented by a third order polynomial, $p^{\diamond}$, where $\diamond$ is a wild card for left and right. The polynomials are valid for a range of view, $rw^{\diamond}$, determined by the vision system. The vehicles dynamics are represented by the yawrate ω, front wheel angle δ, and longitudinal velocity v. The future path of the vehicle, h time steps ahead, is represented as the distance to the lane marker $d_{t+h}^{\diamond}$.

ment for any safety related system, computational efficiency is also of particular importance in the automotive industry. The computational efficiency of an algorithm, determined by its *time complexity*, is proportional to the number of operations, and thereby also to the runtime needed for the computation of the output of the prediction model, see [12] and [13] for a detailed introduction to computational efficiency. Despite the general belief of extended computation capabilities in the most recent vehicles [14], the computational power on board is normally restricted. It is therefore of great importance that any threat assessment method is computationally efficient to be able to run in real-time. However, advanced network architectures, such as networks based on recurrent layers or ensembles of networks, tend to be computationally demanding [10]. In order to justify the increase in cost for higher computational power, it is important that a computationally demanding prediction model is significantly better in terms of predictive performance.

It is precisely this trade-off between predictive performance and computational complexity that this work is focusing on. We will build upon a previous work by the same group of researchers [15], that explores a Multiple Linear Regression (MLR) prediction model for the detection of unintended lane-departures. Despite the simplistic model architecture and the small number of model parameters, the linear regression model showed promising results regarding the predictive performance. In this new work, the MLR's predictive performance, used in the application of Lane Keeping Assist (LKA), is analysed with respect to the selection of input signals and down-sampling rate, for prediction horizons between 0.5 and 1.75 s. The performance is benchmarked against a kinematic Constant Velocity (CV) model and a Multiple Non-Linear Regression (MNLR) model, using a real world data set, which is derived from highway roads driving. We will show that the MLR is significantly better compared to the CV model and almost as good as the MNLR model in terms of predictive performance. However, the MLR model has significantly lower time complexity when compared to the MNLR model. Hence, the MLR seems to be a good trade-off between real-time and predictive performance.

The main contributions of this work are:

- A sensitivity analysis on how the selection of input signals affects the MLR model's predictive performance.
- A detailed benchmark against a kinematic model and a MNLR model using a real world data set.
- Insights on how the parameterization of the learning-based prediction models affects the computational complexity in runtime.
- A detailed study of the input signals' frequency characteristics and empirical distributions.

The paper is organized as follows. Sec. II covers an introduction to lane keeping assist systems. The problem statement and the prediction models used for threat assessment are presented in Sec. III, followed by an introduction to the data set in Sec. IV. Moreover, Sec. V elaborates on the implementation aspects, while Sec. VI presents the results and the analysis. Finally, the conclusions are given in Sec. VII.

II. Lane keeping assist

In general words, a LKA system aims to prevent lane departures, using a prediction model designed to foresee the vehicle's future path. Should the vehicle and it's occupants be at risk, an automatic steering maneuver is to be triggered, leading the vehicle towards the center of the ego lane. The accuracy of the future path predictions is crucial for the system's effectiveness: if prediction errors are made, either actual lane departures go undetected, which compromises the vehicle's and it's occupants safety, or unnecessary interventions are performed, ultimately disturbing or annoying the driver. All these aspects highlight the essential nature of TA within an LKA system. In the remaining of the paper, the TA part of the LKA system is referred to as the *prediction model*.

An illustration of the considered vehicle system is given in Fig. 1. Given in-vehicle measurements, the underlying idea of the TA method is to predict, point-wise at the time instance $t+h$, the relative distance between the vehicle and the left and right side lane markers d_{t+h}^l and d_{t+h}^r, respectively. An *activation* of an automatic steering maneuver is triggered when either the predictions $\hat{d}_{t+h}^l \leq \tau$ or $\hat{d}_{t+h}^r \leq \tau$ is true, i.e., whenever a lane departure is predicted. The parameter τ is a

design parameter related to how close to the lane marking an activation should be triggered, where $\tau = 0$ means that the vehicle reaches exactly the lane marking.

The in-vehicle measurements are the yaw rate denoted by ω_t, the front wheel angle denoted by δ_t and the longitudinal velocity denoted by v_t^{long}. The geometry of the lane markings relative to the ego-vehicle, as illustrated in Fig. 1, is captured by a forward looking vision system, that also includes an estimation software. Here, p_t^l and p_t^r denote the approximate lane marker polynomials on the left and right side, respectively. They can generically be written as:

$$p_t^\diamond(x) = a_{0,t}^\diamond + a_{1,t}^\diamond x + a_{2,t}^\diamond x^2 + a_{3,t}^\diamond x^3, \tag{1}$$

where x is the longitudinal distance, t the time instance, and the $\diamond$ symbol a wildcard indicating the side of the vehicle. The vision system is also estimating the range of view $rw_t^\diamond$, which is the longitudinal distance for which the polynomials are considered valid by the sensor.

III. Motion Prediction Models

This section focuses on prediction models that could be used for TA purposes. First, an introduction to multi-step predictions and sparse sampling of time series is provided. Second, the computational aspect of prediction models is discussed. Third, a linear regression-based prediction model, analysed later in this paper, is also described. Last, a non-linear regression model and a simpler kinematic model, to be used for benchmark purposes, are also presented.

A. Multi-step prediction and sparse sampling

Define a sampled time series as a sequence of states Y_t, ordered by time instant $t \in \mathbb{N}$, as:

$$Y_t = [y_{t,1}, y_{t,2}, \ldots, y_{t,K}]^\intercal, \tag{2}$$

where $y_{t,k},\ k \in [1, K]$, denotes the $k : th$ signal in the time series and $\intercal$ the matrix transpose operator. Notation-wise, the time series is called *univariate* if $K = 1$ and *multivariate* if $K > 1$, i.e., a vector valued time series. The time series is typically determined by observations from an unknown underlying process that exhibits random properties. Assume that the time series is derived from an auto-regressive process [16]. To be precise, let the state Y_{t+1} be derived as:

$$Y_{t+1} = g(Y_t, Y_{t-1}, ..., Y_{t-p-1}) + W_t, \tag{3}$$

where g is an arbitrary and unknown function and $p \in \mathbb{N}$ determines the number of samples used in the process. The process is driven by a white noise $W_t \in \mathbb{R}^K$, with $\mathbb{E}[W_t] = 0$, $\mathbb{E}[W_t W_t^\intercal] = \Sigma \in \mathbb{R}^{K\times K}$ and $\mathbb{E}[W_t W_s^\intercal] = 0$ for $t \neq s$. Hence, each state in the time series is a function of old samples, in a consecutive order and with a uniform sample rate determined by the sample time T_s.

A time series prediction model is used to estimate the future state $\hat{Y}_{t+h}$ of the series, where h indicates the number of future steps, based on state observations up to t:

$$\hat{Y}_{t+h} = \hat{g}_h(Y_t, Y_{t-1}, ..., Y_{t-p-1}). \tag{4}$$

For the sake of the clarity of the notation, note that the symbol $\hat{}$ will generally denote the predictive instance of a given variable throughout the remaining of the paper.

The h-step prediction in (4) can then be achieved by using either a direct or a recursive approach [17], [18]. A direct prediction model predicts the $t + h$ state in one step, while a recursive approach is based on a 1-step prediction model that is used, recursively, forward in time so to reach the $t+h$ state. However, while the recursive approach gives a proper path in the interval $[t, t + h]$, it is h times heavier to compute when compared to the direct model. If only the $t + h$ state is of interest, as for instance in the LKA problem, the usage of the direct approach therefore leads to a significant reduction in the computational burden. Fortunately, in terms of predictive performance, both the approaches have been shown to give similar performance for linear univariate prediction models and on various data sets, see [17].

Another important aspect is that the time series might be over-sampled. This implies that the difference between two consecutive samples is very small, which may complicate the distinction between the contributions from the signal and the embedded noise. If the signals in the time series are sufficiently filtered, it is possible to down-sample the time series without loosing any valuable information as long as the Nyquist sampling theorem is satisfied. A proper down-sampling increases the difference in magnitude between consecutive samples, therefore reducing the risk of fitting to the noise when estimating the model coefficients [19]. Define the time instance set Γ as:

$$\Gamma = \{\gamma_1, \gamma_2, \ \ldots\ , \gamma_d\}, \tag{5}$$

consisting of unique, non-negative sample offsets that determine which old samples should be used as inputs to the model. Using the time instance set (5), the h-step prediction problem (4) can now be formulated as:

$$\hat{Y}_{t+h} = \hat{g}_{\Gamma,h}(Y_{t-\gamma_1}, Y_{t-\gamma_2}, ..., Y_{t-\gamma_d}). \tag{6}$$

Note that the prediction model formulation in (6) is, without loss of generality, an extension of the regular auto-regressive prediction model where the introduced offsets allow for sparse sampling among the observed states. Moreover, the modified prediction model using the sparse sampling technique is able to down-sample the time series in real time, while maintaining the original prediction frequency f_s.

In some cases, however, one may not be interested in predicting all signals $\hat{y}_{t+h,k}$ in $\hat{Y}_{t+h}$, or some of the observed signals in Y_r, i.e., for $r \leq t$, might be redundant and not contributing to the model's predictive performance. All such signals can therefore be dismissed, and the prediction model (6) reformulated in order to predict *specific* signals in the time series, denoted by $\hat{\mathcal{T}}_{t+h} \subseteq \hat{Y}_{t+h}$, such that:

$$\hat{\mathcal{T}}_{t+h} = \hat{f}_{\Gamma,\Psi,h}(Y_{\Psi,t-\gamma_1}, Y_{\Psi,t-\gamma_2}, ..., Y_{\Psi,t-\gamma_d}), \tag{7}$$

where Ψ denotes the signal selection set, i.e., a set indicating the $Q \leq K$ signals that should be used from the Y_t vector such that $Y_{\Psi,t} \subseteq Y_t$. Note that the prediction model (7) has two hyperparameters: (i) the time instance set Γ and (ii) the

signal selection set Ψ. As stated in Sec. III-A, the observed time series (3) is driven by an unknown function g, with the unknown order p. Hence, choosing a model structure, with the corresponding hyperparameters, for the prediction model $\hat{f}_{\Gamma,\Psi,h}$, is therefore not straightforward.

B. Time complexity

As emphasized in Sec. I, the runtime properties of the prediction model is an important aspect for the considered application. The running time of an algorithm is a function of the number of operations and of the time duration of each operation needed to compute the output. However, the running time may vary depending on the hardware, and it is therefore common to abstract upon the running time, typically referred to as analyzing the algorithm's *time complexity* [20]. A forward pass of a (linear) neural network is dominated by summation and multiplication operations. Conceptually for floating-point operations, multiplications are harder to perform than summations. For example, a CPU based on the Intel Silvermont architecture is needing 66% more clock cycles to perform a multiplication operation than an addition operation [21]. Some processors, that use specialized floating-point units (FPU), e.g., CPUs based on the ARM Cortex-M4 architecture, can compute summations and multiplications equally fast [22], but when comparing the energy needed to perform the operations, it has been concluded that a multiplication needs approximately four times more energy than a summation [23]. To generalize the analysis on runtime performance, the results should be hardware-agnostic. The time complexity is therefore defined as the number of multiplications made to compute the output of the prediction model, since they are the most complex and energy demanding operations to compute. Hence, summations and other operations are associated with a zero cost. Recall that the focus on the paper is on the models' relative differences, rather than the absolute true time complexity.

C. Prediction models based on multiple regression

Based on the generic formulation given in (7), two learning based prediction models are implemented and evaluated in this work: a multiple linear regression model and a multiple non-linear regression model.

The MLR model is inspired by the Vector Auto-Regressive (VAR) prediction model used in system identification literature to model linear dynamic processes [24] as well as in econometrics [16]. The VAR model is known to perform well on linear problems and is easy to design since there exists a closed form solution.

The MLP is also straightforward to implement and easy to train, while providing high predictive performance similar to more advanced and computationally demanding non-linear models, as highlighted in, for example, [7], [10]. Another recent study [25], using the same data set as in this work, also indicates that the MLP's performance is even slightly higher than the more advanced, uncertainty aware Gaussian Multiple Layer Perceptron (GMLP) model. Hence, the MLP will be considered as a representation of the state-of-the-art, and used in this paper as a predictive performance upper-bound indication for benchmark and analysis purposes.

1) Multiple linear regression prediction model :
A direct MLR prediction model with $d = |\Gamma|$ number of time instances, is defined as:

$$\hat{\mathcal{T}}_{t+h} = \sum_{i=1}^{d} A_i Y_{\Psi,t-\gamma_i}, \tag{8}$$

where the dependency on the d previous states is determined by the weight-coefficients in the matrices:

$$A_i = \begin{bmatrix} \theta_{1,1}^{(i)} & \dots & \theta_{1,Q}^{(i)} \\ \vdots & \ddots & \vdots \\ \theta_{R,1}^{(i)} & \dots & \theta_{R,Q}^{(i)} \end{bmatrix}, \tag{9}$$

and R and Q are the number of signals in the prediction vector $\hat{\mathcal{T}}_{t+h}$ and in the state vector $Y_{\Psi,t-\gamma_i}$, respectively. The model's coefficients are estimated in closed form using a sequence of samples from the time series. Rewrite (8) in matrix notation such that:

$$\hat{\mathcal{T}}_{t+h} = BZ_t, \tag{10}$$

where,

$$B = [A_1, \dots, A_d] \in \mathbb{R}^{R \times dQ}, \tag{11}$$

$$Z_t = [Y_{\Psi,t-\gamma_1}^{\intercal}, \dots, Y_{\Psi,t-\gamma_d}^{\intercal}]^{\intercal} \in \mathbb{R}^{dQ \times 1}. \tag{12}$$

Assuming there exists N observations of $(\mathcal{T}_{t+h}, Z_t)$ yields:

$$\mathcal{T} = [\mathcal{T}_{t+h} \dots \mathcal{T}_{t+N+h}] \in \mathbb{R}^{R \times N}, \tag{13}$$

$$Z = [Z_t, \dots, Z_{t+N}] \in \mathbb{R}^{dQ \times N}. \tag{14}$$

The model coefficients can now be found by solving the least square (LS) problem:

$$\arg\min_{B} \frac{1}{N} ||\mathcal{T} - BZ||_2^2, \tag{15}$$

which has the optimal closed form solution:

$$B^* = \mathcal{T} Z^{\intercal} (ZZ^{\intercal})^{-1}. \tag{16}$$

As defined in Sec. III-B, the time complexity of the MLR is determined by the number of multiplication operations needed for a forward pass. This yields:

$$TC = dQR, \tag{17}$$

or, in words, defined as the product of the number of time instances, the number of signals and the number of outputs.

2) Multiple non-linear regression prediction model: The MNLR prediction model is, as mentioned before, based on a Multilayer Perceptron model. The MLP model originates from the concept of Feed-Forward artificial Neural Network [26], consisting of an input layer, followed by a number of fully connected hidden layers, and a final output layer. Since the MLP is fully connected, the $i : th$ neuron in the hidden layer L is using a linear combination of the neuron outputs of the previous layer $L - 1$:

$$s_i^L = \sum_{j=1}^{M} w_{i,j}^L o_j^{L-1}, \tag{18}$$

where $w_{i,j}^L$ denotes the weighting parameter for the $i:th$ neuron in layer L on the $j:th$ neuron in hidden layer $L-1$. A neuron's output is then given by:

$$o_i^L = \Upsilon(s_i^L), \tag{19}$$

where Υ is the activation function that operates on the scalar input. The activation function provides non-linearity to the model, where a classical choice is the sigmoid function:

$$\Upsilon(x) = \frac{1}{1+e^{-x}}. \tag{20}$$

The sigmoid function has a consistent derivative and is thereby suitable for the back-propagation training algorithm [27]. On the other hand, it is known to occasionally cause a phenomenon called the *vanishing gradient problem*. The phenomena, which is especially visible for deep MLPs with many hidden layers [28], can prevent the back-propagation algorithm from updating the weights. To avoid this, the Rectified Linear activation function (ReLu) was introduced in [29]. It is worth mentioning that empirical testing showed no significant difference while using a sigmoid function or a ReLu, and therefore the sigmoid function was used in this work.

The time complexity, expressed as the number of multiplications needed for computing the output of an L hidden layers MLP, is given as:

$$TC = dQM + (L-1)M^2 + MR, \tag{21}$$

where M is the number of neurons per hidden layer.

D. Kinematic prediction model

In the literature, there exist different types of kinematic models. Among the most popular is the Constant lateral Velocity (CV) model, which has its strength when the road is fairly straight and where the driver is not driving in a "sporty" manner [30]. Other models such as the Constant Turn-Rate and Velocity (CTRV) model or a Constant Turn-Rate and Acceleration (CTRA) model are preferably used in situations where the vehicle is turning, i.e., on curvy roads, see [30]. However, such models assume that the turn rate over the prediction horizon is constant, which is unlikely on straight roads. Additionally, the turn-rate, in this paper denoted as the yaw-rate signal ω_t, is hard to be precisely measured given the fact that it is easily affected by external sources of noise, such as vibrations. As a consequence, when used on straight roads, it tends to make, quite frequently, predictions where the vehicle is falsely departing from lane.

In the scope of the LKA system considered in this paper, the Operational Design Domain (ODD) is set to driving on highway roads, i.e., roads that are fairly straight. The CV model computes the future distance $d_{t+h}^{\diamond}$ based on the assumption that the lateral velocity remains constant over the prediction horizon. The model is derived by using the heading angle:

$$\psi \approx \frac{d}{dx} p_t^{\diamond}(0), \tag{22}$$

which is approximated, due to the law of small angles, as the first derivative of the lane marker polynomial. The lateral speed is then given by:

$$v_t^{\diamond,lat} = v_t^{long} \sin\psi. \tag{23}$$

The relative lateral distance, h steps ahead, is finally given as:

$$d_{t+h}^{\diamond} = p_t^{\diamond}(0) + v_t^{\diamond,lat} h T_s. \tag{24}$$

It is worth mentioning that, from a time complexity perspective, the constant velocity (CV) model is a very effective prediction model for ADAS applications. The prediction is computed by a few multiplication operations, which makes it computationally competitive, when compared to many of the alternative prediction methods, and especially data driven approaches. The CV model is also commonly used, at least partly, in performance benchmarks, see [31], [32] and [33]. Hence, the CV model will be used later for benchmark purposes, as a lower bound in terms of time complexity and predictive performance.

IV. Data set

In this work, a large set of real-world data is used for the validation of the proposed motion prediction models. The data set has been collected by professional drivers driving a fleet of test-vehicles under different weather conditions, road types and countries such as Sweden, Germany and China. All signals have been retrieved via the vehicle CAN-bus, and each signal is sampled with a sampling frequency of $f_s = 1/T_s = 40$ Hz.

A. Data selection

The quality of the signals in the data set varies with respect to different factors such as weather conditions, worn road markers, etc. As some of the threat assessment methods used in this paper are data-driven, it is of importance that the considered data is consistent and representative of the chosen ODD, i.e., that it properly reflects lane departure situations on highway roads, and therefore effectively excluding situations such as regular lane-changes, queuing, forks, merges and roundabouts.

Let the *extracted data set* be a subset of the real world data set, where each data sample fulfills the following criteria:

- the right and left lane markers are present;
- the lane width is not wider than 4 m;
- the curve radius of the road is larger than 250 m;
- the longitudinal velocity is higher than 60 km/h;
- no turn indicator is used when departing from lane;
- no intended lane change is performed within a 4 s period after a lane departure.

B. Data annotation

The annotation of the extracted data set is straightforward since each signal is a time series. Given a time instant t and a prediction horizon of H seconds, the future lateral distance to the lane marker is simply given by the actual value of $d_{t+h}^{\diamond} = p_{t+h}^{\diamond}(0) = a_{0,t+h}^{\diamond}$, where $h = H/T_s$. Hence,

$$\mathcal{T}_{t+h} = \begin{bmatrix} d_{t+h}^{l} \\ d_{t+h}^{r} \end{bmatrix}. \tag{25}$$

The annotated data is divided into a non-event data set $|\mathcal{B}|=$ 3000 and an event data set $|\mathcal{C}|=$ 12645, where $|\ |$ denotes the number of time series sequences in the set. The non-event set contains 11 seconds long time series sequences of normal driving in lane, i.e., where no lane departures occur, and is used to evaluate the false-positive performance. The event set contains time series sequences where each time series ends with a lane departure, i.e., the shortest distance among $d^{\diamond}_{t+h}$ is equal to 0. The length of the sequences in the event set, in terms of number of samples before the departure occurs, is a design parameter affecting the balance between normal driving and driving leading to a lane departure. The length used in this work is proportional to the prediction horizon, where empirical testing concluded that a length of $4H$ seconds produces a fair trade-off between normal driving and lane departure driving. The event set is divided into an estimation set $|\mathcal{C}^e|=$ 10645 used for model coefficients estimation, a calibration set $|\mathcal{C}^c|=$ 1000 used for performance calibration and a test set $|\mathcal{C}^t|=$ 1000 used for testing the model performance. Note that $\mathcal{C}^e$, $\mathcal{C}^c$ and $\mathcal{C}^t$ are mutually exclusive, and that all data samples have been standardized.

V. Implementation and setup

The prediction models are implemented in Python code, where the MLR model is estimated using the algebraic closed form solution given in (16) and the MNLR is trained using the Keras framework [34] together with TensorFlow [35]. The MNLR model consists of $L = 3$ fully connected hidden layers, with $M = 40$ neurons in each layer and an output layer with $R = 2$ linear neurons, one for each side of the vehicle. The MNLR model's architecture is based on the authors' own experience from the implementation and testing phase, where the number of neurons per hidden layer was found by a simple line search, similar to the method used in [36] and [37], by increasing the number of neurons in steps of 10. The networks were trained using early stopping [38] and a first-order, gradient-based optimizer for stochastic loss functions, called the ADAM optimizer [39], that minimizes the prediction error in terms of Mean Squared Error (MSE). From the preliminary evaluation, it was evident that the MSE did not improve for M values larger than 40. Regarding the number of hidden layers, it has been shown that, at least theoretically, a feed-forward neural network consisting of only one hidden layer is able to estimate any nonlinear function, if the number of neurons are sufficiently many [40]. However, some problems may benefit from deeper networks, e.g., when the problem is believed to rely on a sequence of steps, in the sense that a part of the problem is solved in the first hidden layer and its output is used in the subsequent layer as input to solve the next part of the problem [26]. Hence, in general, it is useful to have a deeper network with more than one hidden layer [26], and consequently a network architecture with several hidden layers has been chosen in this work.

The hardware used for numerical computations is a desktop computer equipped with a 24 core CPU, 192 GB RAM and Windows 10. The closed form solution is computationally efficient (the computation time is less than 5 s per model), while the numerical solution for the MNLR takes significantly more time (approximately 2 hours per model). The CV model is also implemented in Python, and it requires no training as it is parameter-less.

TABLE I
Signal selection sets

	$a^l_{0,t}$	$a^r_{0,t}$	$a^l_{1,t}$	$a^r_{1,t}$	δ_t	ω_t	$a^l_{2,t}$	$a^r_{2,t}$	$a^l_{3,t}$	$a^r_{3,t}$	rw^l_t	rw^r_t	v_t
$\Psi 0$	x	x											
$\Psi 1$	x	x	x	x									
$\Psi 2$	x	x	x	x	x								
$\Psi 3$	x	x	x	x	x	x							
$\Psi 4$	x	x	x	x	x	x	x	x					
$\Psi 5$	x	x	x	x	x	x	x	x	x	x			
$\Psi 6$	x	x	x	x	x	x	x	x	x	x	x	x	
$\Psi 7$	x	x	x	x	x	x	x	x	x	x	x	x	x

A. Prediction horizons

The length of the prediction horizon is related to how much time is needed for an automated steering maneuver to avoid a future lane departure, denoted as the Time Duration of Maneuver (TDM). The Time to Lane Crossing (TLC) is a metric for how much time remains before the vehicle is departing from the lane [32], [41]. The relationship between TLC and the TDM is given as:

$$TDM = \frac{d^{\diamond}_t - 0}{(\frac{v^{\diamond,lat}_t - 0}{2})} = 2 \times TLC, \tag{26}$$

where an automated maneuver is mitigating a lane-departure by controlling the vehicle towards a zero distance to the lane marker and zero heading. Based on empirical experience, the TDM is typically no longer than 3.5 s, which corresponds to a $TLC = 1.75$ s. Hence, TLC value sets the requirement for the prediction horizon, and ultimately the resulting TDM values [1].

Both the MLR and the MNLR prediction models are implemented for six different prediction horizon values, namely $[0.5, 0.75, 1, 1.25, 1.5, 1.75]$ seconds. The lowest prediction horizon (0.5 s), though maybe too short for practical use, remains pertinent for the relative performance analysis.

B. Signal selection

Selecting the input signals in Ψ, i.e., which signals should be included in $\hat{f}_{\Gamma,h}$, is a trade-off between the model precision and the computational complexity, where more information may give a better precision but increases the computational burden. However, it is not guaranteed that more input signals yield an increased precision. Redundant or contradictory signals might reduce the effectiveness of the model training and give a poor generalization capabilities on new data.

Different combinations of signals have been gathered in *signal selection sets*, see Tab. I. The simplest signal selection set $\Psi 0$, includes only the distance to the two sides' lane-markers, while the most rich set $\Psi 7$ contains all available signals. The sets grow incrementally with one signal type at

[1] Remark that the required maneuver time, in the general case, is dependent on how strong the vehicle's steering actuator is and the safety constraints on the system.

TABLE II
TIME INSTANCE SETS

	Sample rate	Sample offsets	$\|\Gamma\|$
$\Gamma 0$	40 Hz	0, 1, 2, 3, 4, ..., 37, 38, 39	40
$\Gamma 1$	20 Hz	0, 2, 4, 6, 8, ..., 36, 38, 40	21
$\Gamma 2$	10 Hz	0, 4, 8, 12, ..., 32, 36, 40	11
$\Gamma 3$	5 Hz	0, 8, 16, 24, 32, 40	6
$\Gamma 4$	2.5 Hz	0, 16, 32	3
$\Gamma 5$	1.25 Hz	0, 32	2
$\Gamma 6$		0, 1, 2, 3, 5, 9, 15, 24, 39	9
$\Gamma 7$	40 Hz	0, 1, 2	3

the time, in order to be able to test the importance of each individual signal in a structured way.

As mentioned in Sec III-A, a consecutive sequence of old samples of the signals are typically used as inputs to the model. However, the notion of sparse sampling introduced in (6) allows for an arbitrarily sequence of old samples, which can be used for down-sampling. The down-sampling is made by consecutive divisions by two, yielding different *time instance sets*, denoted $\Gamma 0$ to $\Gamma 5$, as shown in Tab. II, where the lowest sampling frequency is 1.25 Hz. In addition, $\Gamma 6$ corresponds to a logarithmic sampling, which showed good performance in a previous study [15]. Observe that the logarithmic sampling is dense for the most recent samples and sparse for the older samples. The intuition behind $\Gamma 6$ is that more recent information should better describe the current situation, while the old information can contribute with the long-term trend. Hence, the logarithmic pattern can be viewed as a way to weight between recent and old information. Remark that $\Gamma 0 - \Gamma 6$ use up to approximately 1 s old data samples, which has been shown to be a preferable historic depth according to the findings in [15]. Finally, the last pattern $\Gamma 7$ uses only the 3 most recent samples to illustrate how the performance is affected when only the most recent data is taken into consideration.

The time complexity of the MLR and the MNLR model, as defined in (17) and (21), respectively, is presented in Tab. III. From the table one can observe significantly higher values of the time complexity for the MLNR model, when compared to the MLR model. It is also evident that the number of inputs should be chosen with care in order to potentially avoid high time complexity, especially for the MNLR model.

VI. RESULTS AND ANALYSIS

This section is divided in three parts. In Sec. VI-A, the data properties are analyzed, as an important basis for the latter discussion on how signals selection and down-sampling affect the predictive performance of the different models. In Sec. VI-B, the predictive performance is analyzed in terms of MSE. Finally, Sec. VI-C analyzes the performance in a LKA application. More precisely, the MSE analysis is used to select sufficient input signals and a proper down-sampling rate for the prediction model. The resulting models from the MSE analysis are then used in the analysis of the LKA application. Remark that both the MSE and LKA performance analysis are important to this study: the MSE, based on sample-wise evaluation, indicates how well the model performs in average

TABLE III
TIME COMPLEXITY FOR COMBINATIONS OF TIME INSTANCE AND SIGNAL SELECTION SET.

		$\Psi 0$	$\Psi 1$	$\Psi 2$	$\Psi 3$	$\Psi 4$	$\Psi 5$	$\Psi 6$	$\Psi 7$
MLR	$\Gamma 0$	160	320	400	480	640	800	960	1040
	$\Gamma 1$	84	168	210	252	336	420	504	546
	$\Gamma 2$	44	88	110	132	176	220	264	286
	$\Gamma 3$	24	48	60	72	96	120	144	156
	$\Gamma 4$	12	24	30	36	48	60	72	78
	$\Gamma 5$	8	16	20	24	32	40	48	52
	$\Gamma 6$	36	72	90	108	144	180	216	234
	$\Gamma 7$	12	24	30	36	48	60	72	78
MNLR	$\Gamma 0$	6480	9680	11280	12880	16080	19280	22480	24080
	$\Gamma 1$	4960	6640	7480	8320	10000	11680	13360	14200
	$\Gamma 2$	4160	5040	5480	5920	6800	7680	8560	9000
	$\Gamma 3$	3760	4240	4480	4720	5200	5680	6160	6400
	$\Gamma 4$	3520	3760	3880	4000	4240	4480	4720	4840
	$\Gamma 5$	3440	3600	3680	3760	3920	4080	4240	4320
	$\Gamma 6$	4000	4720	5080	5440	6160	6880	7600	7960
	$\Gamma 7$	3520	3760	3880	4000	4240	4480	4720	4840

for all driving data; the LKA analysis, focusing on the overall scenario, indicates how well the system behaves when needed.

A. Data analysis

This section provides a detailed overview to the signals' properties in terms of empirical distribution and frequency content.

1) Empirical distributions: The empirical distribution of the different signals is depicted in Fig. 2. As stated in Sec. II, the road polynomial approximates the roads geometry relatively the vehicle. Hence, the distance to the lane marker at time t is given by $p_t^\diamond(0) = a_{0,t}^\diamond$. As seen in Fig. 2a, the empirical distributions for the distance on the left and right side are unimodal, non-symmetric and take values in between -0.1 and 2 m. Furthermore, it can be observed that the density around 0 is low, meaning that very little data exist for that distance to the lane marker. Hence, a learning-based prediction model need to learn how to predict small distances to the lane marker mainly using data from the peak of the distribution.

Let the heading angle towards the lane marker be derived by using the change rate of the distance, $\frac{d}{dx}p_t^\diamond(0) = a_{1,t}^\diamond$, as:

$$\psi^\diamond = \tan^{-1} a_{1,t}^\diamond, \tag{27}$$

where $\psi^\diamond \approx a_{1,t}^\diamond$ for low numerical values of the change rate. Hence, the heading angle take values in the interval $[-0.03, 0.03]$ rads, which corresponds to ± 1.7 degrees. Notice that a heading angle of 1 degree and a speed of 90 km/h yields a lateral velocity equal to 0.44 m/s. Moreover, the road curvature can be computed as:

$$\kappa = \frac{\left\| \frac{d^2}{dx^2} p_t^\diamond(0) \right\|_2}{\left(1 + \left(\frac{d}{dx} p_t^\diamond(0)\right)^2\right)^{\frac{3}{2}}} \approx \left\| 2a_{2,t}^\diamond \right\|_2, \tag{28}$$

which yields a road radius $R = 1/\kappa$. Looking now to the distribution of the curvature in Fig. 2c, one can observe that it is very dense for low values, which corresponds to a tendency of having straight roads. Furthermore, the curvature's change-rate $a_{3,t}^\diamond$ has a unimodal and symmetric distribution, as seen in Fig. 2d.

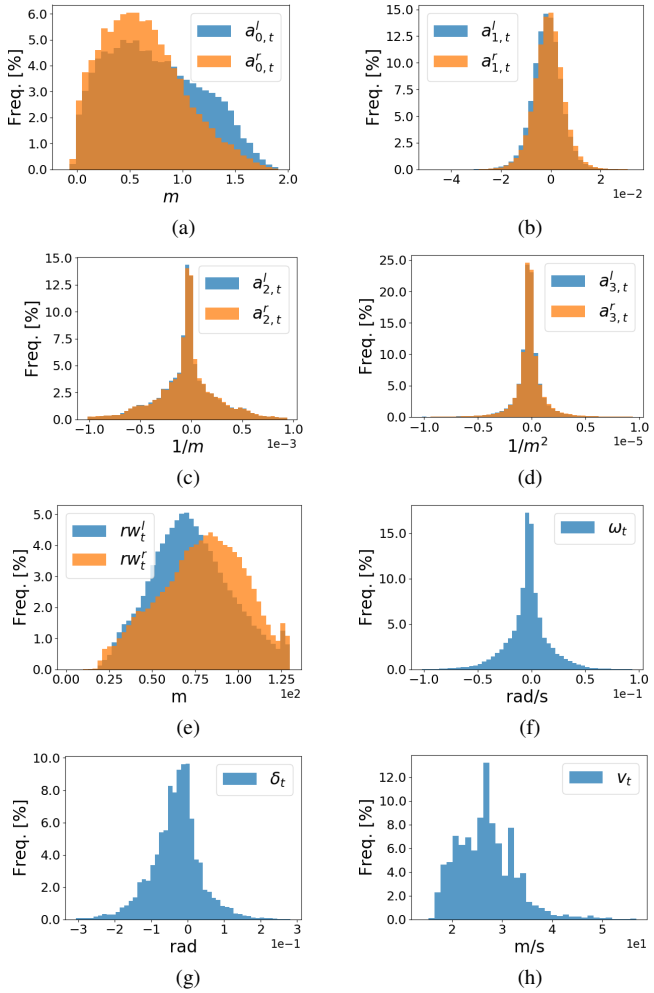

Fig. 2. Histograms showing the statistical properties of: the coefficients of the lane marker polynomials (a, b, c and d) and the range of view (e), for the left (l) and right (r) side, as well as of the yawrate (f), the front wheel angle (g) and the longitudinal velocity (h).

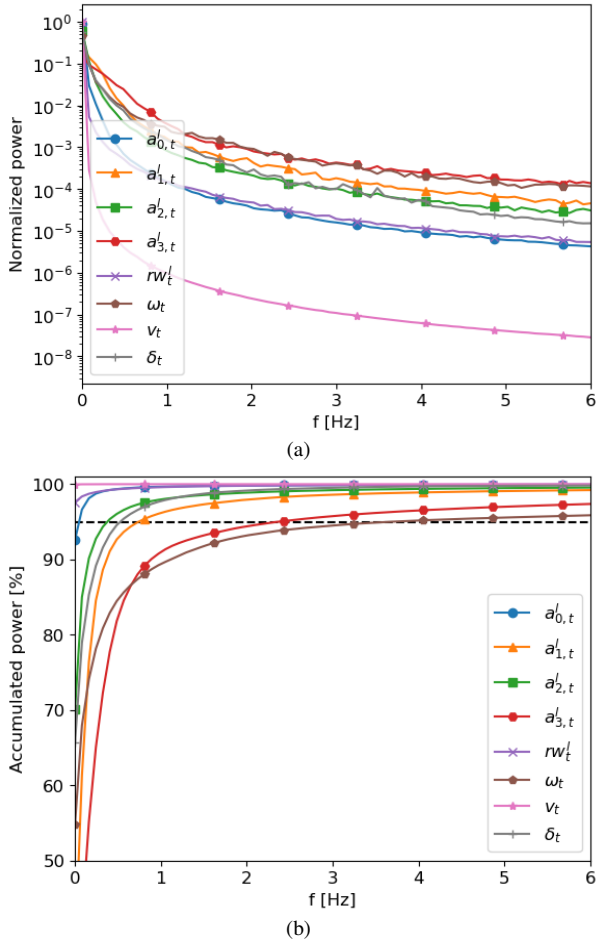

Fig. 3. Panel (a) depicts the normalized discrete power spectrum density of the time series signals, while panel (b) is the cumulative power spectrum density. The black dashed line indicates the 95% level.

The distributions of the range of view $rw_t^{\diamond}$, presented in Fig. 2e, differ significantly from the previous figures in terms of the behavior of the peaks. To the left side, the peak is located at 70 m while on the right side the peak is at 90 m. This seems to indicate that the right hand side lane marker is more visible to the camera sensor on average, which might be explained by the assumption that the lane markers to the right is less worn and more seldomly occluded by other vehicles.

Fig. 2f and in Fig. 2g show the distribution of the yaw rate ω_t and steering angle δ_t respectively. One can observe rather symmetric distributions, though a small bias on the steering angle distribution can be perceived, probably caused by miss-calibration of the sensors.

Finally, Fig. 2h shows the distribution of the vehicle's speed v_t. It shows the characteristics of a multimodal distribution, as a natural effect of driving on roads with different speed limits.

2) Frequency analysis: The Power Spectrum Density (PSD) is used to visualize the bandwidth wherein the signal has power [42], and is presented in Fig. 3a. The discrete PSD function is derived by averaging over an ensemble of 500 PSDs, calculated from random time series sequences in the estimation data, and all of equal length. For convenience, the PSD is normalized such that the sum over all power equals to 1. However, the signals may contain high frequency noise, which would yield a very high bandwidth. In this work, the bandwidth is defined as the frequency interval $[-f_b, f_b]$ such that the Cumulative sum over PSD (CPSD), given as:

$$CPSD = \sum_{k=-f_b}^{f_b} PSD(k), \tag{29}$$

contains 95% of the total signal power given by the interval $[-\infty, \infty]$. Fig. 3b shows the CPSD for all signals. One can observe that the bandwidth of the velocity v_t and range of view rw_t signals are approximately 0, which indicates that they are almost constant within the 500 time series sequences, on average. The higher order coefficients $a_{1,t}, a_{2,t}, a_{3,t}$, together with the steering wheel angle δ_t and the yaw rate ω_t, have a higher bandwidth, where the largest is 3.7 Hz for the yaw rate. The distance to the line-marker $a_{0,t}$ is of special interest as it is used as the ground truth in the annotation of the estimation data. The $a_{0,t}$ signal has a low bandwidth of less than 0.1 Hz, which means that it is changing slowly. From a prediction model perspective, it is beneficial that the distance values

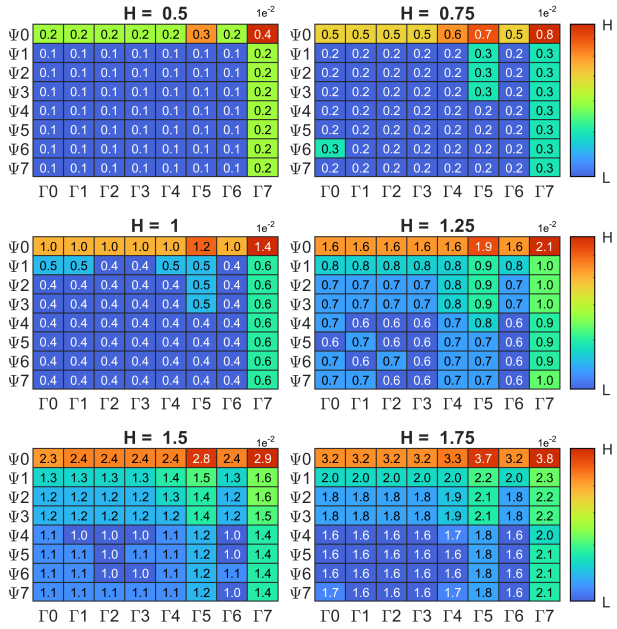

Fig. 4. Heat map for the MLR model, representing the relative performance for different combinations of signals and time instances used in the sampling. Red and blue color indicate a high and low MSE value, respectively.

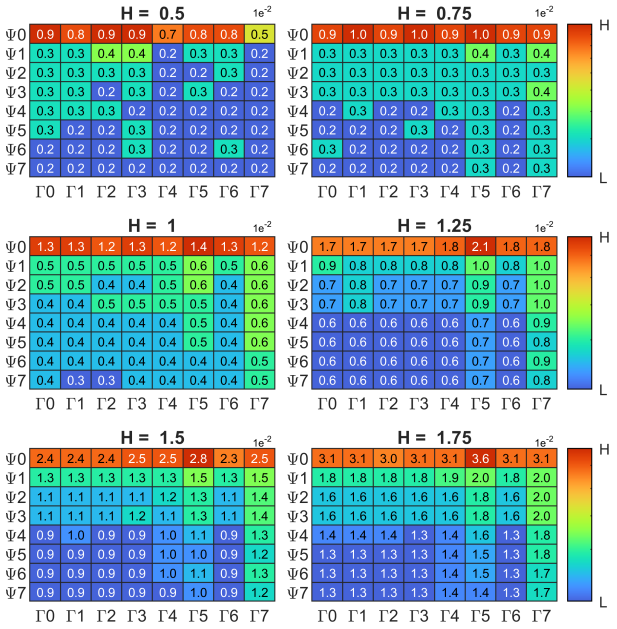

Fig. 5. Heat map for the MNLR model, representing the relative performance for different combinations of signals and time instances used in the sampling. Red and blue color indicate a high and low MSE value, respectively.

varies slowly, but nevertheless it remains challenging to predict its future changes. Furthermore, one can also observe that the signals have significant power within the frequency interval $[0, 4]$ Hz. Hence, in accordance with the Nyquist sampling theorem, it should be possible to retrieve the signals using a sample rate as low as 8 Hz without loss of information. The power spectrums in Fig. 3a and Fig. 3b show that the signals' power are very low for frequencies higher than 4 Hz, which indicates that the signals are sufficiently low-pass filtered to allow down-sampling without causing aliasing.

B. Model selection based on MSE

The goal of this section is to find suitable hyperparameters of the learning-based models, i.e., the signal-selection set Ψ and time instance set Γ. For that purpose, an exhaustive grid search approach was used, spanning all combinations of the signal-selection set (see Tab. I), the time instance set (see Tab. II), and different prediction horizons values. The results are given in Fig. 4 and Fig. 5, for the MLR and MNLR model, respectively, and are presented in the form of a heat map, where the blue cells correspond to lower MSE values. One can see that the MNLR is over-fitting the training data for the shorter prediction horizons 0.5 and 0.75, especially when using only the distances to the lane markers ($\Psi 0$) as input signals. This is evident by the fact that the MNLR is consistently performing worse than the MLR model for those horizons. As stated in Sec. V-B, the signal selection set grows incrementally with one new type of input signal for each new configuration. Hence, one can see that the $a_1^\diamond$, δ and $a_2^\diamond$ signals (in configurations $\Psi 1$, $\Psi 2$ and $\Psi 4$) contribute with significant improvements compared to only using the $a_0^\diamond$ signals in $\Psi 0$. Adding more signals, $\Psi 5$-$\Psi 7$, yields no further improvement for the MLR, but a small benefit could be seen for the MNLR.

In terms of time instances, the MLR model seems to be insensitive to the down-sampling for sampling rates equal or higher than 5 Hz ($\Gamma 3$). The MNLR model shows similar characteristics, excluding the results for 0.5 s prediction horizon that is affected by the over-fitting, as mentioned before. Comparing the results from the MLR and the MNLR model shows that the MLR is a better choice for prediction horizons up to 1 s, while the MNLR is better for prediction horizons longer than 1.25 s. The logarithmic sampling used in $\Gamma 6$ shows similar performance as $\Gamma 2$, but uses only 9 time instances instead of 11. The consecutive and shallow sampling used in $\Gamma 7$ is under-performing when compare to the others.

Based on these results, and the findings in Sec. VI-A2, $\Gamma 3$ seem to be the preferred choice of down-sampling rate, since it provides approximately equal performances as the $\Gamma 0 - \Gamma 2$ and $\Gamma 6$, despite having a lower number of input signals. In terms of signal-selection, Ψ_4 is the preferred choice, since it yields a competitive predictive performance while keeping the complexity cost low, see Tab. III for details on complexity.

C. Performance analysis for LKA

The models' predictive performance is further analyzed in terms of the LKA application. The test data set $\mathcal{C}^t$ is used to compute the True-Positive Rate (TPR), i.e., how many True-Positive (TP) activations are triggered when needed. However, even though the prediction model is trained on a specific prediction horizon H, it is unlikely that the corresponding triggering time on test data is identical to the designed prediction horizon. Therefore, a TP is encountered if the prediction model detects a true lane departure up to $2H$ seconds before the departure takes place. The False-Positive Rate (FPR) is computed using the non-event set $\mathcal{B}$ given τ, where an False-

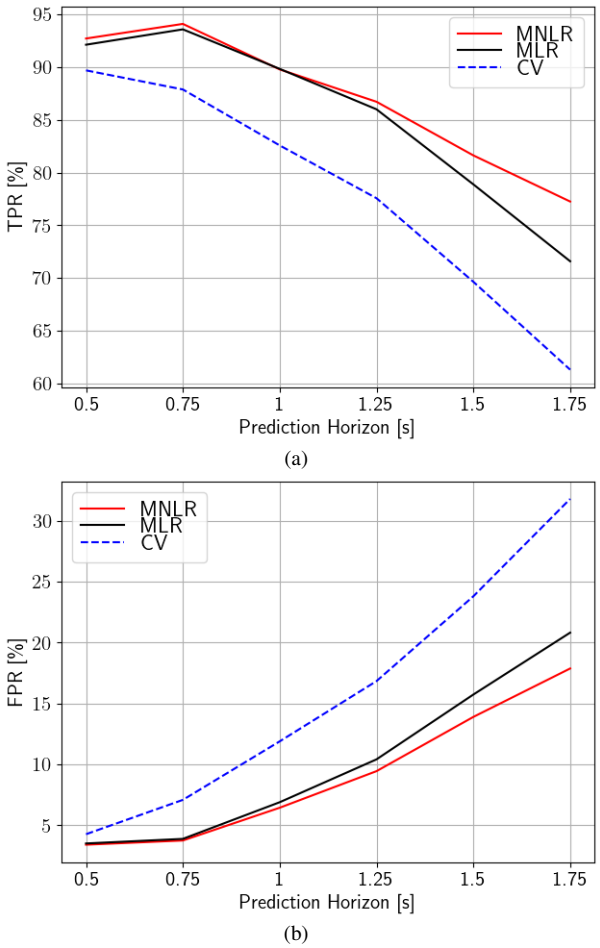

Fig. 6. The predictive performance in the LKA application is derived by using the signal selection set $\Psi 4$ and time instance set $\Gamma 3$. The true positive rate and the false positive rate are shown in panel (a) and (b), respectively.

Positive (FP) event is encountered if an unwanted activation is triggered, i.e., any activation in the non-event set $\mathcal{B}$.

However, a fair comparison between prediction models in terms of TPR and FPR can only be achieved if they are tuned to the same mean triggering time. Note that, for a perfect prediction model, the mean triggering time, obtained for $\tau = 0$, should be equal to H s. Unfortunately, in practice, there are no such guarantees with respect to the mean triggering time. However, the mean triggering time can be calibrated to $\bar{t}_h = H$ by adjusting the threshold τ. The calibration is performed for all models using a small calibration data set $\mathcal{C}^c$.

Based on the results of Sec. VI-B, the MLR and MNLR models are used with the signal selection set $\Psi 4$ and down-sampling rate $\Gamma 3$. Fig. 6 shows the performance in terms of TPR and FPR for the MLR, MNLR and CV models, when used in the LKA application. The relative performance of the learning-based models, when compared to the CV model, improves as the prediction horizon increases. For the shorter prediction horizons $0.5-1$ s, the performance is similar for the MLR and MNLR, while the MNLR performs slightly better for longer horizons. However, as seen in Tab. III, the MLR and MNLR have a time complexity of 96 and 5200, respectively. Hence, the improvement in performance, when using a MNLR instead of a MLR, comes with the cost of higher computational demands.

VII. Conclusions

This work addresses performance and implementation aspects of a multiple linear regression (MLR) model in the scope of unintended lane-departure detection for automotive applications. Using a real world data set, the MLR is benchmarked against a multiple non-linear regression (MNLR) model and a kinematic, constant velocity model. The underlying dimensions of the analysis are the predictive performance and the corresponding computational complexity. Aspects such as input signal selection, down-sampling of the input data, as well as model selection, are also discussed.

The results show that the MLR prediction model, despite its moderately higher computational costs than the CV model, largely outperforms the kinematic model. The results presented also show that the MLR prediction model has approximately the same performance as the MNLR model for prediction horizons up to 1 s, and slightly reduced performance for longer prediction horizons. However, considering the significantly lower time complexity of the linear alternative, these results seem to indicate that the linear model is a good choice if the computational power is limited, offering a good trade-off between predictive performance and time complexity.

Regarding the real world data, the results show that the most important input signals are the road geometry and the front wheel angle. The results indicate that the sampling rate can be as low as 5 Hz without sacrificing the predictive performance of unintended lane-departures detection. Furthermore, it is evident that the down-sampling significantly reduces the computational complexity for both the MLR and MNLR prediction model, which lowers the demand for fast and expensive hardware for real-time computations.

Future work should analyze the MLR model's performance for different scenarios, such as collision detection with oncoming traffic, for instance. Additionally, other sources of information might be explored, such as driver monitoring cameras or vehicle-to-vehicle (V2V) communication.

Acknowledgments

This research was supported by Zenseact and the Swedish Governmental Agency for Innovation Systems/FFI through the contracts 2014-05621 and 2019-05828.

References

[1] D. L. Hendricks, M. Freedman, J. C. Fell *et al.*, "The relative frequency of unsafe driving acts in serious traffic crashes," United States. National Highway Traffic Safety Administration, Tech. Rep. DOT-HS-809-206, 2001.

[2] C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2017.

[3] J. Dahl, G. Rodrigues de Campos, C. Olsson, and J. Fredriksson, "Collision avoidance: A literature review on threat-assessment techniques," *IEEE Trans. on Intelligent Vehicles*, vol. 4, no. 1, pp. 101–113, 2019.

[4] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *IEEE conference on computer vision and pattern recognition*, pp. 2174–2182, 2017.

[5] L. Chi and Y. Mu, "Learning end-to-end autonomous steering model from spatial and temporal visual cues," *Proceedings of the Workshop on Visual Analysis in Smart and Connected Communities*, pp. 9–16, 2017.

[6] K. Min, D. Kim, J. Park, and K. Huh, "RNN-based path prediction of obstacle vehicles with deep ensemble," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 10, pp. 10 252–10 256, 2019.

[7] V. Ilić, D. Kukolj, M. Marijan, and N. Teslić, "Predicting positions and velocities of surrounding vehicles using deep neural networks," *IEEE Zooming Innovation in Consumer Technologies Conference*, pp. 126–129, 2019.

[8] H. Cui, T. Nguyen, F.-C. Chou, T.-H. Lin *et al.*, "Deep kinematic models for kinematically feasible vehicle trajectory predictions," *IEEE International Conference on Robotics and Automation*, pp. 10 563–10 569, 2020.

[9] X. Mo, Y. Xing, and C. Lv, "Interaction-aware trajectory prediction of connected vehicles using cnn-lstm networks," *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*, pp. 5057–5062, 2020.

[10] Y. Xing, C. Lv, H. Wang, D. Cao, and E. Velenis, "An ensemble deep learning approach for driver lane change intention inference," *Transportation Research Part C: Emerging Technologies*, vol. 115, p. 102615, 2020.

[11] J. Dahl, R. Jonsson, A. Kollmats, G. R. de Campos, and J. Fredriksson, "Automotive safety: a neural network approach for lane departure detection using real world driving data," *IEEE Intelligent Transportation Systems Conference*, 2019.

[12] R. Sedgewick and K. Wayne, *Algortihms, fourth edition*. Addison-Wesley, 2011.

[13] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education, 2006.

[14] J. Jagst, "Challenges impacting the mass deployment of autonomous vehicles," *ATZelectronics worldwide*, vol. 15, no. 1, pp. 8–13, 2020.

[15] J. Dahl, G. R. de Campos, and J. Fredriksson, "A path prediction model based on multiple time series analysis tools used to detect unintended lane departures," *IEEE Intelligent Transportation Systems Conf.*, 2020.

[16] H. Lütkepohl, *New introduction to multiple time series analysis*. Springer Science & Business Media, Berlin, 2005.

[17] H. Cheng, P.-N. Tan, J. Gao, and J. Scripps, "Multistep-ahead time series prediction," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 765–774, 2006.

[18] S. B. Taieb, G. Bontempi, A. F. Atiya, and A. Sorjamaa, "A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition," *Expert systems with applications*, vol. 39, no. 8, pp. 7067–7083, 2012.

[19] D. Pollock, "Stochastic processes of limited frequency and the effects of oversampling," *Econometrics and Statistics*, vol. 7, pp. 18 – 29, 2018.

[20] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson Education, 2006.

[21] *Intel 64 and IA-32 Architectures Optimization Reference Manual*, Intel, 2016.

[22] *Cortex-M4 Technical Reference Manual*, ARM, 2010.

[23] M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. IEEE, 2014, pp. 10–14.

[24] L. Ljung, "System identification: theory for the user," *Prentice Hall PTR*, 1999.

[25] J. Larsson and M. Sjöstedt, "A lane departure detection system based on uncertainty aware machine learning," Master's thesis, Chalmers University of Technology, Electrical Engineering, 2020.

[26] I. Goodfellow, Y. Bengio *et al.*, *Deep Learning*. MIT Press, 2016.

[27] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[28] S. Hochreiter, Y. Bengio, P. Frasconi *et al.*, "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *Field Guide to Dynamical Recurrent Networks*, J. F. Kolen and S. C. Kremer, Eds. Wiley-IEEE Press, 2001.

[29] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[30] 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.

[31] A. Zyner, S. Worrall, and E. Nebot, "Naturalistic driver intention and path prediction using recurrent neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1584–1594, 2019.

[32] W. Wang, D. Zhao, W. Han, and J. Xi, "A learning-based approach for lane departure warning systems with a personalized driver model," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 10, pp. 9145–9157, 2018.

[33] F. Wirthmüller, J. Schlechtriemen, J. Hipp, and M. Reichert, "Towards incorporating contextual knowledge into the prediction of driving behavior," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[34] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[35] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," https://tensorflow.org, 2015.

[36] G. Wang, J. Qiao, J. Bi, W. Li, and M. Zhou, "Tl-gdbn: Growing deep belief network with transfer learning," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 874–885, 2018.

[37] G. Wang, Q.-S. Jia, J. Qiao, J. Bi, and M. Zhou, "Deep learning-based model predictive control for continuous stirred-tank reactor system," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[38] Y. Yao, L. Rosasco, and A. Caponnetto, "On early stopping in gradient descent learning," *Constructive Approximation*, vol. 26, no. 2, pp. 289–315, 2007.

[39] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2015.

[40] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[41] W. van Winsum, K. A. Brookhuis, and D. de Waard, "A comparison of different ways to approximate time-to-line crossing during car driving," *Accident Analysis & Prevention*, vol. 32, no. 1, pp. 47–56, 2000.

[42] P. Stoica, R. L. Moses *et al.*, *Spectral analysis of signals*. Pearson Prentice Hall Upper Saddle River, NJ, 2005.

John Dahl received the M.Sc. degree in Systems, Control and Mechatronics in 2013 from Chalmers University of Technology, Sweden, where he is currently working toward the Ph.D. degree. He is currently doing research with the Department of Electrical Engineering and Zenseact AB, Gothenburg. His research interests include active safety and estimation techniques, in particular, for threat-assessment and decision-making in collision-avoidance systems.

Gabriel Rodrigues de Campos received his Ph.D. in Automatic Control in 2012 from Grenoble University/Grenoble INP, France. He is currently a researcher with Zenseact in Gothenburg, Sweden. Prior to joining the Zenseact, he was a postdoctoral fellow with the Department of Signals and Systems, Chalmers University of Technology, Sweden and the DEIB, Politecnico di Milano, Italy. His research interests include cooperative and distributed control, safety assurance, and threat-assessment and decision-making techniques.

Jonas Fredriksson received his M.Sc. in Computer Science Engineering from Luleå University of Technology, Sweden in 1997 and Ph.D. in Automatic Control from Chalmers University of Technology, Sweden in 2002. Currently, he is Professor in Mechatronics at the Department of Electrical Engineering at Chalmers. His research activities include modeling, control and simulation, with a special interest in automotive applications.