Automotive Safety: a Neural Network Approach for Lane Departure Detection using Real World Driving Data

John Dahl[1,2,3], Rasmus Jonsson[1], Anton Kollmats[1], Gabriel Rodrigues de Campos[1], and Jonas Fredriksson[3]

***Abstract*— Lane departures, where the vehicle leaves the lane due to driver inattention, drowsiness, or incorrect situation assessment, are one of the most serious accident and fatality prone scenarios. To further improve traffic safety, we are asking the question: How much can a neural network approach improve the reliability of lane departure predictions compared to traditional model-based methods? Our results show a relative improvement in reliability of 7% in terms of true positive rate and 22% reduction of the false positive rate with respect to a constant velocity model method. The key contributions of this work are the introduction of sparse sampling in the input data, a thorough comparison with a baseline solution, and the evaluation on real world driving data.**

I. INTRODUCTION

Nowadays, new vehicles come equipped with ever-more Advanced Driver Assistance Systems (ADAS), designed to support drivers in a wide range of scenarios whenever the drivers fail to maintain safety. Nevertheless, a major theoretical challenge in the presence of human drivers remains how to precisely distinguish a safe (though perhaps aggressive) driving behaviour from an unsafe one. The reader can refer to [1] and [2] for a recent review on threat assessment and decision making for automotive applications.

One of the situations most prone to serious accidents and fatalities are run-off-road and head-on scenarios. Over the last two decades different solutions have been proposed to this problem, in particular for the design of Lane Departure Warning (LDW) and Lane Keeping Aid (LKA) systems. A simple yet powerful indicator of unintended lane departure is the Time-to-Line-Crossing (TLC), defined as the time duration available before a specified part of vehicle crosses the lane boundary. Leveraging the notion of TLC and Distance-To-Line-Crossing (DLC), [3] proposed and analyzed different methods for TLC computation. In [4], authors used set invariance theory and reachability analysis tools for the design of two different model-based threat assessment methods based on vehicle, driver and driver-in-the-loop models. In [5] authors proposed a learning-based approach that leverages a personalized driver model, obtained by combining a Gaussian mixture model and a hidden Markov model. This personalized driver model is then used for the design of an online model-based lane departure predictor, which was benchmarked to a basic time-to-lane-crossing method (identical to the CVM model considered later in this paper). In [6], authors tried to identify driver correction actions with respect to imminent lane departures. The key concepts behind this work are driver behaviour models of the Directional Sequence of Piecewise Lateral Slopes (DSPLS) representing lane-crossing and driver correction events. Within the scope of data-driven methods, the authors of [7] used a feed forward neural network to assess the risk of an unintended lane departure, using a data-set derived from human-driving in a simulator. While the data gathered through a simulator might capture the realistic behaviour of a human-driver, the environment is still assumed to be deterministic and a simplified representation of the real world, therefore disregarding measurement uncertainties and changing environment conditions. Using the same data set as in [7], a Support Vector Machine (SVM) methodology is also proposed in [8]. Authors exploit a nonlinear binary SVM with different kernel functions, and provide prediction results for 0.2s and 0.4s horizons.

In this work, we present a threat assessment and decision making method for a LKA system using a data-driven approach, where the overall goal is to predict whether a lane departure will occur within specific time horizons prior to an event. We use a Multi-Layer Perceptron (MLP) network and evaluate and benchmark our results against a classical kinematic constant velocity model using a real world driving data set. The contributions of this work are two-folded: i) a data-driven threat assessment method that uses sparsely sampled data as input and ii) an evaluation and benchmarking analysis performed in real data. Regarding the second topic, we will argue that our MLP method can outperform a constant velocity model, which in many cases remains an acceptable choice for a simplistic threat assessment. It is worth mentioning that a run-off-road scenario is similar in many aspects to a lane changing scenario, see for example [9], [10]. However, an important distinction between the two cases is that run-off-road scenarios are usually due to driver unawareness and distraction, while lane changing maneuvers are normally voluntary.

The remainder of the paper is organized as follows: Section II introduces the real world data set, while Section III describes the proposed threat assessment method. Section IV covers the evaluation procedure, while the experimental results are presented in Section V. Finally, Section VI presents our conclusions and future perspectives.

Work supported by Zenuity AB, Vinnova and AI Innovation of Sweden.

[1] Zenuity, Gothenburg, Sweden. `name.surname@zenuity.com`
[2] AI Innovation of Sweden, Gothenburg, Sweden.
[3] Electrical Engineering Department, Chalmers University of Technology, Gothenburg, Sweden. `name.surname@chalmers.se`

II. Real world driving data set

In this paper we use a vast data set collected by professional drivers driving a fleet of test-vehicles under different weather conditions, on various roads and in different countries, such as Sweden, Germany and China.

A. Signal definition

The test-vehicles were equipped with a front looking vision system able to measure the geometries of the lane markings relative the ego-vehicle, as illustrated in Fig. 1. Here, d^l and d^r denote the distance from the front bumper of the vehicle to the lane-markers on the left and right side, respectively, while ψ^l and ψ^r represent the heading (i.e., the angle of the vehicle) with respect to the left and right lane marker, respectively. Note that, if the absolute distance between two lane markers is constant, then the left and right heading angles are approximately equal. However, in situations where the road width increases/decreases, the headings can significantly differ and even have different signs. The ego-vehicle's longitudinal velocity and acceleration are denoted by v^{long} and a^{long}, respectively.

B. Data selection

The quality of the data set varies with respect to different factors such as weather conditions or worn road markers. Since the proposed threat assessment method is data-driven, it is paramount that the data is consistent and representative of the chosen scenario, i.e., that it properly reflects lane departure situations. Let the *extracted data set* be a subset of the real world data set, where each data sample fulfills the following criteria:

- sensor measurements of right lane marker are available;
- sensor measurements of left lane marker are available;
- 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 lane change is performed within a 4 s period after a lane departure;

Since the MLP is trained given a specific set of inputs, we ensured that signal measurements are always present. A maximum road width is used to exclude fork and lane merging scenarios where the road width tends to be large. Furthermore, we limited the scope to highway roads by excluding small road radius and low velocities. We also consider an unintended lane-departure to be characterized by the absence of any obvious signs of an intentional departure by the driver. Hence, any departure where the turn indicator is used has been excluded. Finally, to ensure that the driver is not conducting a lane-change, the vehicle must return to its ego-lane within a reasonable time after the departure.

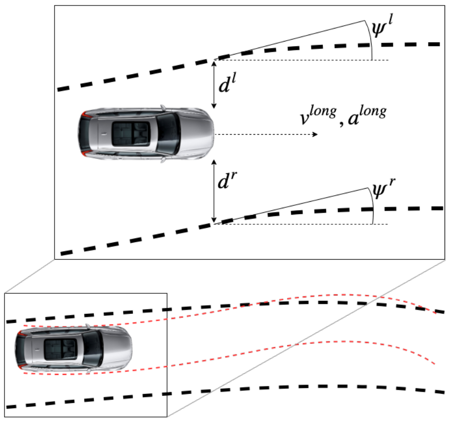


Fig. 1. Scenario illustration: the vehicle, following the path represented by the dashed red lines, is going to depart from the lane by crossing the lane marker (black dashed lines) on the left hand side.

C. Data annotation

The extracted data set described before is divided into two subsets, the first denoting the *event set* $\mathcal{C}$, and the second the *non-event set* $\mathcal{B}$. The event set is used for training the network and the evaluation of the positive performance, while the non-event set is used to evaluate the negative performance. In the scope of this work, we define the positive performance as the threat assessment method's ability to detect a lane departure, and the negative performance as the threat assessment method's ability to avoid triggering erroneous interventions.

1) Construction of the event set $\mathcal{C}$: A simple linear search algorithm is used to find events in the data set where the vehicle is departing from the ego-lane, i.e., whenever d^l or d^r change from a positive to a negative value.

For each event, we define t_m as the time instant when the vehicle is departing from its lane. Let the event be represented by a tuple $c \in \mathcal{C}$ containing the input and target output data such as:

$$c = \langle \mathcal{U}^c_{t_1:t_m}, \mathcal{T}^c_{t_1:t_m} \rangle, \tag{1}$$

where the input is a tuple of time series which are m samples long, covering a time-span[1] $[t_1, t_m]$, and given as:

$$\mathcal{U}^c_{t_1:t_m} = \langle d^l_{t_1:t_m}, d^r_{t_1:t_m}, \psi^l_{t_1:t_m}, \psi^r_{t_1:t_m}, v^{long}_{t_1:t_m}, a^{long}_{t_1:t_m} \rangle. \tag{2}$$

Note that there are no overlaps in $\mathcal{C}$, i.e., no data samples are shared among different members of $\mathcal{C}$.

Let h denote the prediction horizon. Since the overall goal is to predict whether a lane departure will occur within a prediction horizon h prior to the event, the target output signal $\mathcal{T}^c_{t_1:t_m}$ is defined as a time series:

$$\mathcal{T}^c_i = \begin{cases} 1, \text{ if } t_m - h \leq t_i \leq t_m, \\ 0 \text{ otherwise}, \end{cases} \tag{3}$$

[1]The time-span is covering a sequence of m time instances starting $m-1$ samples before the event at time instant t_m.

for $i \in [1, m]$, where $\mathcal{T}_i^c = 1$ if a departure will occur within the horizon and 0 otherwise, see Fig. 2 for an illustration. In this work, the whole event set $\mathcal{C}$ consists of 8276 events. Furthermore, the events are randomly separated into a training set $\mathcal{C}^T \subset \mathcal{C}$ and an evaluation set $\mathcal{C}^E \subset \mathcal{C}$, containing 90% and 10% of the events in $\mathcal{C}$, respectively, and such that $\mathcal{C}^T \cap \mathcal{C}^E = \varnothing$.

For consistency, we want the balance between the positive and negative samples to be constant for all prediction horizons, see Fig. 2. The length m of the time-series is dependent on the prediction horizon h, and was chosen in such way that the ratio $hf_s/m = r$ holds, where $f_s = 40$ Hz is the sample frequency. The ratio r determines how many negative samples should be included prior to an event. This implies that a higher number of negative samples yields a better possibility to learn normal, non-event driving while fewer negative samples yields better training for lane departure cases (events). In this work we use $r = 0.25$, which empirically proved to yield good results. This value is also similar to the ratio within the interval $[0.25, 0.31]$ used in [7]. This implies $m = \{80, 120, 160, 200\}$ samples for $h = \{0.5, 0.75, 1, 1.25\}$s, respectively. Hence, using $m = 200$ samples, the corresponding driving time for $\mathcal{C}$ is approximately 12 hours.

2) Construction of the non-event set $\mathcal{B}$: The non-event set contains only in-lane driving, i.e., it is free of lane departure events[2]. Structure-wise, the non-event set is identical to the event set, such that:

$$b = \langle \mathcal{U}^b_{t_1:t_n}, \mathcal{T}^b_{t_1:t_n} \rangle, \tag{4}$$

where $b \in \mathcal{B}$, the time series are of length $n = 480$ and the target output $\mathcal{T}_i^b = 0$ for $i \in [1, n]$. In total, the non-event set consists of 199528 members (corresponding to 665 hours of driving), and is thereby many times larger than the event set $\mathcal{C}$.

D. The sliding window technique with sparse sampling

To form input-output data pairs for training and evaluation purposes, a sliding window technique is used[3]. For a given time index i in the time series and a member $\diamond$ of either $\mathcal{C}$ or $\mathcal{B}$, the corresponding data pair is expressed by $\langle \mathcal{U}_i^\diamond, \mathcal{T}_i^\diamond \rangle$. By utilizing this construction with a sliding i over the indices in the time series, e.g., from sample $i = 1$ until $i = m$ in the case of $\diamond \in \mathcal{C}$, one extracts m data pairs per each member $\diamond$.

It is also desirable to make use of sparse historic data, yielding a memory effect, in order to limit the measurement noise influence and to capture trends over time. Let us define an offset set:

$$\Gamma = \{\gamma_1, \gamma_2, \dots, \gamma_l\}, \tag{5}$$

[2]Note that even though there are no events present in the non-event set $\mathcal{B}$, there might still be situations where the vehicle is very close to departing the lane, which are in general challenging situations for any type of threat assessment method and can lead to false lane departure detections.

[3]The sliding technique is used for both the $\mathcal{C}$ and $\mathcal{B}$ set.

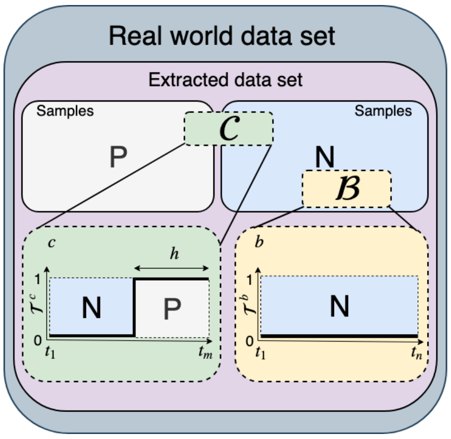


Fig. 2. A subset of the real world data set, called the *extracted data set* is derived by applying the criteria in Section II-B. P represents the set of samples where $\mathcal{T}^\diamond = 1$ ($\diamond$ being a substitute for $b \in \mathcal{B}$ or $c \in \mathcal{C}$) and the N the set where $\mathcal{T}^\diamond = 0$. An element $c \in \mathcal{C}$ consists of time series where the first part, in chronological order, is taken from the N set and the later part from the P set. A member $b \in \mathcal{B}$ consist of time series, where all samples are taken from the N set. Here, m and h represent the time series length and prediction horizon, respectively.

where γ corresponds to an offset expressed in number of samples. Furthermore, an input pattern is introduced as:

$$p_i^\Gamma = \{t_{i-\gamma_1}, t_{i-\gamma_2}, \dots, t_{i-\gamma_l}\}, \tag{6}$$

defining the historic time instances, sampled relative to the current time instance t_i, see Fig. 3 for an illustration. The offsets can be chosen freely[4] as long as $t_{i-\gamma_k} < t_i$ for $k \in [1, \dots, l]$ holds. Using the input pattern in (6), pairs are formed as:

$$\langle \mathcal{U}^\diamond_{p_i^\Gamma}, \mathcal{T}_i^\diamond \rangle. \tag{7}$$

The offset sets used in this paper are taken as follows: $\Gamma_1 = \{0, 1, 2\}$, $\Gamma_2 = \{0, 1, 14\}$, $\Gamma_3 = \{0, 7, 14\}$ and $\Gamma_4 = \{0, 2, 9, 14\}$. Offset set Γ_1 uses the three most recent samples as an input, which should give the network an averaging capability, if needed. In Γ_2, the same principle is applied but using only the two latest samples together with the 14^{th} older sample as an input. The idea is that the last sample should help finding the trend of the trajectory. The offset set Γ_3 is specialized for finding the trend with evenly but sparsely sampled inputs. In Γ_4 we used two recent samples and two older samples.

Note that the aforementioned sliding window technique implies that each member from $\mathcal{C}$ and $\mathcal{B}$ generates $(m - \max(\gamma_k))$ and $(n - \max(\gamma_k))$ pairs of input-output data, respectively, where $k \in [1, l]$.

III. Threat assessment

This section provides an introduction to the principles behind the threat assessment methods considered in this

[4]The definition in (6) allows for sparse sampling of the historic data. Thus, old samples are not even forced to be considered in a consecutive temporal order, which gives the designer the freedom to choose any desired input pattern.

paper. The methods have been implemented in Python using the frameworks Keras [11] and Tensor-Flow [12].

A. The neural network approach

Since the expected outcome is a binary classification, the problem can be interpreted as a logistic regression problem. We rely on the hypothesis that the relationship between the input and output signals might be non-linear. Hence, we choose to use a fully connected Multi-Layer Perceptron (MLP) method due to its well known ability to learn non-linear relationships in the data. It should be emphasized that other ML based methods such as, e.g., SVM, logistic regression or random forests could lead to similar performances to the ones reported in this paper. However, an analysis and comparison of different ML approaches is outside the scope of this work and should be considered in future research.

Let the MLP function be denoted by $f_{h,\Gamma}$ such that:

$$\hat{\mathcal{T}}_i = f_{h,\Gamma}(\mathcal{U}_{p_i^\Gamma}), \tag{8}$$

where i is the time index, $\hat{\mathcal{T}}_i$ the estimated output, Γ the time offset set, p_i^Γ the input pattern for a given c, and h the prediction horizon. Hence, the MLP is trained given a specific combination of h and Γ, referred to in the remainder of the paper as a *MLP configuration*. The MLP consists of four fully connected layers, where the first three layers have 128 neurons, each followed by a single neuron output layer. All layers are using sigmoid activation functions. For more information about neural networks, see e.g. [13]. Note that we have also analyzed, in complement to the content of this paper, the performance with respect to the number of neurons per layer in the range 16 - 512 neurons, but 128 neurons per layer yielded the best performance.

A first-order gradient-based optimizer for stochastic loss functions, called the ADAM optimizer [14], was used. The optimizer is claimed to be well suited for large scale optimization problems with a large amount of training data and/or many parameters. The loss function is chosen as the binary cross-entropy function defined as:

$$L(\hat{\mathcal{T}}_i, \mathcal{T}_i) = -\mathcal{T}_i \log(\hat{\mathcal{T}}_i) + (1 - \mathcal{T}_i) \log(1 - \hat{\mathcal{T}}_i), \tag{9}$$

where $\mathcal{T}_i$ is the target output value and $\hat{\mathcal{T}}_i$ the predicted output value of the network. The binary cross-entropy function penalizes large deviations from the target while perfect predictions yield no penalty. Furthermore, the input signals are normalized with its mean and standard deviation to reduce the impact of large differences in magnitude among the signals, and a drop-out of 30% for each layer is used to prevent over-fitting, see [15].

The MLPs were trained for 200 epochs using the training event set $\mathcal{C}^T$ consisting of 7449 events for a prediction horizon of $h = \{0.5, 0.75, 1, 1.25\}$s using the sliding window technique described in Section II-D. Finally, for decision-making purposes, a manoeuvre is supposed to be triggered or activated if the output of the network $\hat{\mathcal{T}}$ exceeds a given confidence level threshold $\mathcal{T}_{trig}$.

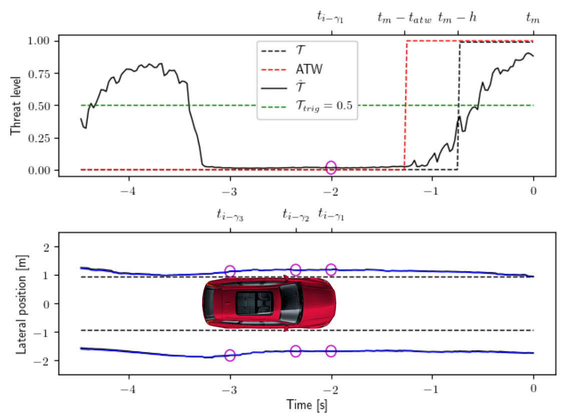


Fig. 3. A lane departure scenario example. The target outcome $\mathcal{T}$ of the MLP method, the Acceptance Time Window (ATW), the triggering confidence level $\mathcal{T}_{trig}$ and the MLP outcome $\hat{\mathcal{T}}$ are illustrated in the upper figure for a prediction horizon $h = 0.75$ s. The lower figure shows the distance to the left and right lane marker and an example of a sparse and uneven sampling pattern (magenta colored circles) of historic data (i.e., prior data) using the sliding window technique described in Section II-D.

B. The kinematic approach

To benchmark the proposed MLP method, a Constant Velocity Model (CVM) is used to estimate the Time-to-Lane Crossing (TLC). To derive the CVM model, let us first compute the lateral velocity v^{lat} given as:

$$v^{(\cdot),lat} = \sin(\psi^{(\cdot)})v^{long}, \tag{10}$$

where the notation $(\cdot)$ means that it can be computed for both left and right side, yielding $v^{l,lat}$ and $v^{r,lat}$ respectively. The estimated Time-to-Lane-Cross ($\widehat{\text{TLC}}$) can then be defined as:

$$\widehat{\text{TLC}}^{(\cdot)} = \frac{d^{(\cdot)}}{v^{(\cdot),lat}}. \tag{11}$$

The CVM approach is explained as follows. The left $\widehat{\text{TLC}}^l$ and right $\widehat{\text{TLC}}^r$ are computed separately, and the minimum $\widehat{\text{TLC}}$ chosen for threat-assessment purposes. Regarding the decision making, an intervention is triggered if the resulting $\widehat{\text{TLC}} \leq T_{trig}$, where $T_{trig} = h$. Note that since the model is parameter-less, no training is required. However, the model has no inherent filtering capabilities, and is thereby sensitive to measurement uncertainties.

IV. Scenario based evaluation

In this section we describe the methodology to assess the performance of the two aforementioned threat assessment approaches, i.e., the proposed MLP method and the CVM approach. We evaluate the precision (i.e., the correctness of the classification) as well as the timing of the decision making, also referred to in the following as the triggering time. The event set $\mathcal{C}^E$ is used to verify the positive performance while the non-event set $\mathcal{B}$ is used to verify the negative performance. The evaluation is performed for every event $c \in \mathcal{C}^E$ and $b \in \mathcal{B}$ respectively, and the outcome of the threat assessment/decision making methods are analyzed in consecutive order, from time instance t_1 until the time instant a triggering decision has been reached, or otherwise until the end of the time series.

For the evaluation of a given element $b \in \mathcal{B}$, if a predicted departure is detected at any time instance the evaluation is stopped and classified as a *False Positive* (FP), and otherwise classified as a *True Negative* (TN). The evaluation of an element $c \in \mathcal{C}^E$ is slightly more complex and is explained as follows. Let us define an Acceptance Time Window $ATW = [t_m - t_{atw}, t_m]$, where t_{atw} is a free parameter which denotes the earliest acceptable time for a triggering, see Fig. 3 for an illustration. Furthermore, let us denote the time instant of the triggering as τ. Now, the outcome can either be classified as a *True Positive* (TP), a *False Negative* (FN) or a FP. A triggering is classified as a TP if $\tau \in ATW$, a FP if $\tau \notin ATW$, and a FN if no triggering occurs.

As the CVM method is widely used within the automotive industry, one can argue that its performance can be considered as a baseline for performance analysis purposes. We assessed the CVM's performance by evaluating the CVM method on every $c \in \mathcal{C}^E$, at the time instant $t = t_m - h$, and form an empirical distribution as shown in Fig. 4, for example. Denote $\widehat{\text{TLC}}^*$ as the maximum argument of the empirical distribution. One can then define the earliest acceptable triggering time for the ATW as $t_{atw} = \kappa|\widehat{\text{TLC}}^*|$, where κ is an tunable, designer-chosen scale factor. A detailed description of the procedure for computing t_{atw} is given in Alg. 1. In this work, we used a $\kappa = 2$ as it seemed to yield an ATW that captures the significant and reasonable part of the estimated TLCs. Computing the ATW for $h = \{0.5, 0.75, 1, 1.25\}$ s yielded a $t_{atw} = \{1.27, 1.48, 1.71, 2.46\}$ s, respectively. Note that, formally, our discussion on this figure entails on setting the event time to $t_m = 0$, which also implies negative time instances for all samples before the event. Even if negative time is a non-realistic concept, this is however just a notation convention introduced for the sake of simplicity of the discussion.

Remark 1: The reader should be aware of the difference between the estimated $\widehat{\text{TLC}}$ computed by the CVM method and the true TLC. The true TLC is defined as $\text{TLC} = t_m - \tau$, while $\widehat{\text{TLC}}$ is an estimate derived according to (11) for $t = \tau$. Close to an event, the two values are expected to be similar, but the correlation might be weak otherwise.

V. Results

In this section we evaluate the performance of the proposed threat assessment method for detecting unintended lane departures using real world driving data. Note that other threatening obstacles such as vulnerable road users and vehicles may be present but are not explicitly taken into consideration by the proposed threat assessment approach, as such topic lies outside of the scope of this work. Our analysis will be articulated around two aspects: (i) the triggering timing and (ii) the relative classification performance.

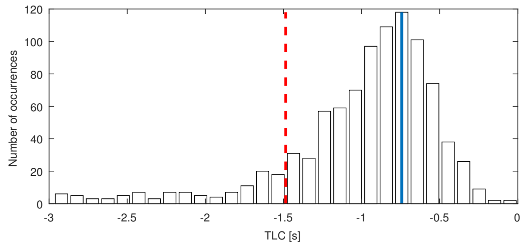


Fig. 4. An empirical distribution function for $h = 0.75$ s used to derive the ATW. The blue solid vertical line indicates the argument maximum $\widehat{\text{TLC}}^*$, while the red dashed line yields the earliest acceptable triggering time $t_m - t_{atw}$, where $t_m = 0$. For convenience, all values above $3s$ have been omitted.

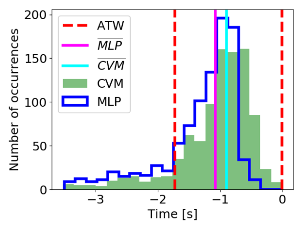
Algorithm 1 The earliest acceptable triggering time t_{atw}

```
$\mathcal{O} \leftarrow \langle \varnothing \rangle,\ t \leftarrow t_m - h$
**for each** $c \in \mathcal{C}^E$ **do**
    Append $\{\min(\frac{d_t^l}{v_t^l}, \frac{d_t^r}{v_t^r})\}$ to $\mathcal{O}$
**end for**
$\widehat{\text{TLC}}^* \leftarrow \text{argmax}(Histogram(\mathcal{O}))$
$t_{atw} \leftarrow 2 \times |\widehat{\text{TLC}}^*|$
```

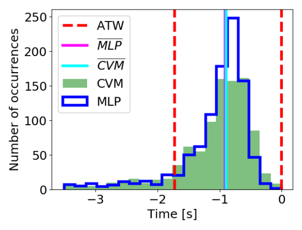
1) Triggering timing: While the decision threshold for the CVM method is set, by construction, to the desired prediction horizon $T_{trig} = h$, there is no obvious way to chose the decision confidence level threshold $\mathcal{T}_{trig}$ for the MLP approach. However, such a choice has a significant impact on the performance, and in particularly on the triggering timing.

Fig. 5 shows the outcomes for four different confidence level thresholds. One can see that the mean triggering time $\overline{MLP}$, computed by averaging over all triggers within the ATW defined before, is decreasing for higher confidence levels $\mathcal{T}_{trig}$. At the same time, the triggerings outside of the ATW, seen at the left of the histograms, are moved into the ATW for high confidence levels. Hence, the TP number increases with higher confidence levels at the same time as the mean triggering time decreases. In other words, the MLP method will eventually take the right decision but it might be too late for a proper corrective manoeuvre. To do a fair comparison between the CVM and MLP methods, in terms of FP and TP performance, the mean triggering time is marginalized by choosing $\mathcal{T}_{trig}$ such that $\overline{CVM} = \overline{MLP}$ holds for each MLP configuration and prediction horizon. Note that the CVM mean triggering time is $\overline{CVM} = \{0.46, 0.67, 0.9, 1.15\}$ s for a prediction horizon $h = \{0.5, 0.75, 1, 1.25\}$ s, respectively.

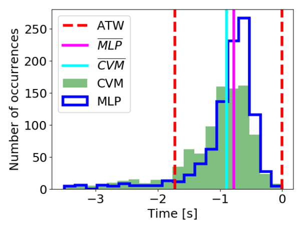
2) Relative classification performance: Before discussing the relative classification performance between the MLP and CVM approaches, it is important to emphasize that the aforementioned methods are implemented without any additional features/adjustments that would robustify the decision making procedure such as a suppress criterion for scenarios where the vehicle drives a long time close to the lane marker or an averaging filter on the decision signal. Even if the individual, absolute performance could be signif-



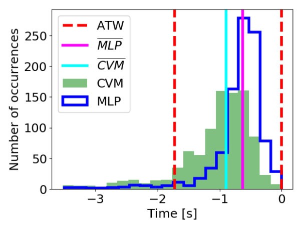
(a) $\mathcal{T}_{trig} = 0.5$



(b) $\mathcal{T}_{trig} = 0.6$

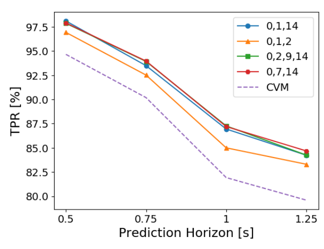


(c) $\mathcal{T}_{trig} = 0.7$

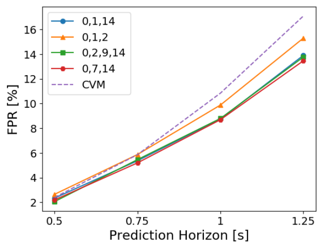


(d) $\mathcal{T}_{trig} = 0.8$

Fig. 5. Triggering timing for the CVM and MLP methods for a prediction horizon of $h = 1$ s. The mean triggering time is computed as the average time for all triggerings within the ATW. The CVM method's mean triggering time $\overline{CVM} = 0.9$ is static since by construction $\mathcal{T}_{trig} = 1$ s. Higher $\mathcal{T}_{trig}$ values yield a higher number of correct decisions but later mean triggering time $\overline{MLP}$ values.



(a) TPR



(b) FPR

Fig. 6. Classification performance for the CVM method and the MLP approach with $\Gamma_1 = \{0, 1, 2\}$, $\Gamma_2 = \{0, 1, 14\}$, $\Gamma_3 = \{0, 7, 14\}$ and $\Gamma_4 = \{0, 2, 9, 14\}$: (a) True Positive Rates (TPRs); (b) False Positive Rates (FPRs).

icantly improved by implementing such features or method adjustments, this lies outside of the scope of this paper and should be considered a part of the industrialization phase. In the remaining of this section, we will therefore solely focus on the relative performance between the CVM and MLP methods as presented earlier in this paper.

The event set is used to derive the True Positive Rate (TPR) for the MLP and the CVM methods. From Fig. 6a one can conclude that all MLP configurations perform better than the CVM, i.e., have higher TPR values with respect to the CVM. The largest improvements are achieved for prediction horizons 1 s and 1.25 s, yielding a relative improvement of 7% and 6%, respectively. The non event set is used to compute the False Positive Rate (FPR), and the corresponding results are depicted in Fig. 6b. One can see that for shorter prediction horizons there is no evident improvement with respect to the CVM method, i.e., the FPR of the MLP method is not significantly lower than the CVM, while for a prediction horizon $h > 1$ s the MLP approach can reduce the false positive rate up to 22%. One can also see that the MLP method using the input offset set $\Gamma_3 = \{0, 7, 14\}$ offers the best performance for longer prediction horizons.

VI. Conclusions

We have shown that a Multi-Layer Perceptron (MLP) can improve, when compared to a Constant Velocity Model, the reliability of lane departure predictions by 7% in terms of true positive rate and almost 22% in terms of the false positive rate. The best results achieved for prediction horizons of 1 s or longer were obtained with sparse historic input data.

Based on the presented results, we pose the following questions for future research:

- Can different network architectures, hyper-parameter optimization and different input signals (e.g., the forward road geometry seen by the front mounted camera), further improve the performance of MLP networks?
- Can alternative machine learning approaches such as SVM or random forests methods outperform a MLP?

References

[1] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, p. 1, 2014.
[2] 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.
[3] S. Mammar, S. Glaser, and M. Netto, "Time to line crossing for lane departure avoidance: A theoretical study and an experimental setting," *IEEE Trans. on Intelligent Transportation Systems*, vol. 7, no. 2, pp. 226–241, 2006.
[4] P. Falcone, M. Ali, and J. Sjöberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Trans. on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.
[5] 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.
[6] P. Angkititrakul, R. Terashima, and T. Wakita, "On the use of stochastic driver behavior model in lane departure warning," *IEEE Trans. on intelligent transportation systems*, vol. 12, no. 1, pp. 174–183, 2011.
[7] J. M. Ambarak, H. Ying, F. Syed, and D. Filev, "A neural network for predicting unintentional lane departures," *IEEE International Conference on Industrial Technology*, 2017.
[8] A. A. Albousefi, H. Ying, D. Filev, F. Syed, K. O. Prakah-Asante, F. Tseng, and H.-H. Yang, "A support vector machine approach to unintentional vehicle lane departure prediction," *IEEE Intelligent Vehicles Symposium*, 2014.
[9] C. Wissing, T. Nattermann, K.-H. Glander, and T. Bertram, "Trajectory prediction for safety critical maneuvers in automated highway driving," *IEEE Conference on Intelligent Transportation Systems*, 2018.
[10] P. Kumar, M. Perrollaz, S. Lefèvre, and C. Laugier, "Learning-based approach for online lane change intention prediction," *IEEE Intelligent Vehicles Symposium*, 2013.
[11] F. Chollet *et al.*, "Keras," https://keras.io, 2015.
[12] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," https://tensorflow.org, 2015.
[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 2015.
[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.