Anomaly Detection for Advanced Driver Assistance Systems Using Online Feature Selection

Conference Paper · June 2019

DOI: 10.1109/IVS.2019.8814175

Christoph Segler
Technische Universität München
12 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Stefan Kugele
Technische Hochschule Ingolstadt
60 PUBLICATIONS **206** CITATIONS
SEE PROFILE

Philipp Obergfell
Karlsruhe Institute of Technology
15 PUBLICATIONS **37** CITATIONS
SEE PROFILE

Mohd Hafeez Osman
Universiti Putra Malaysia
32 PUBLICATIONS **128** CITATIONS
SEE PROFILE

2019 IEEE Intelligent Vehicles Symposium (IV)
Paris, France. June 9-12, 2019

Anomaly Detection for Advanced Driver Assistance Systems Using Online Feature Selection

Christoph Segler[1], Stefan Kugele[2], Philipp Obergfell[1], Mohd Hafeez Osman[2], Sina Shafaei[2], Eric Sax[3], and Alois Knoll[2]

***Abstract*— *Context*: As we move towards higher levels of automation in autonomous driving, we see an increase in functionality that either assists or takes over in both normal and emergency scenarios. These new functionalities can be intentionally switched off by the user, but can also be deactivated unintentionally by (i) accident, (ii) a software malfunction, (iii) a hardware defect, or (iv) an intrusion. *Aim*: In addition to already applied methods at design time, we aim to recognise mistimed and/or unintended deactivation of vehicle functions, in particular, *driver assistance functions* (ADAS), at run-time. Upon recognition of the occurrence, we propose to inform the user and the original equipment manufacturer (OEM) in order to improve both the future and the current system behaviour, to support development processes, and to support already conducted safety measures. *Method*: Based on a feature subset, selected by streaming feature selection, we learn the nominal behaviour of the driver in the interaction with *ADAS functions* in order to find deviations. The approach considers the technical challenges of automotive E/E architectures and is optimised to reduce communication and computational complexity. We evaluate this approach with recorded *real car data* from customers participating in a field study. *Results*: Based on eight datasets, we traced a total of 17 state-of-the-art *ADAS functions* per participant, yielding to a total of 136 runs. We observed that during 24 among them, the user deactivated the functions at least once for more than a few seconds. For 13 of these 24 runs, we were able to detect and flag possible non-nominal behaviour. *Conclusion*: As at least one participant configured a convincingly large number of ADAS functions, we need a dynamic system to monitor the configuration of these functions actively. Our approach was capable of detecting potential non-nominal behaviour in up to 52% (13/24) out of these reconfigurations. This result is promising and will receive further attention in future work.**

I. Introduction

Today, E/E (electric/electronic) architectures are best characterised as historically grown, mostly federated, partly integrated architectures with often pragmatic, cost-efficient, and ad-hoc solutions. Current trends in the automotive industry are introducing new, increasingly complex software functions into vehicles [1]. The ever-growing availability of computing resources, memory, and newest technologies allows for new levels of *automated driving* (i. e., levels 3 to 5 according to SAE J3016 [2]) and *intelligent systems*.

[1]BMW Group Research, New Technologies, Innovations, Garching bei München, Germany {christoph.segler, philipp.obergfell}@bmwgroup.com
[2]Technical University of Munich, Department of Informatics, Garching bei München, Germany {stefan.kugele, hafeez.osman, sina.shafaei}@tum.de, knoll@in.tum.de
[3]Karlsruhe Institute of Technology, Department of Electrical Engineering and Information Technologies, Karlsruhe, Germany eric.sax@kit.edu

TABLE I: Anomaly Model

Possible reasons		**Actions**		
		Vehicle	**Driver**	**OEM**
Vehicle	SW/HW fault	depend. pattern, CC message	safe state	analyse fault
Driver	intentionally, maloperation	CC message, safe state	—	Improve UI
Environment	IT attack	CC message, containment	safe state	analyse security threat

The structure follows the generic concept of endangerment scenarios and mitigation strategies introduced by Gleirscher and Kugele [4].

Automated driving poses the highest functional safety requirements (i. e., ASIL D according to ISO 26262 [3]), ranging from partial automation, which is available already today, to the highest level of full automation (i. e. robot vehicle at level 5). To cater to these needs, a multitude of new customer functions and hardware devices will be added.

Intelligent systems come along with connectivity (e. g. car-to-car and car-to-infrastructure) that increases the attack surface. Currently, connectivity is mostly used for customer convenience functions such as calendar and e-mail synchronisation or real-time traffic information; However, with the introduction of higher levels of automation, it will also be used as part of automation functions, e.g. highly precise and accurate real-time map data are needed.

In this work we consider the *deactivation* of *driver assistance functions* such as side impact warning, lane-change warning, and cross traffic alert. A driver can deactivate such functions manually (*personalisation*). A deactivation per se is not a problem if intended by the driver: for instance, a driver might deactivate traction control when being stuck in an iced parking lot; However, if such functions have been deactivated due to a software or hardware fault, or even after an IT attack by an intruder, this is a severe problem that we try to identify and to mitigate. In addition to already applied methods at design time, we need to learn a *personalised driver model* that contains information about the situations (aka *context* or *operational situation*) in which drivers intentionally deactivate functions. We refer to this as *nominal behaviour*. We evaluate the approach with recorded traces (cf. Section III) of real customer vehicles.

Based on *nominal behaviour*, we consider several reasons for anomalies or deviations from this behaviour. Table I depicts an overview of the considered reasons of a function deactivation and possible actions as countermeasures taken by the involved parties *vehicle*, *driver*, and the *OEM* (Original Equipment Manufacturer, i. e., the car maker). For all areas of deactivation, i. e., the vehicle itself (e. g. software and hardware faults), the driver (e. g. intentionally or by maloperation), or the environment (e. g. IT attack, intruder) the driver is informed about the potential anomaly by showing a *Check Control message* (CC message). Thus, the driver is informed about the *anomalous behaviour* and can confirm it as intended or as not intended, yielding a more precise personalised driver model. By receiving this notification, the driver can then mitigate the potentially hazardous driving situation by stopping the car and bringing it into a safe state.

Technical Challenges: Today, ECU- (electronic control unit) and signal-based development approaches are the predominant ones in systems engineering and have reached their limits of mastering complexity. Data maintenance of signal and message databases requires a big effort. However, as long as we have not reached a fully service-oriented architecture (SOA) (cf. e. g. [5]) on system level allowing to analyse and understand complex feature interactions in functional networks, we have to rely on messages and signals.

The presented approach heavily relies on data collected at run-time which raises multiple challenges since current E/E architectures are not designed for data analytics: (i) Due to their federated architecture, data is distributed over multiple ECUs and cannot be retrieved at a central point. (ii) Sensors, functions, and ECUs generate a huge amount of data which currently ranges from 2-3 GiB of pre-processed bus data per hour (video and radar data not included). Considering such amounts, it is not feasible and cost-efficient to store all the data locally in the car or transfer it to a backend infrastructure. (iii) Current vehicles generate around 4000 signals that are useful for data analytics or machine learning use cases. This high amount leads to the "curse of dimensionality" [6] in data analytics and machine learning. The "curse" describes the problem of having a too large amount of data that may contain irrelevant, redundant, or not suitable data yielding a reduced accuracy and training efficiency. This effect can be minimised by applying *feature selection* algorithms (cf. Section III-B). (iv) Moreover, also the heterogeneous structure of the collected data itself is challenging. In vehicle networks, data is being sent asynchronously with a variety of sampling rates, leading to the challenge of synchronising the data or evaluate each signal independently. (v) Moreover, the variety of data types that car signals are mapped to, is challenging. Signals can be discrete, continuous, or even both at the same time. This reduces the number of possible algorithms and raises the need for a high amount of pre-processing on the data.

Research Questions: For this work, we pose the following research questions:

RQ1 How to provide proactive safety management for customer functions in automotive systems at run-time?
RQ2 How to detect safety violations for functions, which are allowed to be disabled by the user?
RQ3 How to gather run-time data for an improved software engineering process?

Contributions: This paper provides the following scientific contributions, but also contributes to the state of practice:

(i) Feature selection for a run-time monitoring concept of customer vehicle functions;
(ii) Accessing anomalous system states with learned user behaviour;
(iii) Feedback loop from user to systems engineering.

Outline: The remainder of this paper is structured as follows. Section II summarises related work followed by the primary approach of this work in Section III. Section IV presents the evaluation with the conducted experiments and gained results followed by their discussion in Section V. Finally, we conclude the paper in Section VI.

II. Related Work

Although researchers were concerned about the security of automotive networks since their implementation, Miller and Valasek [7] were the first to raise attention about critical vulnerabilities inside the car. In their work they presented exploits causing severe safety problems, e. g. they were able to engage the breaks or control the lights of cars. Hence, they emphasise the need for additional security measures.

Müter et al. [8] presented an attack recognition scheme for in-vehicle networks. The scheme is composed concerning typical characteristics of automotive networks such as Controller Area Network (CAN) which includes eight attack detection sensors. These pre-defined sensors serve as recognition criteria for automotive threats. Then, Müter and Asaj [9] proposed an entropy-based anomaly detection approach. From our point of view, using pre-defined sensors for recognition will limit the capability of attack detection. On the contrary, we learn the users' nominal behaviour to detect anomalies of safety and security-related issues resulting in a more generic approach.

In the aviation domain, Das et al. [10] proposed multiple kernels learning to detect potential safety anomalies in a dataset gathered from operations of commercial fleets. They focused on four types of (human interaction) faults or anomalies that are mostly related to missing switches in a sequence, wrong sequence of switches, and several (known) faults. They showed that their approach is better compared to two earlier approaches: Orca [11] and SequenceMiner [12]. Das et al. [13] extended this work by investigating the system level anomaly of the same system domain. We focus on single control sequences such as turning off safety functions (e. g., traction control) that are arguably easier to be accessed and attacked by the user and the environment.

Baldoni et al. [14] utilised the network traffic data from the monitored systems to construct a failure prediction mechanism. They introduced failure prediction architecture (called CASPER) purposely to identify the abnormal behaviour of defined performance metrics. CASPER investigates the failure symptoms by using the combination of Hidden Markov models and Complex Event Processing. While CASPER focused on the *black box* and *non-intrusive* failure detection, our solution is more focused on the state of a vehicle function which may provide more information about safety violation.

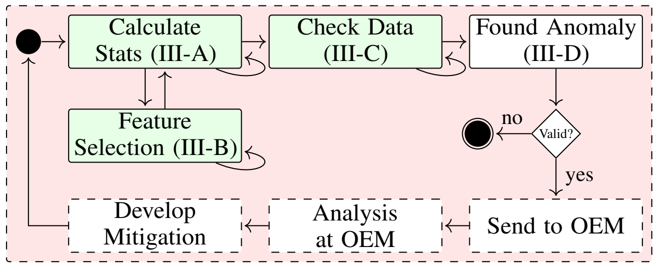

Fig. 1: Process: ↻-steps are repeatedly done at run-time.

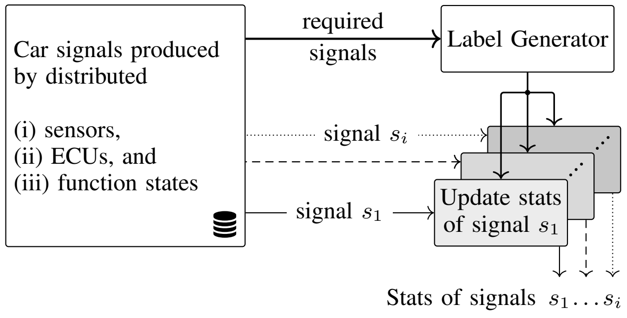

Fig. 2: Statistic calculation of each signal.

The approach by Taylor et al. [15] is explicitly designed to detect anomalies on CAN buses. The authors present the limits of their first approach—measuring inter-packet timings over a sliding window—and report on the successful use of one-class support vector machines. In contrast, our approach is far more technology-agnostic by abstracting from communication protocols and does not only detect intruded CAN messages.

All previously discussed approaches only focus on the low-level anomalies directly visible inside the car. However, with the high level of interconnection, this is not sufficient for detecting all types of attacks. For example, the exploit found by Hunt [16] allowed adversaries to remotely set the climate control due to a missing authentication in the manufacture's Web API used by its smartphone app. From the car's perspective, the malicious requests looked perfectly valid, although the driver would never set the temperature to a level, where s/he is freezing or sweating. Luckily, the safety impact of that vulnerability was minimal, but we expect more and more features to be remotely available in the future.

III. Approach

This work is a continuation and significant refinement and extension of the previous work [17] in which we showed that it is possible to learn the context of a car function when being used. We used this information for a simple anomaly detection approach; However, this proof-of-concept has only been evaluated for one test vehicle. Fig. 1 depicts the general process of the refined approach. Each of the process steps is described in the following Sections III-A to III-D.

A. Calculation of Statistics

In the first step of our approach, the statistical distribution of all car signals is calculated. Let s_i with $1 \leq i \leq m$, $m \in \mathbb{N}^+$ be a signal and m the number of signals. Moreover, let j, $1 \leq j \leq n$, $n \in \mathbb{N}^+$ be a class and n be the number of possible classes. A class j represents the function's state which can have n possible states (e. g. the traction control can be switched on or off). With $\overline{s_i}$ we denote the current value of signal s_i. The fundamental idea of Algorithm 1 is to *continuously update* (as long as new signal observations are received, i. e., $\overline{s_i} \neq \perp$) two $m \times n$ matrices $\boldsymbol{\mu} = (\mu_{ij}) \in \mathbb{R}^{m \times n}$ (for the mean) and $\boldsymbol{S} = (S_{ij}) \in \mathbb{R}^{m \times n}$ (interim value for the standard deviation) (cf. Fig. 2). Hence, we are able to assess each signal s_i in each class j. With o_{ij} we denote the number of observations of signal s_i in class j. The calculation is based on the semi-numerical calculation of Knuth [18] and Welford [19]. With (1) we are able to calculate the mean of signal s_i independent of the class j and with (2) the standard deviation for each signal s_i in class j.

$$\mu_i = \frac{\sum_{j=1}^{n} \mu_{ij} o_{ij}}{\sum_{j=1}^{n} o_{ij}} \quad (1) \qquad \sigma_{ij} = \sqrt{S_{ij} / (o_{ij} - 1)} \quad (2)$$

As a first step, the label for the corresponding function is generated. In this case, the label holds the information of the current status of the function for which we want to detect anomalies (class j) and is generated based on status signals from the function itself. For all m signals that are accessible in the vehicle's architecture, we use the current value of the signal $\overline{s_i}$ and the generated class j as input for the algorithm.

Since the calculation is independent for each signal, we do not have to consider the synchronisation of all signals and there is no overhead in sending the data to a central place. This can lead to a suboptimal feature subset because we do not consider the relationship of signals to each other. However, we accept this drawback due to the posed challenges (cf. Section I). Furthermore, in the calculation itself, there is only a single dependency to the last state of μ_{ij}, S_{ij}, and o_{ij} and no additional data of previous states have to be stored.

B. Feature Selection

To resolve the "curse of dimensionality" [6] (see Section I) our model selects a subset of signals showing the best correlations to the label (here: state of the *driver assistance function*). This is achieved by using the supervised feature

Algorithm 1: Statistics calculation for each signal s_i.

```
Data: Value of signal $s_i$ denoted by $\overline{s_i}$ and label $j$
Result: Statistics matrices $\boldsymbol{\mu}$ and $\boldsymbol{S}$
$o_{ij} \leftarrow 0$;  $\forall 1 \leq i \leq m$ and $\forall 1 \leq j \leq n$;
while ($\overline{s_i} \neq \perp$) do
    if ($o_{ij} = 0$) then
        $\mu_{ij} \leftarrow \overline{s_i}$; $S_{ij} \leftarrow 0$; $o_{ij} \leftarrow 1$;
    else
        $\mu_{ij} \leftarrow \mu_{ij} \oplus (\overline{s_i} \ominus \mu_{ij}) \oslash o_{ij}$;
        $S_{ij} \leftarrow S_{ij} (\overline{s_i} \ominus \mu_{ij}) \otimes (\overline{s_i} \ominus \mu_{ij})$;
        $o_{ij} \leftarrow o_{ij} + 1$;
```

selection algorithm *Fisher score* [20]:

$$\mathit{fisherscore}(s_i) = \frac{\sum_{j=1}^{n} o_{ij}(\mu_{ij} - \mu_i)^2}{\sum_{j=1}^{n} o_{ij}\sigma_{ij}^2} \quad (3)$$

This algorithm calculates a score for each feature[1] s_i. This calculation is based on the mean μ_i and standard deviation σ_i of each signal s_i, and the mean μ_{ij}, standard deviation σ_{ij}, and number observations o_{ij} of each signal s_{ij} in class j. All these values have already been calculated in the previous step (cf. Section III-A) and can be directly used to calculate the score. This score is then collected in a central place to rank the features according to their correlation to the label. This is the only step which is conducted in a non distributed manner but will only lead to a minimal amount of communication due to the small number of parameters. It is also conceivable to do this ranking distributed in the architecture by comparing feature by feature. Only the highest correlating (i. e., ranked) features are used for anomaly detection in the next step.

The advantage of this scoring method is that it (i) uses data that was calculated from data streams and (ii) can be applied for discrete and continuous data (cf. Section I). Another advantage—which can also be considered a drawback—is the evaluation of each feature/signal individually. While performing feature selection and evaluating each feature on its own, the correlation between features cannot be considered in the collection, leading to a suboptimal feature subset. Referring to the challenges stated in Section I, this drawback is accepted in favour of resolving the issues of the highly distributed architecture and highly heterogeneous data. Hence, all features can be evaluated in a distributed manner and a minimal overhead in communication is achieved.

We use the top k features out of all ranked m signals as the basis for the next steps. For each use case, the threshold k has to be set as a trade-off between accuracy and computational complexity. The higher the k, the more possible features can describe the context for a particular state, but could also lead to the before mentioned "curse of dimensionality" and increased calculation time in the next step.

C. Checking New Data for Anomalies

For detecting anomalies, our approach is based on finding outliers for each of the previously selected top k features. An outlier is defined as "patterns in data that do not conform to a well-defined notion of normal behaviour" [21]. Anomaly detection is a process to find these outliers in data by comparing with some predefined patterns or rules.

Our approach is based on the Grubbs' test [22] (4). In this test, the current value $\overline{s_i}$ of each signal s_i in the currently observed class j is compared to the previously calculated mean μ_{ij} and standard deviation σ_{ij}. The current observation is considered as anomalous if the value z_i exceeds a threshold. This threshold is calculated using the number of observations o_{ij} of the signal s_i and the value of the t-distribution with a significance level of $\alpha/(2o_{ij})$ and a degree of freedom of $o_{ij} - 2$ referred to as $t_{\alpha/(2o_{ij}),o_{ij}-2}$. The value of α indirectly influences the sensitivity of this anomaly detection and has to be chosen function-specific.

$$\frac{|\overline{s_i} - \mu_{ij}|}{\sigma_{ij}} = z_i > \frac{o_{ij}}{\sqrt{o_{ij}}} \sqrt{\frac{t^2_{\alpha/(2o_{ij}),o_{ij}-2}}{o_{ij} - 2 + t^2_{\alpha/(2o_{ij}),o_{ij}-2}}} \quad (4)$$

For every feature in the selected subset, the calculation is done for each new observation of the signal. Since this anomaly detection is signal-independent, it can be distributed in the architecture and calculated directly at the source of the signal.

D. Anomaly Feedback-Loop

After detecting an anomaly for some signals out of the feature set, which is above a specific threshold, the correctness of the detection must be validated by the driver. The finding is presented as CC message on the head-unit with a warning explaining the target function and its possible abnormal behaviour. The driver is expected to confirm whether the detection of the anomalous function was correct or the driver changed the state of the function by her/himself. If the detected anomaly gets validated by the driver, then it will be sent to the OEM for further analysis and investigation.

IV. Evaluation and Results

We used already collected data from a field study conducted at the BMW Group to evaluate our approach. This has the advantage of being able to re-run our approach on the same data and evaluate the results for each run. The implementation does not differ from an implementation in a real car.

A. Data Collection

TABLE II: Datasets

Dataset	Samples	Features	Size [GiB]
1	169507	3836	4.9
2	863143	3931	25.0
3	720489	3772	19.9
4	850606	3906	24.3
5	287115	3832	8.1
6	208141	3780	5.8
7	671113	3820	18.7
8	226818	3910	6.6

The car data has been collected from eight current generation BMW 7 Series (G11), which have been equipped with hardware loggers, collecting all messages which are sent over FlexRay, CAN, and Ethernet buses (so-called traces). Messages appear in the same order as they were sent.

Car signals were extracted with a sampling rate of one second from these messages. Except for the sampling rate, this data is identical to the data which would be used, when running our approach directly in the car. Each dataset contains ≈3.800 car signals.[2] Based on the time the car was driven, the total number of samples varies from ≈170.000 to ≈850.000 samples. An overview is given in Table II.

As an evaluation platform, we used an NVIDIA Digits DevBox (Core i7-5930K, 64GB RAM). By using a single core of the CPU, the calculation took ≈450s per function and hour of trace.[3]

[1]In data analytics or machine learning, information that can be used as input for an algorithm is referred to as a *feature*. In this work, we consider the pre-processed vehicle signals as features.

[2]Depending on the car's options and usage, the number of signals varies.

[3]We also ran our approach directly in a test vehicle on a dedicated car-pc (Intel Xeon E3-1268, 32GB RAM) at run-time. However, we did not use this method due to missing data that can be evaluated.

TABLE III: Selected driver assistance functions and number of datasets with a certain time-ratio in function reconfiguration

Functions	**Possible States**	**Reconf.** $< 1\%$	**Reconf.** $\geq 1\%$
Front-end Collision Warning	{on, off}	8	-
Active PDC Emergency Intervention	{on, off}	1	7
Cross-Traffic Alert	{on, off}	8	-
Lane Departure Warning (LDW)	{on, off}	7	1
LDW Steering Intervention	{on, off}	7	1
LDW Sensitivity	{always, reduced}	7	1
Lane Change Warning (LCW)	{on, off}	7	1
LCW Steering Intervention	{on, off}	7	1
LCW Sensitivity	{early, med., late}	6	2
Side Collision Warning	{on, off}	8	-
Dynamic Marker Light	{on, off}	8	-
Speed Limit Assist	{on, off}	8	-
Speed Limit Assist Offset	[-15;15] [km/h]	7	1
Steering Wheel Vibration Strength	{light, med., strong}	6	2
Drive Mode Sport	{on, off}	1	7
Driving Stability Control DSC	{on, off}	8	-
Dynamic Traction Control DTC	{on, off}	8	-
	Total	**112**	**24**

Table III lists all *driver assistance functions* which we use in this evaluation with their names and possible states.

B. Results

1) Inclusion/Exclusion Criteria: The experimental setting requires a function that changed its state at least once in a run in order to have at least two classes for the feature selection. Moreover, we exclude runs in which the function is in the same state for over 99% of the run and vice versa less than 1% in different states. The reason is, that we observed, that the feature selection algorithm is not accurate enough in this case and will not provide reasonable results. In total, we evaluated our approach with 24 test runs (cf. Table III).

2) Parameters: For this first evaluation of our approach we set the parameters as follows: For anomaly detection, we consider the top $k = 30$ features and set a threshold of $\alpha = 0.1$ for the t-distribution in the Grubbs' test. For further investigation, different parameters can be evaluated and depending on the criticality of the observed function the parameters should be set accordingly. As depicted in Table III, *driver assistance functions* are deactivated very rarely, but for almost all functions there is at least one dataset, in which the functions were deactivated/reconfigured at least once for more than 1% of the time. Only the function *Active PDC Emergency Intervention* and the *Drive Mode Sport* were changed in almost every dataset.

3) Metrics: We use two different metrics to evaluate the runs: For the evaluation of the feature selection, we use the *Jaccard distance* [23] of the top $k = 30$ features between consecutive time steps of 300s. This allows visualising the difference in the feature sets between two observed time steps. The k-sized feature sets are fundamental for the anomaly detection approach. The optimal curve for this metric should be a high *Jaccard distance* for the initialisation at the beginning and it should subside—meaning no more changes in the feature subset.

For the evaluation of the anomaly detection, we consider all recorded actions in the trace as nominal behaviour of the user. To describe the anomaly detection, we use the ratio of features for which an anomaly is detected. In this ratio, we only consider features within the k-sized feature set. This ratio is calculated for two classes: (1) Once for the currently active class of the function in the trace. This ratio corresponds to the *false positives* of the anomaly detection and represents a detected anomaly in the recorded trace, which we consider as nominal behaviour. This ratio should always be as low as possible. (2) The second ratio considers the opposite state of the function recorded in the trace. This state was not recorded in the trace and we consider this hypothetical change in the function's state as not intended by the user and we refer to this as an anomaly, due to not being present in the real trace. This ratio corresponds to the *true positive* rate in the anomaly detection and should always be as high as possible. The *true negative* and *false negative* ration can be calculated with the other two values and is not mentioned, due to simplicity.

4) Results: For the analysis of our results, we classify the performance of the anomaly detection into six patterns:

P1 For the first changes in the state of the function, no false positives are detected. After these changes, the true positive rate increases and, a not by the user triggered change in the function's state, will lead to detected anomalies. (4/24 runs, cf. Fig. 3a);

P2 With the first changes in the function's state, a false positive is detected. This false positive will subside and we observe a high true positive rate. In this case, the user is informed at the first change of the setting. Afterwards, a change which is not triggered by the user will be detected as an anomaly. (3/24 runs, cf. Fig. 3b);

P3 With a change in the function's state, a false positive is detected. It will subside and no false positives or true positives will be detected any more. In this case, the user is informed at the first change and no anomalies will be detected afterwards. (6/24 runs, cf. Fig. 3c);

P4 In this pattern, the function is in a specific state for a longer duration than in the other states and each time the function is in the less observed state, we observe a high false positive rate and a low true positive rate. Vice versa in the more observed state, a low false positive rate and a high true positive rate are observed. This leads to a message for the user, each time s/he is changing to the less used state of the function. (6/24 runs, cf. Fig. 3d);

P5 No true positives or false negatives were observed. In this case, the user will not receive any message and no anomalies can be detected. (4/24 runs, cf. Fig. 3e);

P6 True positives and false negatives were observed almost at all times. In this case, the user will receive a notification; regardless if intended or not. (1/24 runs, cf. Fig. 3f).

A summary of all runs and the assignment to the patterns is given in Table IV. An excerpt of six runs is depicted in Fig. 3.

In Fig. 3a an exemplary result for **P1** is shown. As depicted in the upper plot the *Lane Departure Warning Sensitivity* is changed by the user in the second half of the trace. Right after this change, the feature subset initialises, due to the first observation of another state. With the initialisation of the

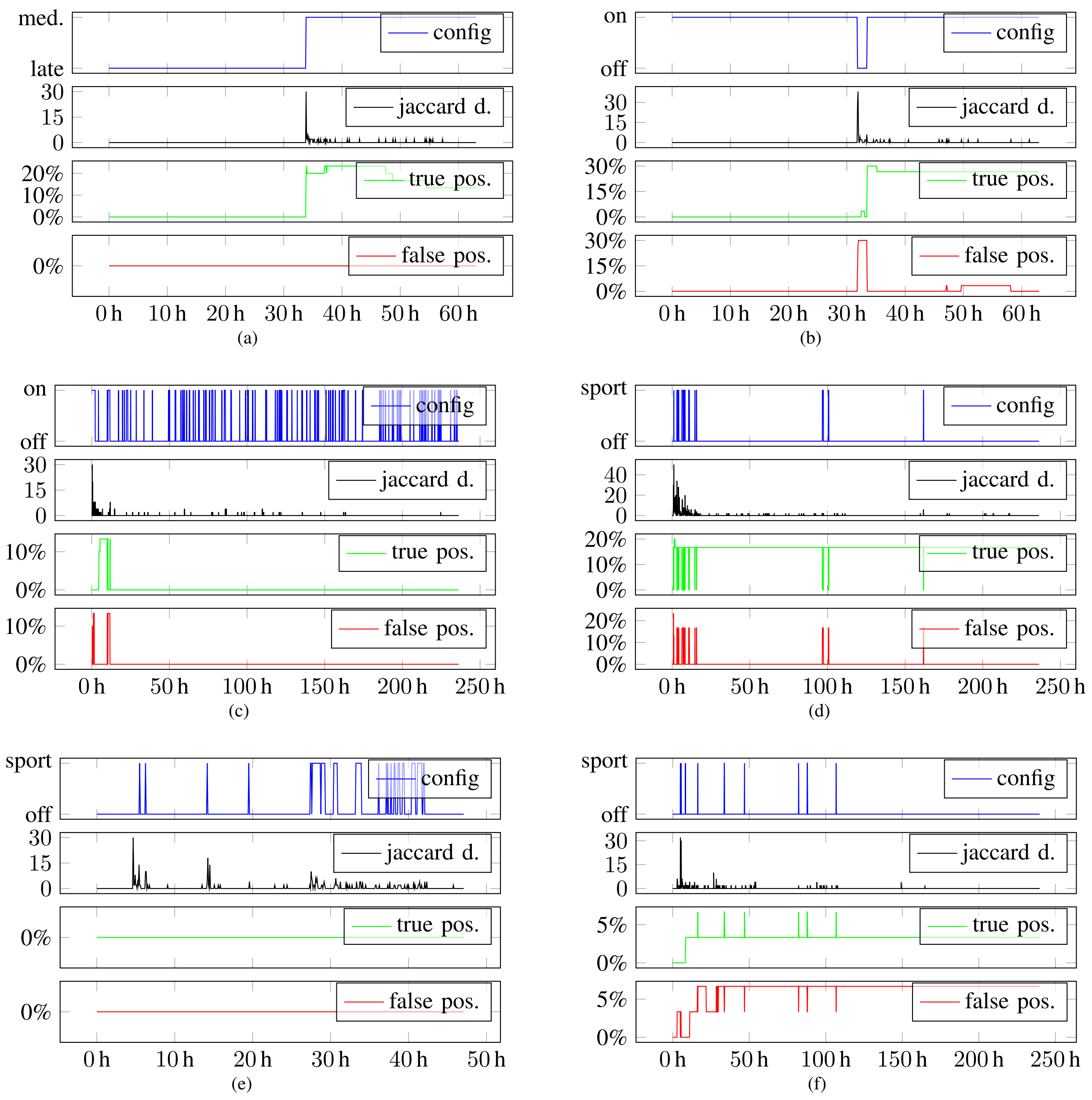

Fig. 3: (a) *Lane Departure Warning Sensitivity* for Dataset 8 in pattern **P1**, (b) *Lane Departure Warning* Dataset 8 in pattern **P2**, (c) *Active PDC* for Dataset 4 in pattern **P3**, (d) *Drive Mode Sport* for Dataset 4 in pattern **P4**, (e) *Drive Mode Sport* for Dataset 1 in pattern **P5**, and (f) *Drive Mode Sport* for Dataset 2 in pattern **P6**.

feature subset, true positives are detected for ≈20% of the features. Within the full trace, no false positives are detected. An example of **P2** is depicted in Fig. 3b. After the first change in the configuration, the feature subset initialises. This leads to a detected anomaly for this trace and to a message to the user (false positive rate). After a short time, the user reconfigures the function. The false positive rate drops to 0%. The true positive rate increases and anomalous changes in the configuration would trigger anomalies. Fig. 3c show an example for **P3**. In this case, the function *Active PDC* is configured multiple times between on and off. In this case, the first change leads to a detected anomaly. After switching back to off, potential anomalies would be detected. Right after switching back to on, another anomaly is triggered and no more anomalies (true & false positives) are detected within the rest of the trace. In Fig. 3d an example for **P4** is shown. In this case, every change to the less observed state (in this case *sport*) triggers anomalies and a message to the user. An example for **P5** can be found in Fig. 3e. In this case, no potential anomalies are detected. We can observe a frequent change in the feature subset each time the configuration changes. The only trace in **P6** is shown in Fig. 3f. In this case, the true positive and false positive rates almost never change throughout the trace.

For the analysis of the feature subset, we have analysed the subsets at the end of each run. To classify the features in the subset, we use the following categorisation:

Fire (🔥) Features with a functional dependency on the class. For instance in the case of the function *Drive Mode Sport*, a change of the drive mode will “fire” features for the configuration of the drive train or engine;

Safety-config (♥) Features that contain parameters of other *driver assistance functions* (cf. Table III). For instance if the state of the *Steering Intervention Setting* correlates to the setting for the *sensitivity* of another *driver assistance functions*.

Personalisation (👤) Features for the personalisation of comfort functions. E. g. temperature setting for the climate control, the fragrance of the ambient air perfuming as part of the car’s climate function or seat position;

Context (🗺) Features which correlate to a specific driving context. E. g. the current road type, the speed limit, child safety lock for windows, or outside temperature;

Time (🕘) Features that change over time, but do not have any reasonable relationship to the *driver assistance functions*. E. g. the fuel level;

Non-related (?) Features where we cannot find any reasonable relationship to the *driver assistance functions*.

An overview of the result can be found in table IV. We observed the following similarities in the feature sets over all runs: (i) In all traces, we can observe a peak in the change of the feature sets when a new state of the function is observed for the first time (initialisation of feature set); (ii) In all traces, the feature subset has a high variance after the first observation of a new class and stabilises over time; (iii) In almost all traces a peak in the Jaccard distance can be observed shortly after a change in the function’s state.

Comparing the feature subsets and the patterns found in the anomaly detection, we could not find any relationships (see table IV). Moreover, when looking at the features itself, they highly vary between each car and each function. One exception is the feature subset for the function *Drive Mode Sport*. Over all cars, the subset contains a large number of *fire features*.

To sum up, function changes that were not triggered by users lead to a detected anomaly in 13/24 runs (**P1**, **P2**, and **P4**). In 16/24 runs, the user was notified at least once when s/he changed the setting of the function (**P2**, **P3**, **P4**, and **P6**), and in 11 runs no anomalies were detected if the change would have not been triggered by the user (**P3**, **P5**, and **P6**).

V. Discussion

We conclude from the first results of the presented case study that the described approach works well—at least in the described setting. It is rare that drivers configure *driver assistance functions*, but this happens, hence personalisation of them is needed and we expect more functions of that type to be introduced in future highly and fully autonomous cars.

Drivers are and behave diverse, i. e., the selected feature subsets vary from driver to driver, meaning that a one-fits-all solution does not work in modelling the drivers’ behaviour.

TABLE IV: Features in feature subset at the end of each run.

	Function Name	Dataset	🔥	♥	👤	🗺	🕘	?
P1	*Lane Departure Warning Sens.*	Dataset 8	3	4	5	7	7	4
	Lane Departure Warning St. Interv.	Dataset 8	3	15	8	2	1	1
	Steering Wheel Vibration Strength	Dataset 2	3	2	12	4	7	2
	Steering Wheel Vibration Strength	Dataset 8	3	6	4	4	8	5
P2	*Active PDC*	Dataset 7	2	-	7	13	6	2
	Lane Change Warning	Dataset 8	4	15	1	5	-	5
	Lane Departure Warning	Dataset 8	4	16	7	-	-	3
P3	*Active PDC*	Dataset 4	6	5	14	2	-	3
	Active PDC	Dataset 6	2	1	18	2	3	4
	Drive Mode Sport	Dataset 6	16	-	4	2	1	7
	Lane Change Warning Sens.	Dataset 4	3	6	14	2	1	4
	Lane Change Warning Sens.	Dataset 8	3	12	5	6	3	1
	Speed Limit Assist Offset	Dataset 8	3	13	5	4	4	1
P4	*Active PDC*	Dataset 8	2	7	14	-	5	2
	Drive Mode Sport	Dataset 4	16	-	7	4	1	2
	Drive Mode Sport	Dataset 5	15	-	5	5	-	5
	Drive Mode Sport	Dataset 7	14	-	7	5	2	2
	Drive Mode Sport	Dataset 8	19	-	5	5	-	1
	Lane Change Warning St. Interv.	Dataset 8	3	9	6	3	7	2
P5	*Active PDC*	Dataset 1	1	-	-	26	-	3
	Active PDC	Dataset 3	2	-	9	8	3	8
	Active PDC	Dataset 5	3	1	-	22	-	4
	Drive Mode Sport	Dataset 1	16	1	1	11	-	1
P6	*Drive Mode Sport*	Dataset 2	16	-	3	8	2	1

This demands a learning approach and feature selection at runtime. Moreover, we learned that feature sets tend to stabilise quite quickly and therefore long traces are not needed and will not improve the feature set.

Our proposed approach did not work well for all runs in the evaluation. We have identified three possible reasons for this result: (1) As mentioned in the introduction, the purely signal-based approach has limitations. We consider that the user’s behaviour can be modelled by already existing car signals. Depending on the complexity of the behaviour this will not always be the case. Future vehicle generations tend to have more and more sensors build in, which could be a better basis for the modelling of the user’s behaviour and a solution to this issue. (2) During feature selection, a suboptimal feature set could be selected, which is not able to fully describe the user’s behaviour. More complex algorithms which also include the correlations between features can be a solution, but come at the cost of adding communication overhead to the architecture and might not be able to run on streaming data (cf. Section I). (3) Based on this subset we model the user’s behaviour for each feature on its own. Linking multiple features could improve accuracy. Due to the aforementioned restrictions, this might not be a reasonable solution (cf. Section I).

Another observation is that different features are highly ranked during the feature selection process by causal functional interactions (e. g. the function *Drive Mode Sport* triggers changes of other functions) rather than by human-based correlations. Hence, this “signal noise” might superimpose the “real” signals reflecting the drivers’ actions. With new functions interacting with multiple actuators and sensors, this superimposition will affect more functions in the future and should be considered for vehicle data analytics.

Another point of future investigation is the question *how* and *when* a driver shall be notified about a possible anomaly and *how* to handle users who never configure the function? Possible solutions are based on the automotive safety integrity level of a function including its severity, controllability, and exposure. One could study the optimal values for the number of ranked features and α.

Threats to the Validity: A major threat to validity concerns the selection of vehicles. There is a huge amount of in-house test vehicles data, but in our case, the approach has to be evaluated on real customer data, due to the behaviour we want to learn from the user. As a premium car manufacturer, we highly consider the privacy of the customers. Hence, it is very difficult—even as an OEM—to get a high number of full *real customer* car traces. We tried to prevent this by using traces at least from different regions of the world. We are aware of this issue and try to gather more data for the evaluation as future work. Another threat is the missing anomalous traces. We decided to evaluate our approach on nominal traces and perform a hypothetical switch to the different state, to not damage a car or even endanger yourself or someone else by creating anomalous traces with a real car.

VI. Conclusion

In this paper, we presented the first results of a study on anomaly detection and reported on a conducted real industrial case study. We proposed a novel approach for providing proactive safety management for customer function in automotive systems at runtime (cf. **RQ1**), which can detect safety violations for personalisable functions (cf. **RQ2**). This approach helps the software engineering process by gathering runtime data from the vehicles already runnings in the field (cf. **RQ3**). This approach was evaluated on real customer vehicle data.

In order to resolve the "curse of dimensionality", we applied the Fisher score feature selection algorithm and performed on the reduced data the statistical Grubbs outliers detection test pointing to anomalous system behaviour. All steps in our approach considered technical challenges of automotive E/E architectures in data analytics and were optimised on saving communication and computing performance.

Based on eight datasets, we traced a total of 17 state-of-the-art *driver assistance functions* per participants, yielding to a total of 136 runs. We observed that during 18% (24/136) of these runs, the user deactivated the *driver assistance functions* at least once for more than a few seconds. For over 50% (13/24) of these runs, we were able to detect and flag possible anomalies.

As at least one participant configured a convincingly large number of driver assistance functions, we need a dynamic system to monitor the configuration of these functions actively. Our approach was able to detect potential anomalies in up to 52% out of these reconfigurations. Initial results of the presented work have been presented as poster in [24]. These results are promising and will receive further attention in future work.

References

[1] M. Broy, "Challenges in automotive software engineering," in *28th International Conference on Software Engineering (ICSE 2006)*.

[2] SAE, *Taxonomy and Definitions for Terms Related to On-road Motor Vehicle Automated Driving Systems, SAE Standard J3016*, 2014.

[3] ISO, "Road vehicles–Functional safety (ISO 26262)," 2011.

[4] M. Gleirscher and S. Kugele, "From hazard analysis to hazard mitigation planning: The automated driving case," in *NASA Formal Methods – 9th International Symposium, NFM 2017*, ser. Lecture Notes in Computer Science, vol. 10227, 2017, pp. 310–326.

[5] S. Kugele, P. Obergfell, M. Broy, O. Creighton, M. Traub, and W. Hopfensitz, "On service-orientation for automotive software," in *International Conference on Software Architecture, ICSA 2017*. IEEE, 2017, pp. 193–202.

[6] R. Bellman, *Dynamic Programming*, 1st ed., ser. Dover Books on Computer Science. Princeton, NJ, USA: Princeton University Press and Dover Publications, 1957.

[7] C. Miller and C. Valasek, "Adventures in automotive networks and control units," IOActive, Tech. Rep., 2014.

[8] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *6th International Conference on Information Assurance and Security, IAS 2010*. IEEE, 2010, pp. 92–98.

[9] M. Müter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Intelligent Vehicles Symposium (IV) 2011*. IEEE, 2011, pp. 1110–1115.

[10] S. Das, B. L. Matthews, A. N. Srivastava, and N. C. Oza, "Multiple kernel learning for heterogeneous anomaly detection: Algorithm and aviation safety case study," in *International Conference on Knowledge Discovery and Data Mining, KDD'10*. ACM, 2010, pp. 47–56.

[11] S. D. Bay and M. Schwabacher, "Mining distance-based outliers in near linear time with randomization and a simple pruning rule," in *9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'03*. ACM, 2003, pp. 29–38.

[12] S. Budalakoti, S. Budalakoti, A. N. Srivastava, M. E. Otey, and M. E. Otey, "Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 39, no. 1, pp. 101–113, Jan 2009.

[13] S. Das, B. L. Matthews, and R. Lawrence, "Fleet level anomaly detection of aviation safety data," in *2011 IEEE Conference on Prognostics and Health Management*, June 2011, pp. 1–10.

[14] R. Baldoni, L. Montanari, and M. Rizzuto, "On-line failure prediction in safety-critical systems," *Future Generation Computer Systems*, vol. 45, pp. 123 – 132, 2015.

[15] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive CAN bus," in *2015 World Congress on Industrial Control Systems Security, WCICSS 2015*. IEEE, 2015, pp. 45–49.

[16] T. Hunt. (2016) Controlling vehicle features of nissan leafs across the globe via vulnerable apis. [Online]. Available: https://www.troyhunt.com/controlling-vehicle-features-of-nissan/

[17] P. Obergfell, C. Segler, E. Sax, and A. Knoll, "Synchronization between run-time and design-time view of context-aware automotive system architectures," in *2018 IEEE International Systems Engineering Symposium (ISSE)*, Oct 2018, pp. 1–3.

[18] D. E. Knuth, *Seminumerical algorithms*, third edition, thirty-third printing, newly updated and revised. ed., ser. The art of computer programming. Boston: Addison-Wesley, 2016, vol. 2.

[19] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[20] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed., ser. A Wiley-Interscience publication. New York NY: Wiley, 2001.

[21] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[22] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[23] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin de la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[24] C. Segler, S. Kugele, P. Obergfell, M. H. Osman, S. Shafaei, E. Sax, and A. Knoll, "Evaluation of feature selection for anomaly detection in automotive e/e architectures," in *41st International Conference on Software Engineering: Companion Proceedings, ICSE 2019, Montréal, Canada, May 25 - May 31, 2019*. IEEE, 2019, to appear.