2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

Evaluation of Feature Selection for Anomaly Detection in Automotive E/E Architectures

Christoph Segler*, Stefan Kugele†, Philipp Obergfell*, Mohd Hafeez Osman†, Sina Shafaei†, Eric Sax‡, and Alois Knoll†
*BMW Group Research, New Technologies, Innovations, Garching bei München, Germany
†Technical University of Munich, Department of Informatics, Garching bei München, Germany
‡Karlsruhe Institute of Technology, Department of Electrical Engineering and Information Technologies, Karlsruhe, Germany

***Abstract*—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 switched off by the user for personalisation. We aim to recognise mistimed and/or unintended deactivation of vehicle functions, in particular, *driver assistance functions* (ADAS), at run-time. This will be done in addition to already applied methods at design 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 and to support development processes. Based on eight customer datasets, we evaluated our approach on 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 de-activated 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 over the full trace.**

***Index Terms*—feature selection, anomaly detection, automotive, E/E architecture**

I. Introduction

We see an increase in *driver assistance functions* that either assist or take over in both normal and emergency scenarios. These functions can be intentionally switched off, but can also be deactivated unintentionally by (i) accident, (ii) a software or hardware malfunction, or (iii) an intrusion.

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, this is a severe problem that we try to identify and to mitigate. In addition to already applied methods at design time, we learn a *personalised driver model* that contains information about the context in which drivers intentionally deactivate functions. We refer to this as *nominal behaviour*. After detecting an anomaly the correctness of the detection must be validated by the driver. The finding is presented as a message on the head-unit with a warning explaining the affected function and its possible abnormal behaviour. The driver is expected to confirm whether the detection of the anomalous function was correct or the function was reconfigured by her/himself. If the detected anomaly gets validated it will be sent to the OEM for further analysis and investigation.

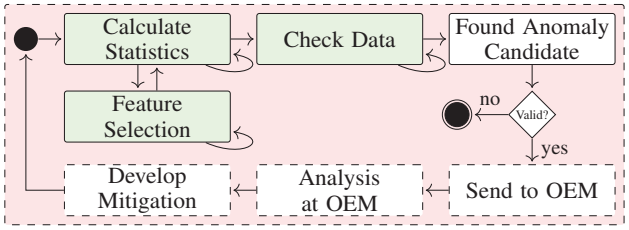

Fig. 1. Process: ↻-steps are repeatedly done at run-time.

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. Hence, our approach is designed to minimise stored and transmitted data, by running in the car in a decentralised structure without relying on more information as the current observed state of a signal and interim values for the calculation.

II. Approach

This work is a continuation and significant refinement and extension of the previous work [1] in which we showed that it is possible to learn the context of a car function when being used. Our refined approach is depicted in Fig. 1.

Calculation of Statistics: 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 and is generated based on status signals from the function itself. Based on this label the statistical distribution of all car signals in each class is calculated. The fundamental idea of this step is to *continuously update* the mean and standard deviation of each signal, and the mean and standard deviation of each signal in each class. This calculation is based on the semi-numerical calculation of Knuth [2] and Welford [3] and can be performed in a streaming manner, meaning no past data has to be stored. Moreover, 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 by the automotive E/E architecture.

2574-1934/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-Companion.2019.00104

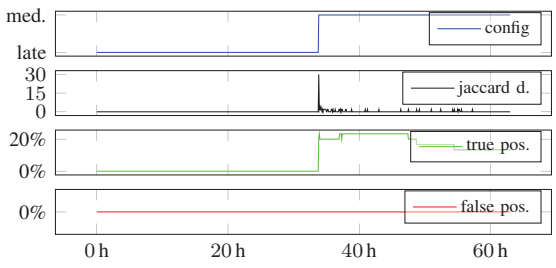

Fig. 2. Exemplary result for *Lane Departure Warning Sensitivity*

Feature Selection: To resolve the "curse of dimensionality" [4] posed by the high number of signals, our model selects a subset of signals showing the best correlations to the label (here: state of the *ADAS function*). This is achieved by using the supervised feature selection algorithm *Fisher score* [5]: This algorithm calculates a score for each signal/feature and is only based on values calculated in the previous step. 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. Only the top k ranked features are used for anomaly detection in the next step.

Checking New Data for Anomalies: For detecting anomalies, our approach is based on finding outliers for each of the previously selected top k features with the Grubbs' test [6]. Since this anomaly detection is only based on the previously calculated values and the current value of each feature, no additional data has to be stored. Additionally, this calculation is signal-independent and can be distributed in the architecture and calculated directly at the source of each signal.

Anomaly Feedback-Loop: After detecting an anomaly for a number of 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 a 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, then it will be sent to the OEM for further analysis and investigation.

III. Evaluation and Results

We used already collected data from a field study conducted at the BMW Group to evaluate our approach. The car data has been collected from eight current generation BMW 7 Series (G11), which have been equipped with hardware loggers. Each dataset contains ≈3.800 car signals and based on the time the car was driven, the number of samples varies from ≈170.000 to ≈850.000. In total we run our approach on 17 *ADAS functions* leading to 136 runs (17*8). The experimental setting requires a function that changed its state at least once for more than 1% of the total length of the trace, leading to 24 runs.

In Fig. 2 an exemplary result is shown. The upper plot shows the current configuration status of the function *Lane Departure Warning Sensitivity*. As depicted, the configuration is changed by the user in the second half of the trace from late to medium. The plot beneath the configuration shows the variation of the feature subsets by using the *Jaccard distance* [7] of the top k = 30 features between consecutive time steps of 300s. As seen right after the reconfiguration, the feature subset initialises, due to the first observation of another state, with less frequent changes over time. The next plot shows the *true positive* ratio of the anomaly detection on the top k = 30 features. This ratio considers the opposite state of the function recorded in the trace. This state was not recorded in the trace, but 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 should always be as high as possible. As seen in the plot, true positives are detected for ≈20% of the features with the initialisation of the feature set. Within this trace, no false positives are detected.

In total, our approach was able to detect possible anomalies in 13/24 runs. In 16/24 runs, the user was notified at least once when s/he changed the setting of the function, and in 11 runs no anomalies were detected if a change would not have been triggered by the user. When comparing the feature subsets between the different traces, we could not find any relationships and they highly vary between each car and function.

IV. Conclusion

In this work, we proposed a novel approach for providing proactive safety management for customer function in automotive systems at run-time, which can detect safety violations for personalisable functions. The approach supports the software engineering process by gathering run-time vehicle data. We evaluated the approach on real customer vehicle data.

We conclude from the first results 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. Furthermore, 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. This demands a learning approach and feature selection at run-time.

References

[1] 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.

[2] 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.

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

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

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

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

[7] 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.