Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard

Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard

Jens Henriksson
Semcon Sweden AB
Gothenburg, Sweden
jens.henriksson@semcon.com

Markus Borg
RISE SICS AB
Lund, Sweden
markus.borg@ri.se

Cristofer Englund
RISE Viktoria AB
Gothenburg, Sweden, Sweden
cristofer.englund@ri.se

ABSTRACT

Machine learning (ML) applications generate a continuous stream of success stories from various domains. ML enables many novel applications, also in safety-critical contexts. However, the functional safety standards such as ISO 26262 did not evolve to cover ML. We conduct an exploratory study on which parts of ISO 26262 represent the most critical gaps between safety engineering and ML development. While this paper only reports the first steps toward a larger research endeavor, we report three adaptations that are critically needed to allow ISO 26262 compliant engineering, and related suggestions on how to evolve the standard.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning**; • **Software and its engineering** → **Software safety**;

KEYWORDS

automotive software, machine learning, safety, interview study

ACM Reference Format:
Jens Henriksson, Markus Borg, and Cristofer Englund. 2018. Automotive safety and machine learning: Initial results from a study on how to adapt the ISO 26262 safety standard. In *SEFAIAS'18: SEFAIAS'18:IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems , May 28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3194085.3194090

1 INTRODUCTION

Machine learning (ML) has seen a drastic increase in popularity thanks to its capability of generalization throughout different domains. In the automotive field, there is a plethora of possibilities utilizing ML for advanced driver assistance systems and autonomous driving. Many companies are testing different ways to use deep learning (DL), for example with perception and end-to-end systems[1]. However, in the automotive field, safety is a high priority. Developing software adhering to safety standards requires rigorous engineering practices, enforced by standards like ISO 26262 [5]. Problems arise since the standards was not designed for tasks like ML. For DL, work on verification and validation methods have been suggested [7], as well as methods to improve stability [4].

The goal of this paper is to find what parts of ML need improved understanding, and which parts are already covered by the tools used today. This paper starts a new structured study on ML and automotive safety. We interview two experts in the field of functional safety and automotive software engineering.

The rest of the paper is organized as follow: Section 2 introduces related work, Section 3 gives the required background, Section 4 describes the research method. Section 5 summarizes the interviews, and finally, Section 6 concludes our work.

2 RELATED WORK

Salay *et al.* analyzed ISO 26262 from an ML perspective, and found 34 methods that apply at software unit level, and assessed their applicability to ML [8]. In addition, they described five areas that need to be updated to realize ML-based systems: hazard identification, fault and failure modes, the use of training sets, level of ML usage, and required software techniques.

Knauss *et al.* performed an interview study with 26 experts to elicit challenges when engineering autonomous cars [6]. Focusing on challenges in software testing, they report significant issues related to: 1) virtual testing and simulation, 2) safety, reliability, and quality, 3) sensors and their models 4) complexity of, and the amount of, test cases, and 5) hand-off between driver and vehicle.

Heckemann *et al.* conducted analytical work resulting in the identification of two primary challenges in developing autonomous vehicles adhering to ISO 26262 [3]. First, the driver is today considered to be part of the safety concept, but that will not be the case in future vehicles – the vehicle will perform driving maneuvers without interventions by a human driver. Second, the system complexity of modern vehicle systems is ever-growing as new functionality is continuously added by the automotive manufacturers to stay competitive on the market. This obstructs safety assessment – the more complex a system is, the harder it becomes for safety engineers to verify that all hazards have been addressed.

3 BACKGROUND

3.1 Automotive safety engineering

When engineering a safe system, safety must permeate all parts of the engineering process. This includes a systematic safety analysis and a methodological approach to managing risks. Safety analysis involves hazard identification and methods to eliminate or mitigate their consequences as well as verification activities that ensure that the safety methods are in place in the system.

SEFAIAS'18, May 28, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5739-5/18/05…$15.00
https://doi.org/10.1145/3194085.3194090

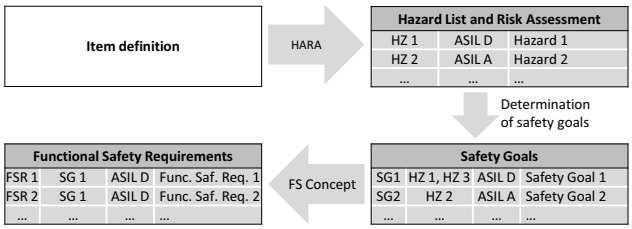

Figure 1: Flow of identifying hazards to creating functional safety requirements

The standard *ISO 26262* is an adaption of *IEC 61508: Functional safety*, to comply with the functional safety of road vehicles. The standard is present throughout the entire automotive life cycle and describes six phases: management, development, production, operation, service, and decommission – with support tailored to the necessary activities during the life cycle.

ISO 26262 recommends the usage of *Hazard Analysis and Risk Assessment* (HARA). The method is used to identify hazards in the system which are then used to determine safety goals and functional safety requirements, see Fig. 1. The hazards go through a risk-based assessment defined in the standard to determine an *Automotive Safety Integrity Level* (ASIL). The integrity level of a hazard depends on Severity: How many injuries can it cause; Exposure: expected frequency of the hazard; and Control: The likelihood of preventing the hazard. ASIL decides how rigorous testing, documenting, etc. are required activities for the item to reduce risk. The ASIL ranking ranges from ASIL D to ASIL A, where ASIL D represents the highest risk. If an item does not require safety management, it is labeled Quality Management (QM).

The standard consists of ten parts that follow a V-model for the different product development phases. Assuming that a safety-critical ML component will be realized as a software unit, especially the development phase on the software level (Part 6) enforces software engineering practices that must be interpreted in a ML context. As an example, for DL components it is unclear how to interpret process requirements on inspections/code reviews, unit test case generation using equivalence partitioning, and structural code coverage metrics. It is evident that certain ISO 26262 process requirements cannot be directly applied to DL.

3.2 Machine learning

Machine learning (ML) is the art and science of letting computers learn without being explicitly programmed. ML results in models that learn from and make predictions on data. Recent advances in ML are due to *Deep Learning* (DL), a subfield within ML, that allows for complex model architectures trained with large data sets. In contrast to regular algorithms, DL (and ML in general) creates trained models optimized for a given objective. This optimization is dependent on either data samples (supervised and unsupervised learning) or reward functions (reinforcement learning).

Due to the learning procedure of DL algorithms, there are multiple properties that affect safety aspects of the system. During the training phase, DL abstracts features from the given input domain which makes verification and validation hard, and causes violation of ISO 26262 development process. The abstractions in the neural net (the architecture of DL) results in DL being considered a black box, without transparency, due to the features being hard to interpret by humans. This pose a problem with the ISO 26262 since traceability is a requirement, which we have previously discussed in the automotive context [2].

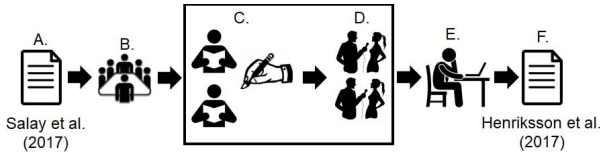

Figure 2: Overview of the research method.

The generalization in DL happens during the training phase. For supervised and unsupervised learning, the model is trained on a subset of inputs to the network. There is no guarantee that the subset is representative of the full input domain, thus assumption is that the model can generalize enough during training. The model is typically trained based on accuracy, a measurement of how often the correct option is chosen, and loss, i.e., a measurement of how far off the model is. Both measurements are used during the training phase, but the results are only an estimate, and do not correspond to the actual performance or reliability of the model while operating on the full input domain. In addition, there is a risk that the model will overfit to the subset during training, by learning certain patterns that merely occur beyond the training subset.

A common problem for DL models is instability. Small changes to the input can cause different predictions, thus a deep neural network (DNN) can be fooled by small changes to the input [4]. The problem occurs when training with local optimization methods, while there might be multiple local optima. Thus, different initializations of weights can cause different behaviors during training and inference, that make characteristics of the model hard to analyze or assign safety requirements.

4 RESEARCH METHOD

Our work constitutes an exploratory study on challenges when developing automotive software that rely on DL in safety-critical applications. We conducted a qualitative analysis based on experience from two experts on functional safety in the automotive domain. Fig. 2 shows an overview of our research method.

Our work was initially inspired by the 75 software development techniques in ISO 26262 analyzed by Salay *et al.* [8] (cf. A). We discussed the findings during a workshop in the SMILE[1] project (cf. B), and decided to limit the scope of our study to techniques mandatory for ASIL D development. At the SMILE workshop, we also identified the two experts from whom we collected opinions. Prior to contacting the experts, we extracted a list of the selected ISO 26262 software development techniques, accompanied with the respective comments of Salay *et al.*

The core of the data collection is presented in the box in Fig. 2, showing a two-staged process. First, we sent the prepared list to

[1] https://www.sics.se/projects/smile-ii

ISO-Table	Method	Adaptation
7	Semi-formal notations	Semi-formal notations on training phase
9	Inspection	Requires inspection of training tools and architecture design
9	Semi-formal verification	Semi-formal verification on training phase
9	Static code analysis	Covered with current tooling
10	Back-to-back comparison test	Additional requirements on test cases and sensitivity
12	Branch coverage	Covered with current tooling
12	MC/DC	Covered with current tooling

Table 1: Seven methods from ISO 26262-6 that need adaptations to cover ML according to Salay *et al.* [8]. The final column reports adaptation recommendations based on our two interviews.

the experts and requested written responses (cf. C). Both experts provided answers within two weeks. Second, to ensure correct interpretations, we interviewed the two experts in independent sessions (cf. D). The interviews followed an hour-glass structure, i.e., we started with general discussions on the topic, then asked specific questions related to our needs for clarifications, and then concluded with open-ended questions and a chance for the experts to add any further comments.

We recorded the two interview sessions, lasting 48 min and 30 min, respectively. We transcribed the recordings, and sent them back to the experts to enable a validation of the content. After receiving the validated transcripts, we conducted a side-by-side analysis of the written responses complemented with the interview transcripts (cf. E). Finally, we report our analysis in this paper (cf. F).

5 RESULTS AND DISCUSSION

Salay *et al.* identified 34 methods related to unit development in ISO 26262 part 6 where 27 of them are highly recommended (++) for ASIL D [8]. Most of these methods (e.g., "initialization of variables") exist to increase interpretability of the unit, which also applies to units that include ML. Similarly, methods like "fault injection test" and "resource usage test" are highly recommended and already applicable for ML. Out of the highly recommended methods for ASIL D, Salay *et al.* argues that seven require adaptation – these are listed in Table 1. Our interviewees had suggestions for how to proceed with these adaptations. We continue by presenting recommendations that they had in common, i.e., we boil down the recommendations for adaptation to three concrete suggestions.

Training phase: Requirements need to be moved from the actual ML application, to the training and architecture design phase. For example, a neural network is an artifact produced by a training phase that creates a functional mapping from an input to an output. Thus, the process itself that allows for the mapping, needs better understanding. Thus, rather than trying to understand the connections in a neural network, we suggest requirements on architectural design and training.

Model sensitivity: ML models are commonly sensitive to outside data. Whether or not a neural network has full branch coverage is irrelevant, but rather how sensitive it is to disturbances. For example, if one alters the input vector slightly the output should not suffer from a large step response. Fault injection is very important for ML, to allow for broad tests of the input domain.

Test case design: Similar to the training phase, the testing phase needs to be designed more thoroughly for ML. Test cases need to be designed to conduct correct evaluation, to ensure that functional expectations are met. The testing phase requires methods to detect when models are uncertain, and outside of the training domain. In addition, test and training phases need to be designed to handle additional data or data augmentation methods to improve training procedures, which increase test coverage and decrease model sensitivity.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we discuss what parts of ISO 26262 that need to be adapted to allow safety-critical ML development in the automotive context. Based on interviews with two experts, we highlight required changes to cover ML training, model sensitivity, and test case design. We plan to expand this initial study with more academic and industrial experts that are active in safety critical systems.

ACKNOWLEDGMENTS

This work was carried out within the SMILE II project financed by Vinnova/FFI. This work was partially supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP).

REFERENCES

[1] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. 2016. End to End Learning for Self-Driving Cars. *CoRR* abs/1604.07316 (2016). arXiv:1604.07316 http://arxiv.org/abs/1604.07316

[2] Markus Borg, Cristofer Englund, and Boris Duran. 2017. Traceability and Deep Learning - Safety-critical Systems with Traces Ending in Deep Neural Networks. In *In Proc. of the Grand Challenges of Traceability: The Next Ten Years*. 48–49.

[3] Karl Heckemann, Manuel Gesell, Thomas Pfister, Karsten Berns, Klaus Schneider, and Mario Trapp. 2011. Safe Automotive Software. In *Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, Berlin, Heidelberg, 167–176.

[4] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2016. Safety Verification of Deep Neural Networks. *CoRR* abs/1610.06940 (2016). arXiv:1610.06940 http://arxiv.org/abs/1610.06940

[5] Peter Kafka. 2012. The Automotive Standard ISO 26262, the Innovative Driver for Enhanced Safety Assessment & Technology for Motor Cars. *Procedia Engineering* 45 (2012), 2 – 10. https://doi.org/10.1016/j.proeng.2012.08.112 2012 International Symposium on Safety Science and Technology.

[6] Alessia Knauss, Jan Schroeder, Christian Berger, and Henrik Eriksson. 2017. Software-related Challenges of Testing Automated Vehicles. In *Proceedings of the 39th International Conference on Software Engineering Companion (ICSE-C '17)*. IEEE Press, Piscataway, NJ, USA, 328–330. https://doi.org/10.1109/ICSE-C.2017.67

[7] Gerald E. Peterson. 1993. Foundation for neural network verification and validation. (1993), 1966 - 1966 - 12 pages. https://doi.org/10.1117/12.152651

[8] Rick Salay, Rodrigo Queiroz, and Krzysztof Czarnecki. 2018. An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software. In *WCX World Congress Experience*. SAE International.