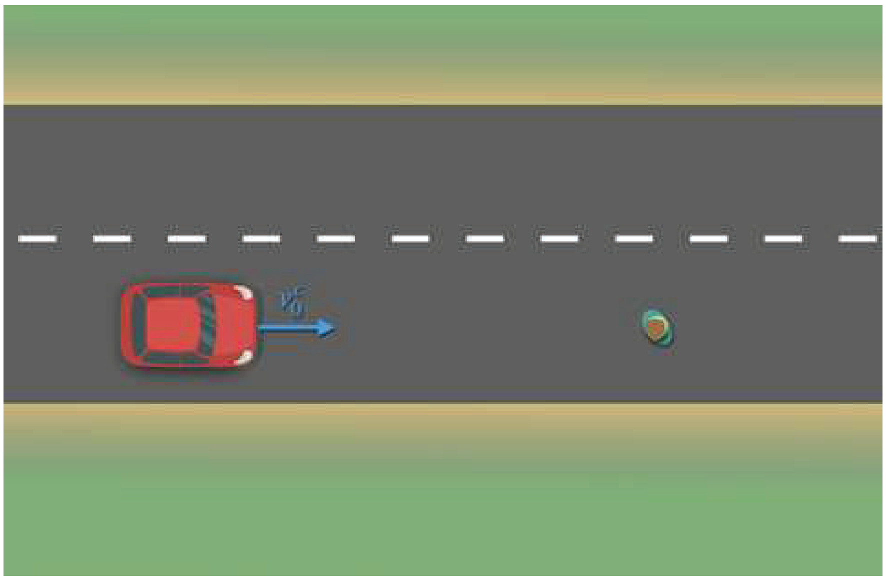
SMILE III

Public report



Authors: Markus Borg, Thanh Bui, Jens Henriksson, Martin Karsberg, Olof Lennartsson, Gustaf Bergström, Sebastian Brink, Sankar Sathyamoorthy, Erik Abenius

Date: 20220430

Project within: EMK

Table of Content

1 Sammanfattning

Framtidens autonoma fordon kommer att förlita sig på Deep machine learning algoritmer (DML), vars korrekta beteende inte kan garanteras genom traditionell mjukvaruteknik. Osäkerheten är ett stort hinder för att garantera kvaliteten på system som använder DML, särskilt i säkerhetskritiska applikationer.
SMILE III antar denna utmaning genom att utveckla metoder som gör att DML-baserade funktioner kan inkluderas i säkerhetskritiska applikationer med kvalitetskrav från industriella standarder. Baserat på resultaten från SMILE I/II kommer SMILE III att vidareutveckla säkerhetsburskonceptet till en referenssystemarkitektur och prototyper, tillsammans med datahanteringsmetoder, som ska underlätta efterlevnad av de utvecklande säkerhetsstandarderna (inklusive den otillräckligt föreskrivna standarden SOTIF).
Resultaten från SMILE III kommer att vara av intresse för fordonstillverkare som planerar att använda DML i säkerhetskritiska applikationer. Resultaten kommer också att ge värdefull input för standardens utveckling. Projektkonsortiet består av forskningspartners: RISE (huvudparter), QRTECH, Semcon, Combitech, ESI Nordic och Infotiv.

2 Executive summary in English

Future autonomous vehicles will rely on Deep machine learning algorithms (DML), whose correct behavior cannot be guaranteed by traditional software engineering approaches. This uncertainty is a big obstacle to ensure the quality of systems using DML, especially in automotive safety critical applications. The SMILE program has been proposed to address this challenge via developing method(s) that allow DML-based functions to be included into safety critical vehicular applications with quality control requirements from industrial standards. The SMILE program consists of several consecutive research projects:

- SMILE I project studied state-of-art within Verification and Validation for DML systems and mapping of the challenges faced by the automotive industry [1]. SMILE I concluded that we should focus on a Safety Cage Concept within the follow-up project(s).
- SMILE II project investigated the latest vehicle safety standards: (i) ISO 26262 [2] - functional safety, and lately complemented by (ii) PAS specification of ISO 21448 [3] - Safety of the Intended Function (SOTIF); implemented and tested a number of safety cage architectures in simulators [4], developed a framework for supervisor comparison [5]; and developed future research tool sets including generated datasets and an end-to-end vehicle controller. SMILE II observed that ISO 26262 alone cannot accommodate ML-based systems [6]–[8], the complement with the most recently published SOTIF (2019) will be more applicable. However, in its PAS edition, SOTIF only focused on what should be covered during systems engineering and leaves out how to achieve these goals in practice.

With the success and promising results from SMILE I/II, the SMILE III project further develops the safety cage concept into a reference system architecture and prototype(s), together with data management approaches, facilitating compliance with the evolving safety standards (including the insufficient prescriptive ISO/PAS 21448 edition).
The results from SMILE III will be of interest for automotive actors that plan to use DML to support autonomous functions in safety critical applications. These results will also provide valuable inputs for the development of related standards. The partners of project consortium are: RISE (lead partner), QRTECH, Semcon, Combitech, Infotiv, and ESI Nordics.

3 Background

Autonomous and highly automated vehicles currently have a considerable momentum[9], where DML is a critical enabling technology. DML is typically used to extract a digital representation of the surroundings from high-dimensional sensor inputs. The DML-algorithms have proven themselves successful for perceiving objects within the complex traffic [10], [11]. Since traffic is highly dynamic the perception models need to be able to generalize well to handle all new situations. However, no machine learning model is sufficiently complete to avoid misbehavior under all circumstances on the road [12], thus also DML will sometimes fail to generalize. Unfortunately, the models trained using DML are particularly opaque in nature. They often consist of huge networks with a number of weight parameters in the order of magnitude of hundreds of millions [6] . Consequently, there are very limited options to analyze miss-classifications from a functional safety perspective, as neither traditional code reviews nor exhaustive safety analysis techniques are possible.

Within this SMILE program, we first accepted that the DML-system will make miss-classifications. The predecessor projects SMILE I/II drives the program into the development of a run-time monitoring system for DML-based perception using the concept of adaptive safety cage architectures [7], or as referred to by [8]: safety supervisors. In the latest SMILE II project, we envisioned a safety cage, encapsulating the DML-based perception model [4], [5] capable of monitoring the input to the DML-based system and thus being able to predict potential anomalies in the model's classification uncertainty. The work in [4] describes this as a classifier having a reject option when the uncertainty is too high, e.g., forcing a human to intervene. In line with the proposal by [7], we will distinguish between a safe region of operation and an invalid region that could lead to a hazardous situation. If the DML-based perception enters the invalid region, the safety cage will invoke an appropriate safe action, such as graceful degradation based on deterministic algorithms. The motivation for focusing on safety cages is that current alternatives to prevent system failures, e.g., fault prevention and fault avoidance, cannot address all malfunctions due to the complexity of the system and the non-deterministic traffic environment[5].

4 Purpose, research questions and methods

We therefore set our long-term goal of the SMILE program to accomplish functional safety by developing the safety cage as a functionally redundant system to the actual control system. For such a solution, the highly complex control function (i.e., applying DML) could be developed according to the quality management standard, whereas the comparably simple safety cage could be addressed by traditional verification and validation (V&V), or possibly even proven correct using formal verification methods [13], [14]. This vision is aligned with the structure of both existing standards: ISO 26262 (FUSA) and ISO/PAS 21448 (ability to control the system performances in detailed scenarios).

In alignment to the program's goal, SMILE III project further develops the concept to address the following research questions:

- RQ1: To allow compliance with the future automotive safety standards, what does a strategy for development, operation, and evolution of safety-critical functions that use DML cover? How can the corresponding safety cases be developed?
- RQ2: In the light of the proposed safety strategy, how can DML and supervisor components work together in a safety cage construct? How can it be adapted to fit the overall system architecture?
- RQ3: To support evolution of the safety-critical functions, how can data management processes be applied to monitor (and improve) the system performance?

SMILE III has connection to other ongoing research initiatives. During the project lifetime, its developing approach has also been adopted by VALU3S project[15] that creates a framework for verification and validation of multiple domains with a specific application in traffic infrastructure use case (UC1).

5 Goal

The overall goal of the SMILE program is to develop enabling technologies that can be used in vehicles to reduce the number of injuries and fatalities in traffic. This will be achieved in close collaboration between research institutes, SME, OEM, and academia in long-term including project lifetime, i.e., the strong SMILE consortium will contribute to Sweden's international competitiveness in machine learning for safety critical applications. This project will strengthen the machine learning competence within the Swedish automotive industry, in particular to support Verification and Validation (V&V) of DML-based solutions - a prerequisite to allow innovative solutions related to functional safety within the complex architecture of electrical systems of cars as pointed out in the Strategic Agenda of the Machine learning within FFI.

Since the area of autonomous vehicles where machine learning is a key-technology is developing fast, it is important to have projects that mix the different aspects of machine learning and in particular, as is the case in this project, have a focus on applicability. Machine learning has the potential to change the industry, making vehicles more intelligent and personalized. However, to be able to exploit the full potential, knowledge about how these methods can be used in a safe and secure way within vehicles is required. This is therefore the focus of this project: To develop methods and tools that can be used to enable safe application in vehicles through verification and validation of machine learning-based systems. This is well in-line with the overall activities within the Electrical and Electronics Systems Engineering domain. To deepen the knowledge in both on- and off-board systems as well as V&V of automotive software.

SMILE III proudly contributes to the development of three new Swedish researchers:

- Jens Henriksson (Semcon) successfully completed his Licentiate in June 2020 and is getting his Ph.D. later this year 2022 (Chalmers University of Technology). His work within the project has been performed in close collaboration with Chalmers research (main supervisor Prof. Christian Berger) while ensuring industry relevance through his industry supervisor (Stig Ursing).
- Mahshid Helali Moghadam is an industrial Ph.D. student at Mälardalen University. She had been employed by RISE as a researcher. She will get her Ph.D. degree shortly June 2022.
- Kasper Socha will finish his Master degree 2022 (Lund University) with his development work within SMIRK. Kasper had recently been employed by RISE as a researcher.

Infotiv, COMBITECH and QRTECH that are members of the project, have broad knowledge within V&V and embedded systems, will be capable of cross fertilizing their knowledge into the automotive industry. RISE, Infotiv and QRTECH are actively involved in an EU project VALU3S [15] and further develop the results from SMILE3 to extend into V&V for traffic infrastructure systems.

6 Results and deliverables

6.1 SMIRK MVP

SMILE III focuses on addressing the safety assurance practices from a system point of view. It thus resulted in a need to have a common and representative use case where all the challenges are presented in its minimal forms.

SMIRK, an experimental pedestrian emergency braking ADAS, has thus been developed and published as a research prototype to facilitate research topics within this project. Together with all related published materials and datasets, we hope that the prototype will facilitate researchers in this area to conduct further research and findings.

Figure 1: SMIRK logo

SMIRK responds to calls for a fully transparent ML-based Advanced Driver-Assistance System (ADAS) to act as a system-under-test in research on trusted AI [16]. SMIRK provides pedestrian emergency braking. By combining trained and coded software, SMIRK is intended to become a baseline Software-Under-Test (SUT) for ML testing research targeting automotive perception applications such as object detection and path planning. To ensure industrial relevance, SMIRK will implement a reference architecture while adhering to development practices mandated by contemporary automotive safety standards [3] and is complemented by a fully transparent safety case.

SMIRK is accessible at https://github.com/RI-SE/smirk. Readers interested in more details can consult our preprint paper [17]. Furthermore, AI Sweden has promised to host the SMIRK dataset (~185 GB) as part of the concurrent Vinnova project RoDL Road Data Lab.

6.2 WP2-Architecture Design

6.2.1 Reference architecture design

We constructed a reference architecture design for the designed experimental setup within SMIRK. The reasoning behind the architecture lies in traditional object detection and classification that most commonly consist of a camera and radar setup that works in a sensor fusion setup to provide situational awareness of surrounding objects. In SMILE III, as the focus has been on verification and validation of machine learning (ML) based systems, it was deemed that a similar sensor setup would be used. To connect to the ML field, a large portion of research is on image neural networks, typically designed as convolutional neural networks (CNN's). The SMIRK architecture uses a CNN as a pedestrian detector, see the gray boxes in Figure 2, that is providing estimates of where in scene pedestrians are located. Combining this with the information from the Radar gives a perception orchestrator that will inform the system if a collision course is detected.

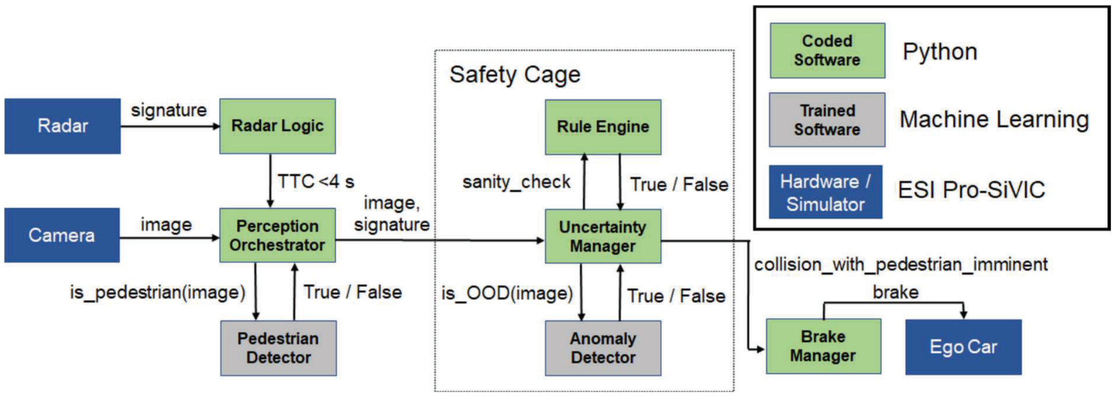


Figure 2: Reference safety architecture

The dashed box in Figure 2 represents one safety measure that is operating on the output of the perception orchestrator. The reasoning being that potential faults in e.g. the pedestrian detector may contribute to failures, and applying a safety cage should be seen as a potential mitigation strategy. The potential risks of the system were analysed in a hazard assessment and risk analysis (HARA) early in the project, and was used as motivation for a set of specific risks that was investigated in SMIRK, namely the effect of missing or overpredicting pedestrians in a scene (so called breaking for ghosts).

For the architectural design, a break manager was also considered and implemented in the Pro-Sivic simulator. In short, the system acts if a pedestrian has been identified in a collision path with the ego vehicle, and thus, a breaking signal is activated.

6.2.2 Safety measures and argumentation

During the project, the different safety requirements are motivated and verified through a set of experiments, where additional safety measures are put in place if the system does not fulfill the requirements by itself. For example, the pedestrian orchestrator (as well as most CNN's) are trained with a majority of true positives (correctly detected pedestrians) and seldom with true negatives (correctly not classified pedestrians). This approach is preferred, as it maintains fidelity by only operating on samples that are within the training domain, and more specifically, within the training set. While empty scenes are included in the training, it is not enough to completely eliminate the occurrence of false positives (wrongly detecting a pedestrian).

SMIRK attempts to mitigate false positive and false negative instances through anomaly detection methods, which are investigated in the safety case work package. SMIRK constructed and derived metrics based on the safety and robustness requirements put on the machine learning part of the system. We argue that, by passing these metric requirements, the system is deemed safe for this ODD, see specifically artifact [W] of SMIRK.

Furthermore, the work package frequently discussed occlusion as an interesting topic. A separate experimental platform was constructed to study what effect occlusion would have on the pedestrian detection module. The same network architecture was used, specifically a YOLOv5s model, ergo a small scale convolutional neural network bottleneck combined with the traditional anchor point implementation introduced in YOLO [18].The result was, perhaps predictably, that the performance was reduced for occluded objects. However, the paper also presented a safety measure that investigated cascading networks as a safety measure, i.e. using sub-networks when the output was deemed too unreliable. The results were presented at SEAA 2021 [19].

6.3 WP3-Safety Case and Strategy

The overall SMIRK safety strategy and safety case development builds on SOTIF and a methodology called Assurance of Machine Learning for use in Autonomous Systems (AMLAS) developed by the Assuring Autonomy International Programme at the University of York, UK. During SMILE III, we invited the developers of AMLAS to give a private tutorial for the project participants. Our invitation was accepted, and in May 2021 the tutorial was given.

6.3.1 SMIRK Safety Strategy

The overall development framework for SMIRK is the SOTIF process (see Figure 3). The process starts in the upper left with A) Requirements specification. Based on the requirements, a B) Risk Analysis is done. For each identified risk, its potential *Consequences* are analyzed. If the risk of harm is reasonable, it is recorded as an acceptable risk. If not, the activity continues with an analysis of *Causes*, i.e., an identification and evaluation of triggering conditions. If the expected system response to triggering conditions is acceptable, the SOTIF process continues with V&V activities. If not, the remaining risk forces a C) Functional Modification with a corresponding requirements update.

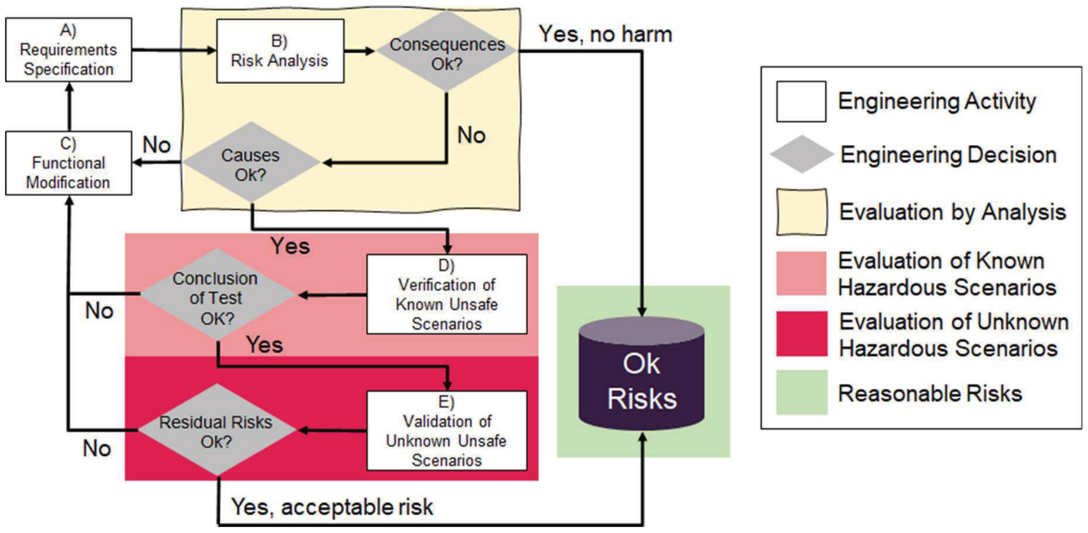

A) Requirements Specification
B) Risk Analysis
Consequences Ok?
Yes, no harm
No
C) Functional Modification
No
Causes Ok?
Yes
Conclusion of Test OK?
D) Verification of Known Unsafe Scenarios
No
Yes
Ok Risks
Residual Risks Ok?
E) Validation of Unknown Unsafe Scenarios
No
Yes, acceptable risk
Engineering Activity
Engineering Decision
Evaluation by Analysis
Evaluation of Known Hazardous Scenarios
Evaluation of Unknown Hazardous Scenarios
Reasonable Risks

Figure 3: A simplified overview of the SOTIF process

The lower part of Figure 3 shows the V&V activities in the SOTIF process. For each risk, the development organization conducts D) Verification to ensure that the system satisfies the requirements for the known hazardous scenarios. If the *Conclusion of Verification Tests* are satisfactory, the V&V activities continues with validation. It not, the remaining risk requires a C) Functional Modification. In the E) Validation, the development organization explores the presence of unknown hazardous scenarios - if any are identified, they turn into known hazardous scenarios. The *Conclusion of Validation Tests* estimates the likelihood of encountering unknown scenarios that lead to hazardous behavior. If the residual risk is sufficiently small, it is recorded as an acceptable risk. If not, the remaining risk again necessitates a C) Functional Modification.

The SMIRK safety strategy is guided by AMLAS. AMLAS provides an overall process and a set of safety case patterns for safety assurance of ML components. Figure 4 shows an overview of the six stages of AMLAS. The upper part stresses that the development of an ML component and its corresponding safety case is done in the context of larger systems context. In our case, the larger context is the development of the SMIRK ADAS, indicated by the blue arrow. The AMLAS process starts in the System Safety Requirements, which in our case come from following the SOTIF process. However, both SOTIF and AMLAS are iterative process, which means that their activities are performed in parallel and there are many interdependencies - for AMLAS, the iteration is highlighted by the black arrow in the bottom of the figure.

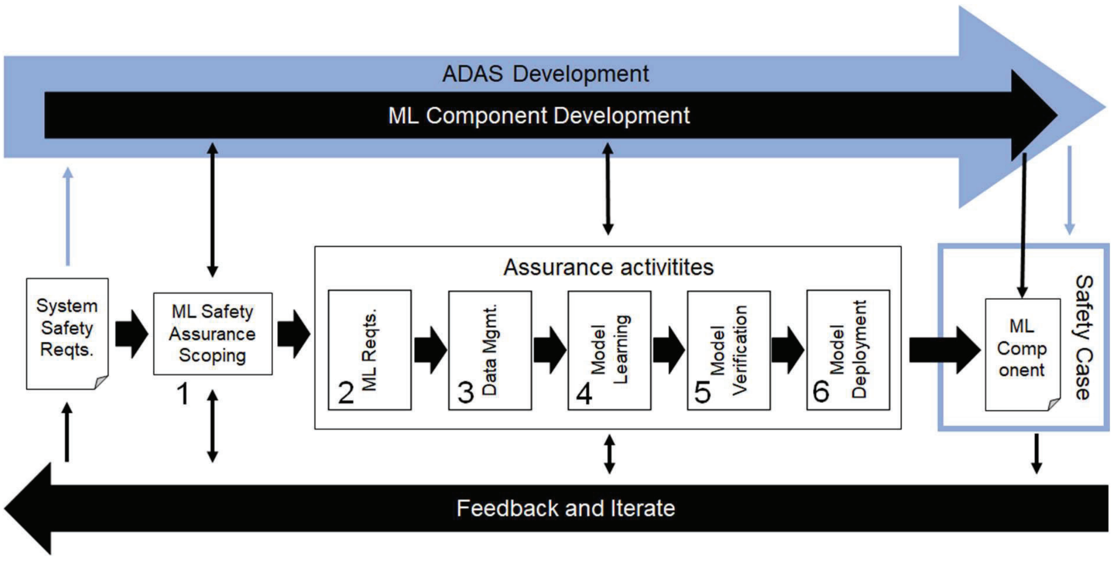


Figure 4: Overview of the AMLAS framework

Starting from the System Safety Requirements from the left, Stage 1 is *ML Safety Assurance Scoping*. This stage operates on a systems engineering level and defines the scope of the safety assurance process for the ML component as well as the scope of its corresponding safety case - the interplay with the non-ML safety engineering is fundamental. The next five stages of AMLAS all focus on assurance activities for different constituents of ML development and operations. Each of these stages conclude with an assurance argument that when combined, and complemented by evidence, compose the overall ML safety case.

2. *ML Safety Requirements Assurance*. Requirements engineering is used to elicit, analyze, specify, and validate ML safety requirements in relation to the software architecture and the ODD.
3. *Data Management Assurance*. Requirements engineering is first used to develop data requirements that match the ML safety requirements. Subsequently, data sets are generated (development data, internal test data, and verification data) accompanied by quality assurance activities.
4. *Model Learning Assurance*. The ML model is trained using the development data. The fulfilment of the ML safety requirements is assessed using the internal test data.
5. *Model Verification Assurance*. Different levels of testing or formal verification to assure that the ML model meets the ML safety requirements. Most importantly, the ML model shall be tested on verification data that has not influenced the training in any way.
6. *Model Deployment Assurance*. Integrate the ML model in the overall system and verify that the system safety requirements are satisfied. Conduct integration testing in the specified ODD.

The rightmost part of the figure shows the overall safety case for the system under development with the argumentation for the ML component as an essential part, i.e., the target of the AMLAS process. The AMLAS argumentation patterns for SMIRK are all presented using the graphical format Goal Structuring Notation (GSN) and can be found in the GitHub repository.

6.3.2 SMIRK Safety Case

Table 1 provides an overview of how the 34 artifacts resulting from following AMLAS present a complete safety cage for the ML component in SMIRK. From a bird's eye view, the artifacts include six argument patterns ([F], [I], [R], [W], [BB], and [GG]) that are instantiated using the other artifacts. While all artifacts can be found on GitHub, this section presents a particularly interesting sample.

Table 1: SMIRK Safety Assurance Table. Numbers in the Input/Output columns refer to the six stages

ID	AMLAS Artifact Title	Input to	Output from
[A]	System Safety Requirements	1, 6	
[B]	Description of Operating Environment of System	1, 6	
[C]	System Description	1, 6	
[D]	ML Component Description	1	
[E]	Safety Requirements Allocated to ML Component	2	1
[F]	ML Assurance Scoping Argument Pattern	1	
[G]	ML Safety Assurance Scoping Argument		1
[H]	ML Safety Requirements	3, 4, 5	2
[I]	ML Safety Requirements Argument Pattern	2	
[J]	ML Safety Requirements Validation Results		2
[K]	ML Safety Requirements Argument		2
[L]	Data Requirements		3
[M]	Data Requirements Justification Report		3
[N]	Development Data		3
[O]	Internal Test Data		3
[P]	Verification Data		3
[Q]	Data Generation Log		3
[R]	ML Data Argument Pattern	3	
[S]	ML Data Validation Results		3
[T]	ML Data Argument		3
[U]	Model Development Log		4
[V]	ML Model	5, 6	4
[W]	ML Learning Argument Pattern	4	
[X]	Internal Test Results		4
[Y]	ML Learning Argument		4
[Z]	ML Verification Results		5
[AA]	Verification Log		5
[BB]	ML Verification Argument Pattern	5	
[CC]	ML Verification Argument		5
[DD]	Erroneous Behaviour Log		6
[EE]	Operational scenarios	6	
[FF]	Integration Testing Results		6
[GG]	ML Deployment Argument Pattern	6	
[HH]	ML Deployment Argument		6

This highest level SMIRK requirement is:

- **SYS-SAF-REQ1:** SMIRK shall commence automatic emergency braking if and only if collision with a pedestrian on collision course is imminent.

Based on a Hazard Analysis and Risk Assessment (HARA), two categories of hazards were identified. First, SMIRK might miss pedestrians and fail to commence emergency braking - we refer to this as a missed pedestrian. Second, SMIRK might commence emergency braking when it should not - we refer to this as an instance of ghost braking.

SYS-SAF-REQ1 into two separate requirements corresponding to missed pedestrians and ghost braking, respectively.

- **SYS-ML-REQ1:** The pedestrian recognition component shall identify pedestrians in all valid scenarios when the radar tracking component returns a time to collision (TTC) < 4s for the corresponding object.
- **SYS-ML-REQ2:** The pedestrian recognition component shall reject false positive input that does not resemble the training data.

These requirements are further detailed through performance requirements and robustness requirements.

For objects detected by the radar tracking component with a TTC < 4s, the following requirements must be fulfilled:

- **SYS-PER-REQ1:** The pedestrian recognition component shall identify pedestrians with a true positive rate of 93% when they are within 80 m.
- **SYS-PER-REQ2:** The false negative rate of the pedestrian recognition component shall not exceed 7% within 50 m.
- **SYS-PER-REQ3:** The false positive per image of the pedestrian recognition component shall not exceed 0.1% within 80 m.
- **SYS-PER-REQ4**: In 97% of sequences of 5 consecutive frames from a 10 FPS video feed, no pedestrian within 80 m shall be missed in more than 20% of the frames.
- **SYS-PER-REQ5:** For pedestrians within 80 m, the pedestrian recognition component shall determine the position of pedestrians within 50 cm of their actual position.
- **SYS-PER-REQ6:** The pedestrian recognition component shall allow an inference speed of at least 10 FPS in the ESI Pro-SiVIC simulation.

For pedestrians present within 80 m of ego car, captured in the field of view of the camera:

- **SYS-ROB-REQ1:** The pedestrian recognition component shall perform as required in all situations ego car may encounter within the defined ODD.
- **SYS-ROB-REQ2:** The pedestrian recognition component shall identify pedestrians irrespective of their upright pose with respect to the camera.
- **SYS-ROB-REQ3:** The pedestrian recognition component shall identify pedestrians irrespective of their size with respect to the camera.
- **SYS-ROB-REQ4:** The pedestrian recognition component shall identify pedestrians irrespective of their appearance with respect to the camera.

Figure 5: SMIRK ML Safety Requirements Argument Pattern.Figure 5 shows the ML safety requirements argument pattern for SMIRK. The top claim is that system safety requirements that have been allocated to the ML component are satisfied by the model that is developed (G2.1). This is demonstrated through considering explicit ML safety requirements defined for the ML model [H]. The argument approach is a refinement strategy translating the allocated safety requirements into two concrete ML safety requirements (S2.1) provided as context (C2.1). Justification J2.1 explains how we allocated safety requirements to the ML component as part of the system safety process, including the HARA.

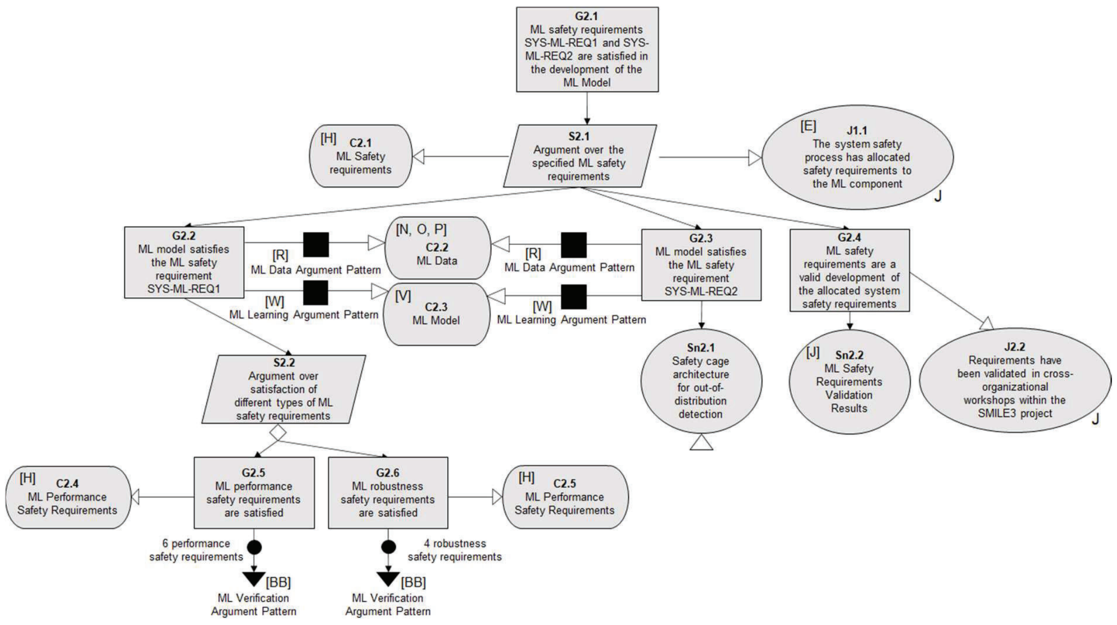

G2.1 ML safety requirements SYS-ML-REQ1 and SYS-ML-REQ2 are satisfied in the development of the ML Model
[H] C2.1 ML Safety requirements
S2.1 Argument over the specified ML safety requirements
[E] J1.1 The system safety process has allocated safety requirements to the ML component
J
G2.2 ML model satisfies the ML safety requirement SYS-ML-REQ1
[R] ML Data Argument Pattern
[N, O, P] C2.2 ML Data
[R] ML Data Argument Pattern
G2.3 ML model satisfies the ML safety requirement SYS-ML-REQ2
G2.4 ML safety requirements are a valid development of the allocated system safety requirements
[W] ML Learning Argument Pattern
[V] C2.3 ML Model
[W] ML Learning Argument Pattern
Sn2.1 Safety cage architecture for out-of-distribution detection
[J] Sn2.2 ML Safety Requirements Validation Results
J2.2 Requirements have been validated in cross-organizational workshops within the SMILE3 project
J
S2.2 Argument over satisfaction of different types of ML safety requirements
[H] C2.4 ML Performance Safety Requirements
G2.5 ML performance safety requirements are satisfied
G2.6 ML robustness safety requirements are satisfied
[H] C2.5 ML Performance Safety Requirements
6 performance safety requirements
4 robustness safety requirements
[BB] ML Verification Argument Pattern
[BB] ML Verification Argument Pattern

Figure 5: SMIRK ML Safety Requirements Argument Pattern.

Strategy S2.1 is refined into two subclaims about the validity of the ML safety requirements corresponding to missed pedestrians and ghost braking, respectively. Furthermore, a third subclaim concerns the satisfaction of those requirements. G2.2 focuses on the ML safety requirement SYS-ML-REQ1, i.e., that the nominal functionality of the pedestrian recognition component shall be satisfactory. G2.2 is considered in the context of the ML data (C2.2) and the ML model (C2.3), which in turn are supported by the ML Data Argument Pattern [R] and the ML Learning Argument Pattern [W]. The argumentation strategy (S2.2) builds on two subclaims related to two types of safety requirements with respect to safety-related outputs, i.e., performance requirements (G2.5 in context of C2.4) and robustness requirements (G2.6 in context of C2.5). The satisfaction of both G2.5 and G2.6 are addressed by the ML Verification Argument Pattern [BB]. G2.3 focuses on the ML safety requirement SYS-ML-REQ2, i.e., that the pedestrian recognition component shall reject input that does not resemble the training data to avoid ghost braking. G2.3 is again considered in the context of the ML data (C2.2) and the ML model (C2.3). For SMIRK, the solution is the safety cage architecture (Sn2.1) developed in SMILE II and further explored in WP4.

Subclaim G2.4 states that the ML safety requirements are a valid development of the allocated system safety requirements. The justification (J2.2) is that the requirements have been validated in cross-organizational workshops within the SMILE III project. We provide evidence through ML Safety Requirements Validation Results [J] originating in a Fagan inspection, a formal requirements inspection done as part of the SMILE III project in the end of 2021 (Sn2.2) – the inspection protocol is available on GitHub.

The ML Requirements Assurance is at the core of the SMIRK safety case. Still, all six stages of AMLAS must be followed to generate the artifacts needed to constitute a complete safety case for the ML component in SMIRK. On GitHub, interested readers can find the output from the *Data Management Assurance* (including data requirements fulfilling the four assurance-related desiderata 1) Relevance, 2) Completeness, 3) Balance, and 4) Accuracy), *ML Learning Assurance* (including a model component description and its model development log), *ML Verification Assurance* (including an ML test strategy with data testing, model testing, and system testing) and *ML Deployment Assurance* (including a description of the operational environment and an erroneous behavior log based on the system testing results). Our preprint paper presents all results in detail.

6.4 WP4-Safety cage

6.4.1 Safety cage Design and Optimization

In SMILE II, QRTECH developed a safety cage concept based on the statistical analysis of neuronal activations in the neural network. This was an empirical method that was tested for different datasets. Based on the positive results we found, a full demonstrator was developed where the safety cage accepted or rejected the classification of a semantic segmentation network (Mask R-CNN).

In SMILE III, the aim was to explore complementary approaches to safety cage, especially we were interested to verify the performance of more mathematically grounded approaches and compare it to the safety cage developed in SMILE II. In this regard, a particular topic of interest is the Bayesian approaches to quantify uncertainty in neural networks. In [20], the authors explain two types of uncertainties that are important for computer vision. Aleatoric uncertainty is used to capture inherent noise in the data while the epistemic uncertainty captures the uncertainty in the model, which can be reduced given enough data. For safety critical applications, it is important to calculate the epistemic uncertainty as this can be used to detect out of distribution objects. Thus for our use case to develop a safety cage, we focus on the epistemic uncertainty.

As a full Bayesian neural network (BNN) is computationally expensive, approximations to BNNs have been proposed. In particular, Gal and Ghahramani, proposed using dropouts as an approximate method to get uncertainties from neural network [21]. In this case, dropouts are not only used while training as a regularization technique, but also used during inference. A single input is then sent multiple times through the network, which varies slightly for each pass due to the dropouts. The scores from several of these passes are then collected, the mean and standard deviation of which gives the actual score and the uncertainty in the prediction. The uncertainty calculated in this way, gives the epistemic uncertainty also known as model uncertainty.

An initial test based on the above method was performed by training a simple neural network on MNIST and using omniglot as the outlier data. Using the dropouts during prediction, we could see that the uncertainties for the outlier data was higher than for the inlier data. A comparison between this result and SMILE II safety cage showed that dropouts were better at catching outliers. However, the questions of scalability and real time performance remained, especially because the predictions have to be run enough

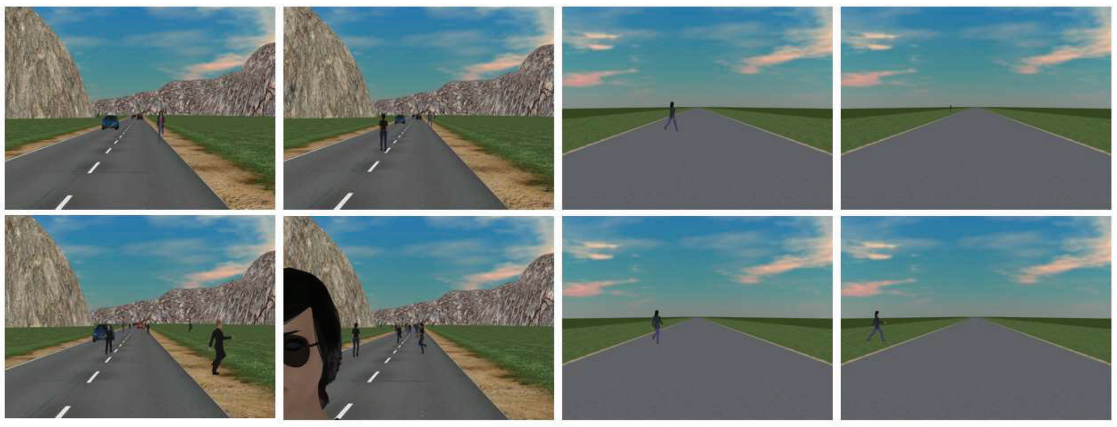


Figure 6 Sample images from training set generated from Pro-SiVIC

number of times so that statistical variables like mean and standard deviations can be calculated. As a first step to check scalability of the above method, we added dropouts to Mask R-CNN and trained them on the highway dataset generated in SMILE II. The dataset contains around 3000 of each car, truck and motorcycles and was generated from the Pro-SiVIC simulator. The network however did not converge during training and did not detect any of the objects in the image. One of the reasons for this might be that, we need significantly more data for training the network with dropouts than the one without dropouts.

At this point, a survey of existing implementations of neural networks that could detect uncertainty was performed and the promising methods were compiled. Based on the requirements for SMIRK, we decided to test the Bayesian version of the YOLOv3 network presented in [22], along with the source code in Github (https://github.com/flkraus/bayesian-yolov3). While the code is shared publicly, the trained weights are not shared. The authors in this work have trained their network on the Eurocity persons dataset, which consists of around 47000 images in which 218000 pedestrians and 20000 riders are labeled. The Eurocity persons dataset is however only available for academic or nonprofit institutions.

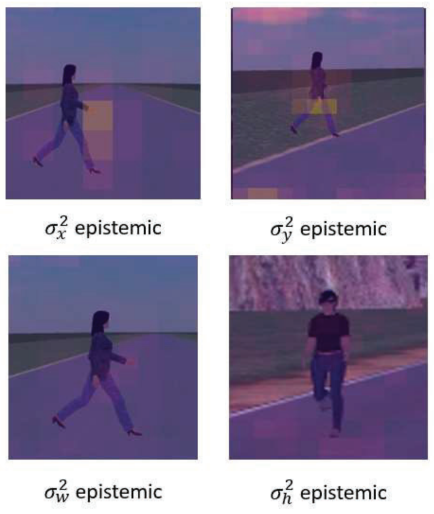


Figure 7: Epistemic uncertainty from Bayesian Yolo network. The yellower regions of the image corresponds to higher uncertainty regions as per the network.

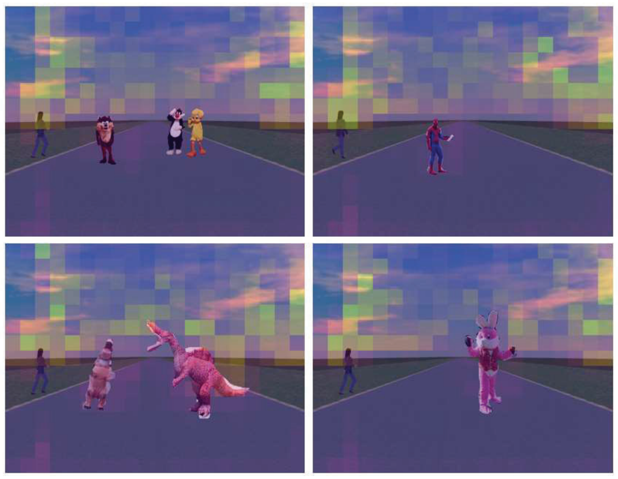


Figure 8: Class MI for outlier data. The yellow regions show where the network is uncertain on average.

In our work, we used the above implementation of the Bayesian Yolov3 code with the initial weights set to the original Yolov3 weights. We used around 10000 images from the WIDER pedestrian detection dataset (https://wider-challenge.org/2019.html) for pretraining the network. The images in the WIDER dataset are from both surveillance cameras and cameras located on driving vehicles. The actual training of the network was done using the output images from Pro-SiVIC. The Pro-SiVIC scenarios were developed by RISE along with some random changes from QRTECH, giving a total of around 7000 images. A sample set of images is shown in Figure 6. We tested the trained network on Pro-SiVIC images containing single and multiple pedestrians on the street. While the network was able to identify the pedestrians, the uncertainties we got from the network were not easily explainable. The results from the test set are shown in Figure 7. We also tested uncertainties with regards to the class for an outlier set that was synthetically created. These results are shown in Figure 8. As mentioned above and can be seen in the figures, the results were not satisfactory. There could be few reasons for the same. As mentioned in the previous paragraph, the authors had trained the network on significantly greater number of images than us. Also, the pedestrians and images generated from Pro-SiVIC are very similar even if they are from different scenarios. This might have led to insufficient learning for the network to predict proper uncertainties. To further test this hypothesis, we trained the network on the highway dataset used in SMILE II. However, the predictions were not satisfactory, and we need more work and data to test the Bayesian methods on the highway dataset.

From the above results, we conclude that more work is needed in building and testing Bayesian methods for semantic segmentation. If the network needs more data than a non-Bayesian network, this leads to further demands on data collection and management. An easier approach till we get better methods for this kind of problems, might be to use empirical methods like the one used in SMILE II. However, we believe these Bayesian methods can already applied for other type of problems that for example, time series analysis.

Apart from the analysis of above methods, we also developed some Heuristic rules that can work on top of the algorithm to get a better safety cage. As SMIRK was designed to include both radar and camera, the information from the radar could be used as an additional check point with regards to the trustability of the machine learning output. As SMIRK is an MVP, the rules are simple and few in number. But this provides a starting point towards combining ML output with heuristic rules. In the Table 2, we show the first version of the rules for SMIRK.

Table 2: A first version of the heuristic rules for SMIRK

Criteria	Rule	Comments
Size (number of pixels)	An object less than P x P pixels cannot be trusted for a decision. (The rule can be translated to specifying 2D angle to avoid dependence on camera resolution)	
Aspect ratio	An object other than the aspect ratio X±x :Y±y is not a pedestrian.	Should encompass or need special criteria for occluded pedestrians, children, person in a wheelchair etc.
Speed	An object moving faster than x km/h is not a pedestrian	The maximum value should encompass runners. The speed comes from radar.
Position on image (on road)	The pedestrian bounding box should lie in the "zone" depending on the position of the camera (depending on the distance). The "zone" contains the pixels in the image that cover roads and sidewalks.	

6.4.2 Safety cage software prototyping in SMIRK

SMIRK detects Out-Of-Distribution (OOD) input images as part of its safety cage architecture. Without losing generality, we decided to implement an OOD detection relying on the OSS third-party library Alibi Detect (https://github.com/SeldonIO/alibi-detect) from Seldon. Alibi Detect is a Python library that provides several algorithms for outlier, adversarial, and drift detection for various types of data [23]. For SMIRK, we trained Alibi Detect's autoencoder for outlier detection, with three convolutional and deconvolutional layers for the encoder and decoder respectively.

Alibi Detect's autoencoder for OOD detection has been trained on the training data subset of the development data. The encoder part is designed with three convolutional layers followed by a dense layer resulting in a bottleneck that compresses the input by 96.66%. The latent dimension is limited to 1,024 variables to limit requirements on processing VRAM of the GPU. The reconstruction error from the autoencoder is measured as the mean squared error between the input and the reconstructed instance. The mean squared error is used for OOD detection by computing the reconstruction error and considering an input image as an outlier if the error surpasses a threshold θ. The threshold used for OOD detection in SMIRK is 0.004, roughly corresponding to the threshold that rejects a number of samples that equals the amount of outliers in the validation set.

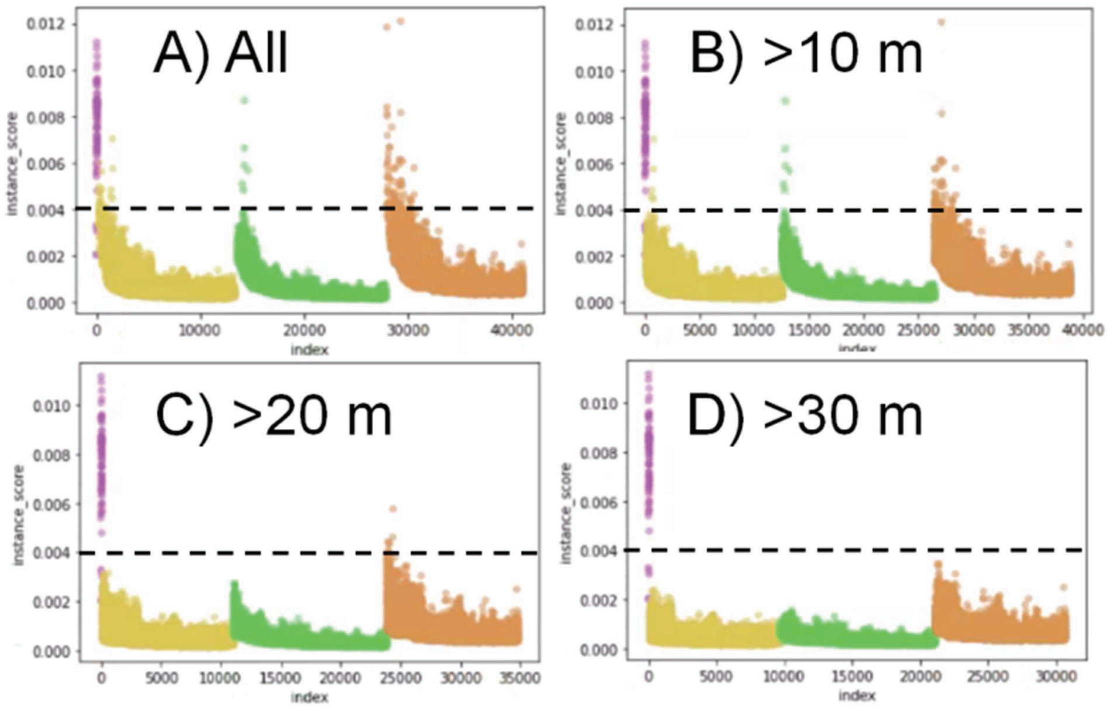


Figure 9: Reconstruction errors for different objects on the validation subset of the develop ment data at different distances from ego car (magenta=cylinder, yellow=female business casual, green=male business, orange=male casual). The dashed lines show the threshold for rejecting objects. In SMIRK, we use alternative B) in the safety cage.

During the iterative SMIRK development, it became evident that OOD detection using the autoencoder was inadequate at close range. Figure 9 shows reconstruction errors (on the y-axis) for all objects in the validation subset of the development data at A) all distances, B) > 10 m, C) > 20 m, and D) > 30 m. The visualization clearly shows that the autoencoder cannot convincingly distinguish the cylinders from the pedestrians at all distances (in subplot A), different objects appear above the threshold), but the OOD detection is more accurate when objects at close distance are excluded (subplot D) displays high accuracy). Based on validation of the four distances, comparing the consequences of the trade-off between safety cage availability and accuracy, the design decision for SMIRK's autoencoder is to only perform OOD detection for objects that are at least 10 m away. We explain the less accurate behaviour at close range by limited training data, a vast majority of images contain pedestrians at a larger distance – which is reasonable since the SMIRK ODD is limited to rural country roads

Furthermore, as the constrained SMIRK ODD ensures that only one single object appears in each scenario, the safety cage architecture applies the policy "once an anomaly, always an anomaly" – objects that get rejected once will remain anomalous no matter what subsequent frames might contain.

6.4.3 Safety cage and GAN attacks

6.4.3.1 Introduction

This sub-study investigated whether a Generative Adversarial Network (GAN) could be utilized to create data that is similar to the real data but different enough to be of interest in a safety cage from a validation perspective. Normally a GAN is used to generate new data or to modify existing data, for example to augment existing datasets or by removing glasses from a person or adding fog. A relatively new use-case is to use a GAN to generate “negative” data, i.e. it can be used for generation of data with the purpose of fooling a different model. By applying an Adversarial Attack (AA) framework with a GAN it is possible to generate images with the intent to fool a given classification, which would result in images that are similar but different enough for this purpose. For example, images with different weather effects or times of day could be generated.

The study used the Modified National Institute of Standards and Technology (MNIST) database [24] to generate data consisting of adversarial attack images to investigate if this could increase robustness. It was done using a GAN with a classifier trained towards adversarial attacks, following the Adversarial Transfer on Generative Adversarial Net (AT-GAN) framework. An adversarial attack is the act of purposefully feeding a Machine Learning model an input with the intent of making the model interpret the input wrong. For example, to manipulate a self-driving vehicles interpretation of a stop sign in order to make the vehicle ignore it. The AT-GAN framework aims to generate non-constrained adversarial examples from any input noise, with the purpose to broaden the diversity of the adversarial examples [25].

First, the GAN consisting of the generator and discriminator was trained on the MNIST dataset of handwritten digits to learn a good distribution of real data. Second, the classes 0 - 7, and 9 were targeted for attacks, using a pre-trained classifier. This resulted in a generator model that generates images aimed to fool the target classifier, training the classifier to identify adversarial attack images. The generated images can then be used in the verification and validation of the ML system. By making sure that the safety cage implementation reacts in the intended way to these not-quite-familiar images in the verification step.

6.4.3.2 Background

By training a GAN on a dataset it is possible to generate new synthetic data with this neural network, resulting in an increase in available data. There are limitations however, since the GAN is trained on a distribution of data it will only be able to generate data in that same scope. The synthetic data can thus not be used to increase the distribution of data.

A GAN works by defining two different networks against each other and having them compete to “fool” each other. The *Generator* network is tasked to generate data from a random noise vector and feed this generated data to the *discriminator* which will try to determine if the fed data is real or generated. Depending on the discriminators result either the generator or the discriminator “wins” and the weights of the losing net is updated.

The GAN itself is a “Auxiliary Classifier-Wasserstein Generative Adversarial Network with Gradient Penalty”. The Auxiliary Classifier enables the discriminator part of the GAN to classify the fed data in terms of what it represents in addition to discriminating between real and generated. In order to increase the training stability and chance of convergence the WGAN-GP uses gradient penalty instead of weight clipping to enforce the Lipschitz constraint: $|f(x_1) - f(x_2) | =< K|x_1-x_2|$. This means that the model is penalized if the gradient norm moves away from its target norm value of 1 [26].

6.4.3.3 Methodology and experiments

Several existing frameworks for GAN and AA were used in this study, along with the popular open-source dataset MNIST. In this section it is described what frameworks were used, why and how.

System

The environment in which this model was trained and tested is a docker container created with the TensorFlow GPU-Jupyter image running on a laptop with:

- Intel core i7-10850 @ 2.7GHz
- 32 GB RAM
- Nvidia Quadro P620 GPU

Software Setup

- GAN framework was developed by [27] and running on TensorFlow 1.15,
- Auxiliary Classifier and Separate AA framework for comparisons by [28], running on TensorFlow 1.15
- An external CNN for evaluation purposes, written with TensorFlow 2.
- Jupyter notebook used to visualize the results and calculate performance, running on TensorFlow 2.

The GAN by [27] consists of several different methods of attack, with the choice of different pre-trained classifiers and attack frameworks. The chosen classifier was the Madry-MNIST classifier as described in [28]. The Madry model is an open-source attack challenge which updates its available pre-trained models based on the best attack submitted to them. They supply both a natural and adversarially trained model as pre-trained for download.
These are compiled and run in a Docker container running TensorFlow 1.15. The training of the GAN ran for 50 epochs. The trained GAN is then ready for attack, this is run for as many epochs as it takes to generate x number of adversarial examples for the given classification. The AC-WGAN-GP model does this by generating a batch of data of the chosen source class and then feeding them to the chosen classifier. The images that manage to fool the classifier are saved as an attack. In this investigation 800 images were the target. This was done for all ten classifications of the MNIST dataset (integers 0-9), however the generation of attack examples on number 8 was terminated due to not converging. This resulted in almost 8000 images of adversarially generated images representing integers meant to fool the Madry classifier. A few of these images can be viewed in the results section below.

The Madry framework contains the functionality to do adversarial attacks by applying a perturbation on existing data [28]. This functionality is used to create 5000 AA examples to use in training the external classifier and for testing.

An external CNN classifier is used to evaluate the different images in an unbiased way. By training this classifier on both adversarial example images and regular data two differently trained models is available to evaluate on. The regular training is simply the MNIST dataset, while the adversarial training is done by switching 5000 of the training images with perturbation based adversarial images.
This classifier has the following architecture:

- Input: 28x28
- Layer1: Fully-Connected (128)
- Layer2: Fully-Connected (10) (out)
- The loss is calculated using Sparse Categorical Cross-entropy and using the stochastic optimizer Adam [29].

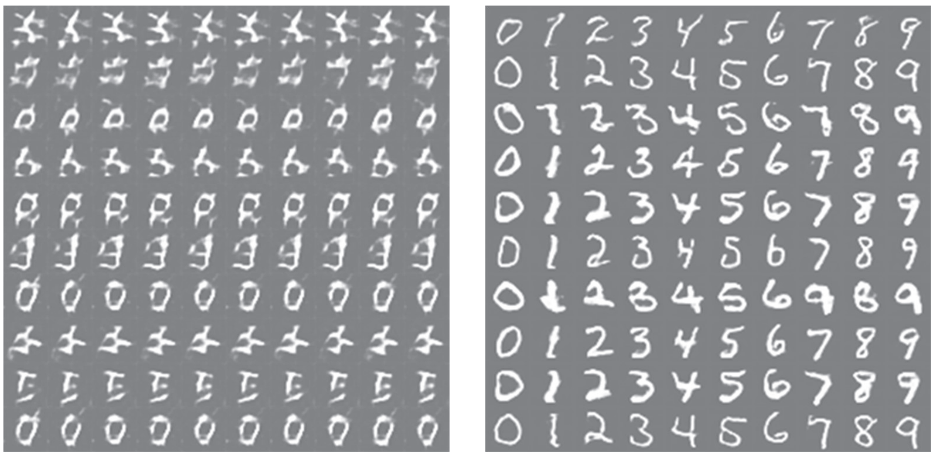


Figure 10: GAN generated realistic images

6.4.3.4 Results

Figure 10 illustrates the generated images from the GAN during training, left image is from the end of epoch 1, while the right image is after epoch 49.

After the GAN has finished training, it can be used for either targeted- or untargeted attacks, difference being if you want to attack a source class to a specific classification or not. Running untargeted attack on all ten classifications (excluding eight) of the MNIST dataset produces data as shown in Figure 11.

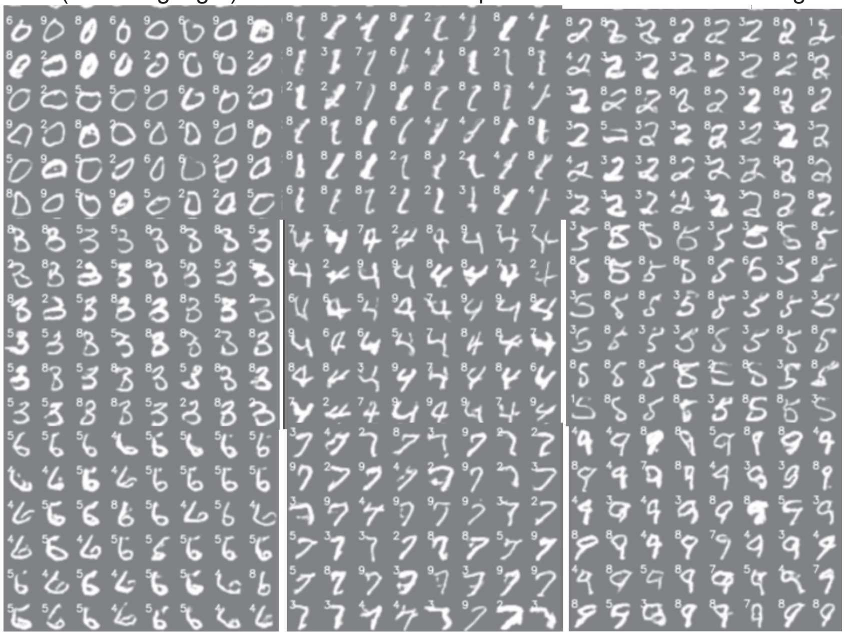


Figure 11: Untargeted attack on all classifications of MNIST dataset

The figures are sorted from zero to nine, excluding the number eight each containing 48 examples. The small number in the top left corner of each generated image represents the adversarial label. So, for example the top left most image has the source integer zero with a target of six, meaning that the goal was to fool the classifier to misread the zero as a six.

Table 3 highlights the performance of the external CNN model trained on different data. One training on the regular MNIST dataset, the other is MNIST with 10% of the images switched to adversarially generated images of MNIST digits. The performance is tested on verification data consisting of 1) Unaltered MNIST data, 2) Adversarially manipulated MNIST test data, 3) A mix of the two, 4) Generated Adversarial digits.

Table 3: Performance of CNN model trained on different dataset

	Accuracy	
Model training:	Adversarially trained	Regular
Image sets		
1. Regular test images:	0.96970	0.97300
2. AA test images:	0.97640	0.55160
3. Mixed test images:	0.96620	0.75610
4. Generated AA images:	0.87711	0.98687

Both models perform similarly on the regular MNIST test data, however when there are perturbation-based AA images mixed in the model trained on such data performs significantly better. 97.6% vs 55.2% and 96.6% vs 75.6% respectively. When feeding the images generated by the GAN, the regular model out-performs the AA trained one by 98.7% vs 87.7%.

6.4.3.5 *Discussion*

The results shows that it is possible to protect a system from adversarial attacks if the system is trained on data modified in the same way as the adversarial attack. The verification system shows an improvement in accuracy for both GAN-generated adversarial attacks and perturbation based adversarial attacked images. The study also shows that it is possible to generate simple images of numbers with GAN network, that can be used in adversarial attacks.

To protect the live system on the vehicle from adversarial attacks might not be of great importance, since the video feed comes directly from sensors on the vehicle. To generate training data for the system might however be of more interest. It is possible that it might be hard to acquire training data for all necessary situations, for example cyclists in rain. It will probably be interesting to generate those images, or possible modify existing training data. In further work, the technique in this study might be possible to expand to real world training data, which then can increase live world performance of autonomous vehicles.

The use of a GAN to generate synthetic data for verification purposes is an interesting proposition, which we believe warrants further work in this study. By using a GAN to generate data within the distribution and then using an attack framework to get unrestricted examples would be a good addition in the verification and validation. In order to get to that stage there is still work to be done.

In this study the MNIST dataset was used to evaluate the effectiveness of the theory, an adaptation of the Pro-SiVIC data and generation of images based on them would be of interest. With the GAN it would be possible to generate images representing a human but try to pass it off as a geometric figure or some other classification. In preparation for this a script has been written to extract the classification objects from the Pro-SiVIC data using the bounding boxes as reference. An example of this can be seen in the Figure 12.

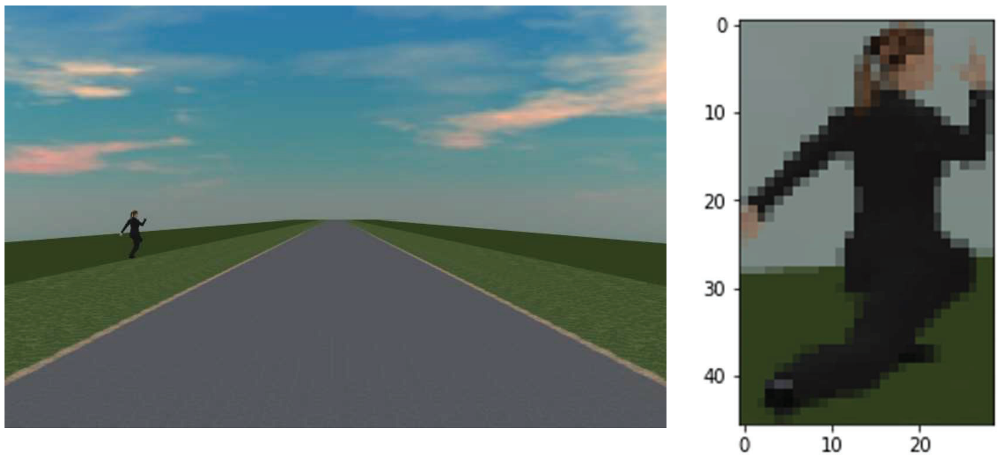


Figure 12: Synthetized pedestrian in Pro-SIVIC

6.5 WP5- Data management and V&V strategy

6.5.1 State of the Art in Data Management

6.5.1.1 *Background*

In the early stages of the SMILE III project a literature study was performed to gather knowledge on data management techniques as well as methods for performing Verification & Validation on Machine Learning-based systems. Multiple articles and literatures were explored, some providing input to the

project in terms of a Workshop on the topic and served as background for the *Data management strategy* and general input to the direction of the project.
The most relevant articles are summarized in this section, divided into the categories

- Verification & Validation
- Operational Design Domain
- SOTIF and ISO 26262
- Test Coverage & Simulation
- Metamorphic Testing, Fault Injection & Adversarial Attacks
- Data Management

As per defined by the Method of the project, knowledge acquisition for the literature study was supposed to be done not only through articles, but networking and conferences, something that was hindered by Covid-19.

6.5.1.2 Verification & Validation

In [30]. a review is done on what theoretically needs to be proved for AD, with emphasize on the difference to traditional software and states that a combination of methods required. [31] proposes such a method to tackle the possible guarantees of a Neural Network (adversarial examples, safety verification, output range analysis and robustness). [32] propose a framework (VeriVis) and provide "A formal framework for specifying ML safety properties" (including local, global safety) for verifying the safety properties of a network. Also, [33] describe how difficult V&V is for AI/ML-based systems (due to e.g., sensitivity to adversarial perturbations) by describing five challenges but also propose solutions to these in a formal methods manner.

6.5.1.3 Operational Design Domain

While A Framework for Automated Driving System Testable Cases and Scenarios presents a Taxonomy for the ODD, Towards an Operational Design Domain That Supports the Safety Argumentation of an Automated Driving System a framework is suggested to enable the mapping between Operational Design Domain (ODD) and Use-Cases (UC) and their Operation Conditions (OC).

6.5.1.4 SOTIF and ISO 26262

[34] (written before SOTIF was released) analyzes ISO 26262 and states that it is not enough for V&V on ML while proposing changes required to accommodate this. Also [30] is written before SOTIF but comments on acceptable residual risk.
Bridges the gap, Driver assistance systems: analysis, tests and the safety case. ISO26262 and ISO PAS 21448 presents a way to integrate the SOTIF lifecycle into the V-model of ISO 26262. In Analysis of Safety of The Intended Use (SOTIF), SOTIF is analyzed with goal of understanding how to maximize Area 1 (safe & known), and thus minimizing the unsafes (and more importantly the unknowns). Key Point 23 provides an approach on how to estimate (quantify) the residual risk.

6.5.1.5 Test Coverage & Simulation

[35] tests safety and neuron coverage (NC), and tries to use it as an argument for how to determine the quality of the network. NC is not a reasonable metric for V&V of Neural Networks. However, it says something about the diversity of the data and how well the Network utilized (for instance, a network where only 33% of the Neurals are activated from the (test) data, might be "too deep".) Similar conclusions are found in [36], where they state that "Neuron coverage is correlated with input-output diversity and can be used for systematic test generation." Combining these with the method in [37] could be a way of testing more efficient.
For more automated testing and simulations, the following articles and tools could offer a way into coverage and automation: [38] (describing a probabilistic programming language for well-distributed test over scenes), **OpenSCENARIO**/**OpenDrive**/**OpenCRG** (all three defines standardized file formatting for simulating real-world driving situations, is maintained and developed by ASAM) and the tool **Virtes** (An open platform for creating, configuring and animating virtual environments) and the project **Adaptive**.

6.5.1.6 Metamorphic Testing, Fault Injection & Adversarial Attacks
In [39] a framework (DeepRoad) is used to test Metamorphic Relations (MR). They display how changed weather conditions and MR can be used to test the network. They also explore input-validation in a way similar to the safety cage. Similar assumptions (weather conditions/distortions) are made in [40], where they state that the object under test should act sufficiently close with changed weather and/or lens distortion.
In [41] examples of MRs that should always hold (mostly concerning data, however) but more importantly state that the "effectiveness of MT depends on the quality of identified MR", insinuating that both people with domain and technical knowledge are needed in order to write reasonable requirements (MR). On top of this they state that a "violation of a MR is the equivalent of a defect/bug in ordinary software code". The same reasoning is found in [42](defects in code), but they also present four MR/MT for testing a CNN. "Fault injection" is used to test the thresholds of the function. In [43] they state that "The point of doing […] fault injection is not to validate functionality, but rather to probe for weak spots that might be activated via unforeseen circumstances."
[44] discuss other possible performance issues (such as the impact of changed camera angle) but also methods to "protect" the ML-model against it. In [14] a tool is proposed (Deep Learning Verification) which they guarantee to find adversarial examples (if they exist). Propagates the network and analyze layer-by-layer. On the same note, [45] proposes another tool with the aim to trick the network by generating adversarial examples (their method is based on a "two player game"). They claim a network is theoretically "safe from adversarial attacks" when no examples can be found, and state that their solution is not network specific.

6.5.1.7 Data Management
In [46]**,** the authors present an API for testing the quality of one Database. The methods for analyzing the data should be of interest (the API itself is not). They also present methods to add new data to existing and making sure distributions etc. are the same. In [47] seven suggestions on how/what to test in the data (and three other aspects) is presented. A Test Score Table is suggested to put a comparable score on a Machine Learning system.
[48] is a survey, which (like the above article) divides the Machine Learning system/lifecycle into four parts. Digging into the data section, the desiderata is explained with aspects such as relevant, complete, balance and accurate.
In [30] a formal requirements are made on the Volume, Coverage of known (critical) scenarios and Minimization of unknown (critical) scenarios data. In [43] it's stated that requirements take the form of training data "that enumerates a set of input values and correct system outputs": The non-safety-related data (general driving data) and specification on what "purely safety requirements that completely and unambiguously define what 'safe' is". And state that "[…] the validation data must be independent and diverse from the training data in every way except the desired features, or else overfitting will not be detected during validation."
[34] analyzes ISO26262 (written before SOTIF): "[…] the complete specification requirement must be relaxed. Partial specifications can be required, where possible. For example, if a pedestrian must be less than 9 feet tall, then this property can be used to filter out false positives. Such properties can be incorporated into the training process or checked on models after training".
In [49] one of their two contributions is Dataset Generator, on which it's said: "Our picture generator can generate large data sets for which the diversity of the pictures can be controlled by the user. This overcomes a lack of training data, one of the limiting problems in training of CNNs. Also, a target synthesized dataset can be used as a benchmark for a specific domain of application."

6.5.2 Data management strategy

6.5.2.1 Background
The data management (DM) strategy explored within SMILE III aims at creating a structured approach of generating and handling data for V&V of ML based system to be used for a safety-critical applications. The DM strategy is derived from Research Question 3. The strategy has been built upon the findings of

the previous SMILE projects and a literature study on how to ensure robustness and safety in safety-critical machine learning models. **Error! Reference source not found.** shows how data management and how the data flow is connected with other topics in the project.

6.5.2.2 Prerequisites

The data management strategy builds upon the assumption that appropriate activities have been done, such as Hazard Identification, Fault Tree Analysis and similar per Functional Safety standards (ISO 26262) [2] and Safety Of the Intended Functionality (ISO/PAS 21448) [3]. The resulting requirements on the Function and System (SMIRK) formulated as need to consist of System Requirements, Operational Design Domain (ODD), Machine Learning (ML) component functionality and safety requirements. From these the requirements on data, ML data requirements, can be formulated and will guide collection and processing.

6.5.2.3 Expansion of the DM strategy

SMILE III DM strategy will describe how to use Pro-Sivic to generate data to test how well the NN can generalize and detect hazards using a safety cage. Using the output of the safety cage, that is placed around the artificial Neural Network (NN), corner cases can be detected. The corner cases will be saved in a data bank. The information given by the data bank can then be used to investigate if the ML data requirements are properly defined. Once the requirements fulfil the operational design domain (ODD), the corner cases in the data bank will be used to fine-tune the NN [28]. The process is repeated iteratively until the NN achieves a satisfactory magnitude of some specific metrics.

6.5.2.4 Outline

Each part of this section is built on the following format: introduction on the topic, practical implementation and relation and consequence to the safety of the safety-critical function of the SMILE III project.

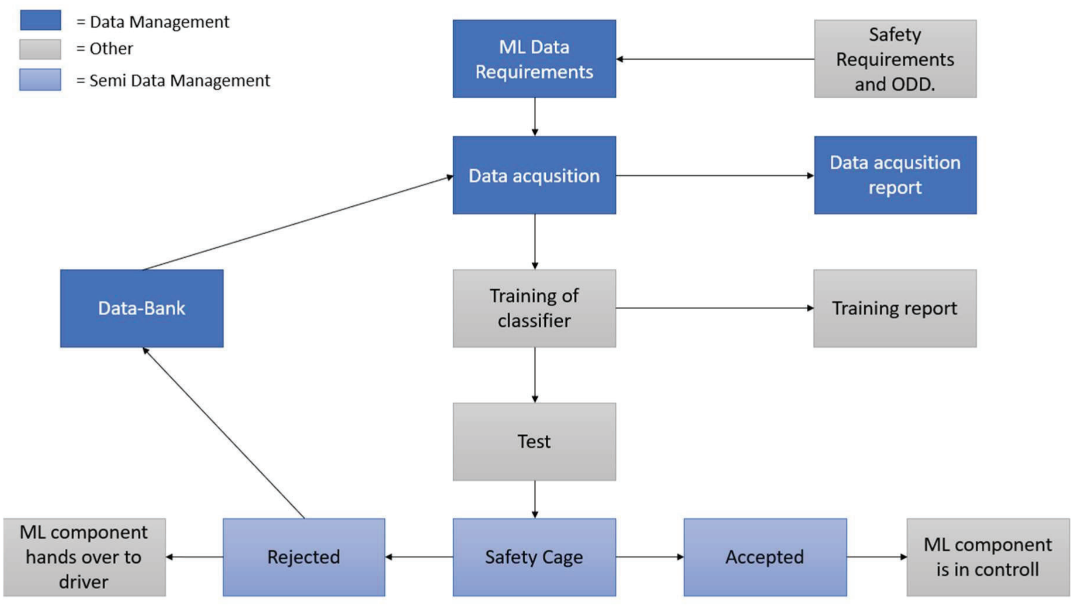


Figure 13: Data Management Flow

6.5.2.5 Machine Learning Data Requirements

Introduction: The requirement on the data used for primary training, validating and testing the Machine Learning component is an important step to properly represent the problem formulation as features in the data. The representation is thus a translation and a description, from system requirements and ODD, to metrics desired in the data.

Practical: The ML data requirements must be formulated in such a way that it contains information on what data is to be collected. The data must contain information that is relevant to the current ODD. The requirements on data must be specified in a manner that every requirement is measurable, as a metric. Thus, Representation, Quantity, Distribution and Quality of the data are few general metrics possible to use as a foundation.

- Specification - When and where is the ego vehicle driving? Can simulated data be used as a representation of real-world data?
- Quantity - Is the amount of data enough to (fully) represent the problem?
- Quality - Is the data representing useful features and of the desired quality?
- Distribution - Are classes represented according to desired distribution?
- Representation - Is the data of the desired type (Camera Images, LiDAR, Radar)?

Relation and consequence: The ML data requirements must depend on the safety requirements. By ensuring this, one can strengthen the safety argumentation. With measurable metrics in the requirements of data, structured argumentation, repetition of experiments, analysis of faults and properties of data can be performed. Further on, enforcing a structured way of answering these above-mentioned metrics, no vital data-related safety requirements are missed. As the safety requirements in SMILE III are influenced by ISO/PAS 21448 (SOTIF)[3], the ML data requirements will be an extension to this and can be argued to follow the SOTIF standard. The implementation of a justification report increases the possibility to repeat the work in the future.

6.5.2.6 Data Acquisition

How data is collected, analyzed, cleaned and preprocessed will be discussed here. The relation and consequence of safety for the safety-critical ML component will be discussed. Argumentation for following the suggested activities will be supplied.

6.5.2.6.1 Data Collection

Introduction: Collection of data for ADAS can be done using sensors attached to a car that drives in everyday traffic. It can also be collected from simulators, simulating and extending the initial dataset. In SMILE III data will be collected in Pro-Sivic, a simulation tool that is manufactured by ESI. The main benefit of using simulators is the possibility to have labelled data out of the box and better control over what kind of data is collected, and by extension generating examples that rarely occur in real-world data. Data will be collected into three datasets, training, validation and test data. These datasets will be used during different times in the development process.

Practical: In SMILE III, primarily simulated data is used. The simulation will produce data according to the requirements, formulated in section 2. It is commonly known in the Machine Learning community that it is a big advantage to have a lot of data. This is where Quantity from the previous section can be used. The quantity of the data must be sufficient. Once data is produced, it should be split into the training and validation datasets. How one should divide data into the two different sets is not trivial, however, a common split is 70-85% training and 30-15%. What is of more importance is that the sub-datasets' (training/validation/test) metrics should mimic the metrics describing the entire dataset according to the metrics described in 2. When dividing data into the different sets the internal structure must be maintained. Separately from the above-described datasets (used in the development of the components and subsystem), another dataset should be collected by a test team. This dataset should be collected on the formulated requirements, as training/test datasets, but kept separate to maintain its impartialness. Additionally, this dataset should contain a significant amount of cases that are known to be hard for the system, corner and edge cases, to classify correctly. During and after the collection of data the information that is needed to reproduce the data must be stored in a data generation log. For example, it can be parameters on cameras that are simulated in Pro-Sivic or specifications of certain pedestrians that are generated in the simulation.

Relation and Consequence: By dividing data into different parts of data sets as described above it is ensured that all the different data sets have the same internal relation of data types. This is important since the ML algorithm must be trained on data that is derived from the ML data requirements. Followed by this it is as important to have similar, not identical, train and test data sets. When producing the verification dataset separately it is an action to prevent bias in the verification data. Bias is preferably avoided at the verification staged of training a neural network. If a team of developers and testers can avoid bias they have a stronger claim for safety in the developed product. The safety argumentation for the Data Management strategy in SMILE III is supported by following the suggested way of data collection. The procedure is heavily supported by the work done by R. Hawkins et.al. [50]. The data generation log provides another level of argumentation for robustness, reproduction ability and safety.

6.5.2.6.2 Data cleaning and preprocessing

Introduction: The data collected according to specification represents an interpretation of the formal requirements, but might still contain data not suitable for the task at hand. Outliers, points statistically observed far from the rest of the data set, are to be excluded as these often indicate extreme observations or wrongful measurements and might skew the overall data set. As data sets are combined, multiple observations of the same case might occur, which by definition will skew the distribution. With a dataset cleaned from abnormalities such as outliers, duplicated and irrelevant data, different preprocessing techniques might be needed to ensure the quality of the data. The level of sophistication required for preprocessing depends on the type of issue at hand, and which part of the process it affects. Normalization, the practice of standardizing all points of the dataset to a common scale, is used as a technique to improve the function's performance during the training phase. E.g., for machine learning the approximation of a function is based on patterns and the relative, rather than the absolute, value of an input signal. Augmentation of data is the process of expanding the dataset. Adding noise to existing data to increase robustness to noise, making the dataset a more general, or complete, representation of the problem. Practical: As duplications might occur when combining different sets, the same possibility exists of collecting irrelevant data: points collected as a consequence of where the relevant data points are collected. As for Outlier and Duplicated data, Irrelevant points effect statistical values (such as distribution and skew) and thus the data set's representation of the problem. In order to ensure the safety of the target function, the dataset must be under control. Thus, observations unrelated to the function's task (based on the requirements and ODD) should not exist. Further, duplicated data might impact the "learning process" of the function, as certain classes might be favoured, undesired, in the data, and must be excluded. From section 2, quality is referred to as a measure of data. When cleaning the data, it is done with the aim of increase the quality of the data. As different inputs might be of different units and scales, normalization of these might speed up learning. It is thus not a safety-critical measure. Normally, min-max scaling is applied, where the points are fit is done to fit the range [0, 1]). Structural issues might occur, where a signal with numerical (e.g. an integer) values is represented by some other type (e.g. as a string) and is needed to be converted to the desired representation. This might, in some cases, be in the interest of safety as control of the input is highly relevant for a learning algorithm. Closely related to structural issues is missing data, where e.g. the measurement of a continuous signal has stopped and NaN points are recorded instead, for a limited period. To make the data representative, again, several approaches may be taken: the samples can be discarded entirely, interpolated with the surrounding points, substituted with a predetermined value (often either 0 or the mean of the instance). Some caution is recommended, as this will affect the function. For instance, for a signal with binary measurements, the interpolation might be 0, 1 or 0.5, where the latter does not represent anything useful, making the dataset more unsafe and representative than before preprocessing.

The final remark for preprocessing of data and datasets is the handling of missing labels and verification of these. The label of the data is the foundation for classification, both internally and e.g. a learning algorithm: Data is typically defined by its label and is usually what is used to cluster different points. When it comes to missing labels, both manual and automatic assignment may be done: the more safety-critical the application to use the data is, the more manual labelling[51] is recommended and in these cases by persons with domain knowledge. On the other hand, automatic assignment of labels require either a sophisticated algorithm (with preconceptions and bias) or a trained Machine Learning model (where explainability is missing) and manual work is needed as well. Missing labels is not a trivial problem to handle, and we dissuade from using such data, to the extent possible. Verification of data labels is of high priority, and require to some extent personnel with domain knowledge to ensure the safety associated with feeding a training algorithm with wrongful labels. Augmentation can be done in different ways: adding noise to existing input data to increase robustness when the dataset is somewhat complete, generating semi-synthetic data based on existing to bridge the gaps or generating fully synthetic data from a simulation of the real world.

Relation and consequences: Assuring the collected data has desired structure, type and being free from outliers, irrelevant and duplicated data is required to ensure the safety of further development. As discussed above, outliers might create wrongful connections in a Neural Network while missing labels (or wrongful) changes the learned target. Both create uncertainty for a problem that is already complex. The level of risk associated with augmentation activities are in increasing order: while adding noise to current data might be representing something slightly different than the collected, it could still be said to represent somewhat in the domain of the original data. Fully synthetic data (or entire datasets), however, have to be carefully analyzed to ensure its validity. In the example of a Perception Model, the original dataset may be images collected in the real-world, while a simulator generates synthetic data where the limits of photorealism are in the hand of the tool. Thus, an image generated in a simulator might appear to contain real-world features to the human eye, while the encoding inside the Perception Model tells another story.

6.5.2.6.3 Analysis

Introduction: Analysis is required to guide aspects of collection and augmentation, e.g. to ensure there is an appropriate class balance within the dataset [48]. This means that one needs to make adequate changes to the dataset that ensures it is complete, accurate, have class balance and is relevant.

Practical: During analysis, the data set needs to be looked through. A good way would be to extract information about the dataset in a programming environment. There, it would be possible to order data types in certain ways in an efficient way. From section 2, it is given that the data must have the right representation for the task at hand. During analysis, the representation of data must be analysed. The analysis stage is thereby set and if done correctly it is possible to estimate the risk that the dataset brings with it. It is during the analysis of the dataset that it is possible to have real insight into what kind of data is included in the dataset. If it here shows that data does not live up to the ML data requirements, it must be resolved and documented.

Relation and consequence: To get an overview of the dataset is very helpful when assuring safety in the system. The analysis procedure will give important insight into how and if the data is full filling the ML data requirements. As mentioned earlier, this makes it possible to estimate the risk of the data that it will inevitably bring to the system.

6.5.2.7 Simulation and testing

Introduction: The essence of SMILE is to add a safety cage in critical applications. An example of this can be:

Ego car has a braking system that is controlled by a machine learning algorithm is an example of a safety-critical application. The ML algorithm is trained to recognize possible hazardous with data from sensors and cameras as input which is placed on ego car. The safety cage will either accept or decline the classification result from the ML component that handles the brakes. If it accepts it, the algorithm will have the control and do the necessary maneuvers to avoid accidents. If the safety cage declines the result from the ML component, it will let go of the control over the brakes and hand the responsibility to the driver.

Safety cage (Practical): In SMILE III, one can assume that the safety cage must act as an indicator of how well images, in this case, are classified. The safety cage can mediate this with different metrics. Such as classification score, AUROC, or anomaly score. By analyzing this output from a classified data type, one can determine if the NN is generalizing ”good enough” or not. Good enough will be a design parameter, usually defined as a threshold and determined by the safety owner/tester.
By implementing the safety cage, the outcome can be any of the three for a given simulated scenario:

- Alternative 1: Rejected based on anomaly score. The data type is classified with a metric value that does not fulfil the requirements to be regarded as safe/certain. That data type will be rejected to the data bank instance.
- Alternative 2: Accepted based on anomaly score. The data is classified with a metric value that fulfils the requirements to be regarded as safe/certain. This type of data will be accepted and not be sent to the data bank instance.
- Alternative 3: Rejected based on miss-classification. The data is classified with a metric value that fulfils the requirements to be regarded as safe/certain, however, the simulation provides that the situation is a false positive/false negative (FP/FN). This means that the model is certain of something that actually is false. This type of classification errors must be detected and the NN should be fine-tuned on this as well. The data type that generated such behaviour must be sent to the data bank instance.

Table 4: Data handling based on safety cage results

Good metric score	Bad metric score	FP/FN
Accept	Reject to use in databank	Reject to use in databank

Relation and consequence: The safety cage is a central part of SMILE-III. Besides its original gaining, it also gives a clear description of what type of data is considered "unsafe". This type of data lands under the category Known and hazardous, also known as area 3 in SOTIF [52]. This additional knowledge on how the classifier performs must be used. If it is used properly, a safer system can be produced. The rejected data will be guided to a databank. This will be covered in the next section.

6.5.2.8 Databank

Introduction: The databank is to be viewed as another data set. But it will only contain data that was rejected by the safety cage. The aim of the data bank is to use the data to fine-tune the classifier with data that it was not performing well on. By doing this one ensures the reduction of cases that are represented in area 3 in SOTIF [52]. This section will contain suggestions on how one can optimize the usage of information in the databank. Data in the databank will consequentially go through the data acquisition phase.

Practical: Data that is sent to the databank will need to be prepossessed, analysed and cleaned in the same way as the original data set was according to section 6.5.2.6.2 before it is used to fine-tune the classifier. To make the fine-tuning more efficient Generative Adversarial Network (GAN) is to be used to produce new data points that are very much alike the original data point. This will make it possible to produce a lot of data points from one rejected data point. It is also suggested that GAN can be used to produce small pixel changes that can have a large impact on how the model performs. By fine-tuning the model with such data, it can have a positive effect in terms of robustness against adversarial attacks. The changes done to the rejected data must be documented in the same way as mentioned in section 6.5.2.6.2.

Relation and consequence: In an ideal case, the previously described steps will happen automatically, which means that the ML component gets more and more reliable continuously. In SMILE, this will not be possible since data will be stored offline. However, for a non-automatic system, the implementation of a databank will increase the chance to build a safe and robust ML component.
Documentation of the underlying steps and results will make it possible to repeat the construction of the ML component. In ISO/PAS 21448 (SOTIF) it is specified that the ML pipeline should make it possible to transform known and hazardous scenarios in area 3 into area 2. Area 2 represents known and safe scenarios. The usage of rejected data in the databank supports SOTIF and will make the system more safe, robust, and repeatable.

6.5.2.9 Discussions

The DM strategy in SMILE III is presented and focused has been on strengthening the safety argumentation, robustness, and repeatability. To ensure this, it is suggested that the ML data requirements are documented properly. Data acquisition ensures a good dataset that the NN will be trained on if the ML data requirements have been defined and interpret correctly. Documentation of data acquisition is necessary and the act of doing so will make the system more repeatable. A description of how the safety cage can work in SMILE is presented to give context to the data management activities related to the safety cage. Rejected data from the safety cage will be sent to a databank, this will then be used to improve the system and the ML component.

6.5.2.10 Relation and Consequence

Data Requirements: The data requirements are modelled from and must depend on the safety requirements. Hence, the quality of the data requirements and the legitimacy of its safety argumentation is highly dependent on the incoming Safety Requirements and ODD. By ensuring this, one can strengthen the safety argumentation.
Along with the requirements on data, requirement metrics, structured argumentation, gathering plan, analysis of faults and properties of data are to be formulated. Further on, enforcing a structured way of answering these above-mentioned metrics, no vital data-related safety requirements are missed.
As the safety requirements in SMILE III are influenced by ISO/PAS 21448 (SOTIF), the data requirements will be an extension to this and can be argued to follow the SOTIF standard. The implementation of a justification report increases the possibility to repeat the work in the future.

Data Acquisition: Gathering data according to a predetermined plan and from the formulated requirements with the measurable metrics guarantees the data follows the general safety requirements. Implied, training-, validation- and testing-datasets should all contain desired qualities individually and in whole. However, in order to prevent requirement bias in the data, the verification dataset should be produced/gathered separately from the rest. As the requirements and the gathering plan by default will be interpreted differently and thus contain the bias of the gathering team, ensuring verification data gathering being performed separate minimize the risk of bias. If the team of developers and the team of testers can avoid this bias they have a stronger claim for safety in the developed product.
The safety argumentation for the Data Management strategy in SMILE III is supported by following the suggested way of data collection. The procedure is heavily supported by the work done by R.Hawkins et.al. [50]. The data generation log provides another level of argumentation for robustness, reproduction ability and safety.

Assuring the collected data has desired structure and type (defined by requirements) while being free from outliers, irrelevant and duplicated data ensure the safety of further development. As outliers might create wrongful connections in a Neural Network and missing labels (or wrongful ones) changes the learned target, both create uncertainty for a problem which is already complex.
The level of risk associated with augmentation activities are in increasing order: while adding noise to current data might be representing something slightly different than the collected, it could still be said to represent somewhat in the domain of the original data. The amount of noise, deviation, should be specified in data requirements. Fully synthetic data (or entire datasets), however, must be carefully analyzed to ensure its validity. In the example of a Perception Model, the original dataset may be images collected in the real-world, while a simulator generates synthetic data where the limits of photorealism are in the hand of the tool. Thus, an image generated in a simulator might appear to contain real-world features to the human eye, while the encoding inside the Perception Model tells another story.
For further argumentation and control of data, an overview is needed. Whether a report or visualization tool is used, assuring safety in the data comes from a clear overview. A structured analysis gives an important insight in how and if the data is full-filling data requirements. This also allows for the possibility to estimate the risk of using certain data and datasets and how the mixing of different datasets bring uncertainties to the system.

Simulation and testing: The safety cage is a central part of SMILE-III. Besides the trivial benefits of it (providing a fail-safe), it may also give a clear description of what type of data is considered "unsafe". This type of data lands under the category Known and hazardous, also known as area 3 in SOTIF [52]. This allows for expanding datasets and/or limiting the ODD and further the safety requirements. This additional knowledge on how the classifier performs can be used to improve not only the data but also the process, as it should make clear areas and scenarios not fully represented in the data. The rejected data will be guided to a databank, where further analysis may be done.

Databank: In an ideal case, placement of rejected data and automatic analysis of it is performed. Throughout the development process, this means the ML component continuously becomes more reliable. In SMILE III, this is not done practically as the data is stored offline. However, theoretically, the implementation of a databank will increase the chance to build a safe and robust ML component. Documentation of the under-laying steps and results, as well as a well-kept database and -bank, makes it possible to repeat the construction of the ML component. In ISO/PAS 21448 (SOTIF), the ML pipeline is specified to make it possible to transform known and hazardous scenarios in area 3 into area 2. The usage of rejected data as an improvement to an immature system supports SOTIF and will make the system more safe, robust and repeatable.

6.6 WP6-Autonomous platform

A Go-Kart platform (Autonomous platform) has been developed as another research platform besides the software simulation platform. It is ready for data collection, integration and testing of the algorithms, approaches proposed by SMILE. This platform will be used as part of the V&V process for the subsequent research projects of the SMILE series. The Go-Kart platform is illustrated in Figure 14. A video demonstrating the platform is also available at: https://youtu.be/m7VoXZ-J48U

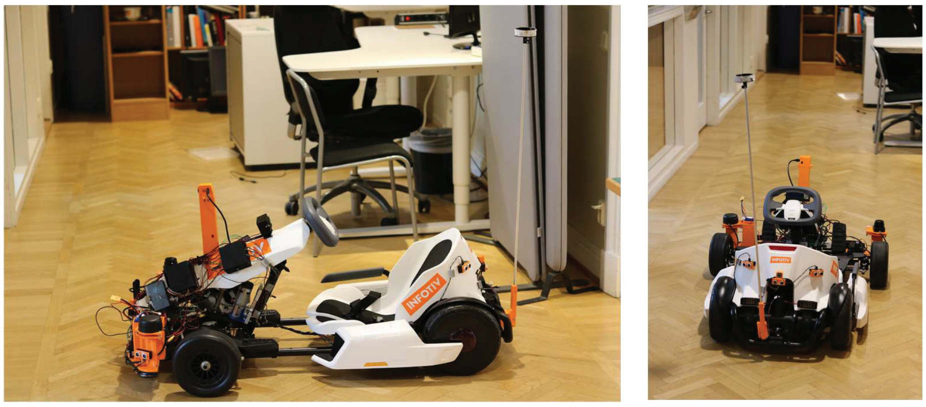


Figure 14: Go-kart autonomous platform, from side and from behind

The autonomous platform collects in realtime data captured from its sensors and publishes the information to different ROS topics. Data from the following sensors are available:

- LiDAR (Figure 15)
- Camera (Figure 16)
- IMU (gyro, accelermometer, magnetometer)
- Wheel speed sensor
- Sonar sensor
- Go-kart state output
- Go-kart driving outputs/requests

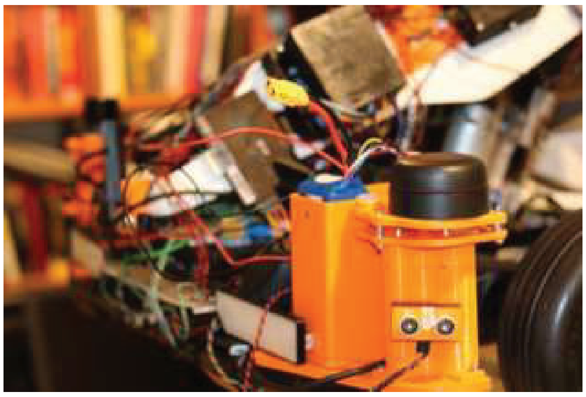


Figure 15: Lidar sensor that is mounted on a flat stand

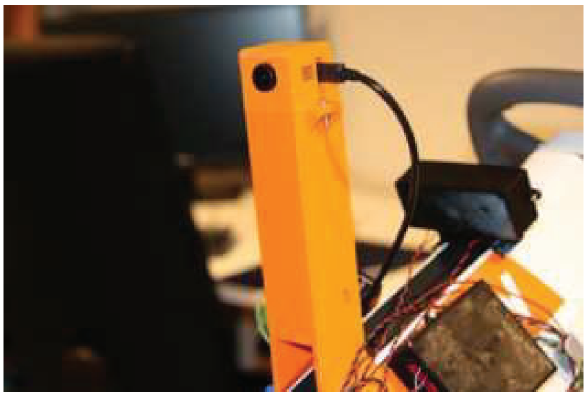
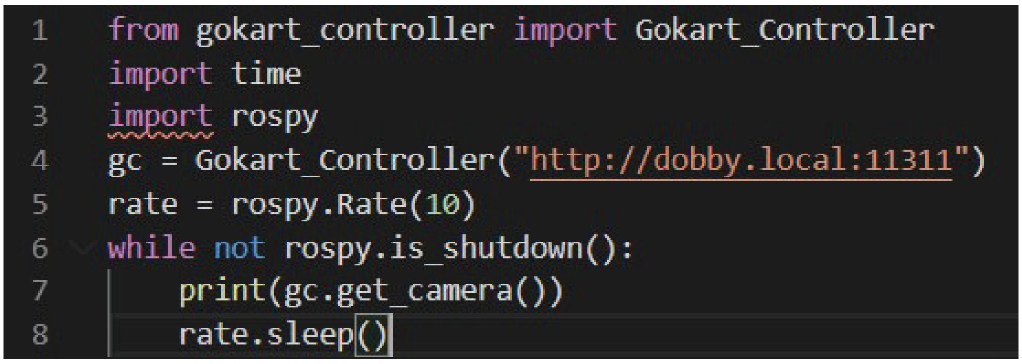


Figure 16: Go-kart camera sensor, mounted on a 3D-printed stand

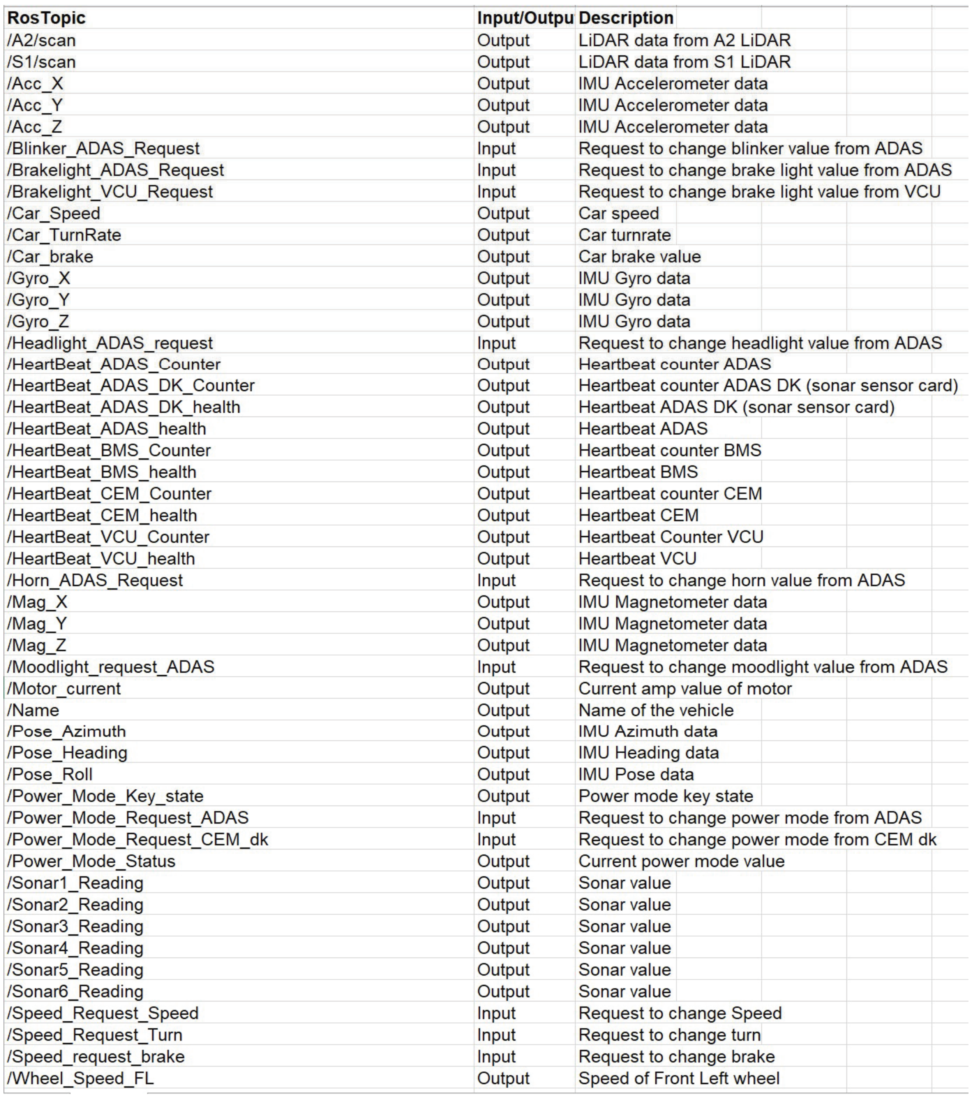
A ROS architecture is implemented on TX2, together with a simplified ROS python interface and SMILE node (Figure 17). Besides the realtime data communication, data can also be recorded to create the V&V datasets.



```python
from gokart_controller import Gokart_Controller
import time
import rospy
gc = Gokart_Controller("http://dobby.local:11311")
rate = rospy.Rate(10)
while not rospy.is_shutdown():
    print(gc.get_camera())
    rate.sleep()
```

Figure 17: Implemented ROS python interface

Figure 18 shows the current available list of data topics implemented in the Go-Kart's ROS component.



RosTopic	Input/Outpu	Description
/A2/scan	Output	LiDAR data from A2 LiDAR
/S1/scan	Output	LiDAR data from S1 LiDAR
/Acc_X	Output	IMU Accelerometer data
/Acc_Y	Output	IMU Accelerometer data
/Acc_Z	Output	IMU Accelerometer data
/Blinker_ADAS_Request	Input	Request to change blinker value from ADAS
/Brakelight_ADAS_Request	Input	Request to change brake light value from ADAS
/Brakelight_VCU_Request	Input	Request to change brake light value from VCU
/Car_Speed	Output	Car speed
/Car_TurnRate	Output	Car turnrate
/Car_brake	Output	Car brake value
/Gyro_X	Output	IMU Gyro data
/Gyro_Y	Output	IMU Gyro data
/Gyro_Z	Output	IMU Gyro data
/Headlight_ADAS_request	Input	Request to change headlight value from ADAS
/HeartBeat_ADAS_Counter	Output	Heartbeat counter ADAS
/HeartBeat_ADAS_DK_Counter	Output	Heartbeat counter ADAS DK (sonar sensor card)
/HeartBeat_ADAS_DK_health	Output	Heartbeat ADAS DK (sonar sensor card)
/HeartBeat_ADAS_health	Output	Heartbeat ADAS
/HeartBeat_BMS_Counter	Output	Heartbeat counter BMS
/HeartBeat_BMS_health	Output	Heartbeat BMS
/HeartBeat_CEM_Counter	Output	Heartbeat counter CEM
/HeartBeat_CEM_health	Output	Heartbeat CEM
/HeartBeat_VCU_Counter	Output	Heartbeat Counter VCU
/HeartBeat_VCU_health	Output	Heartbeat VCU
/Horn_ADAS_Request	Input	Request to change horn value from ADAS
/Mag_X	Output	IMU Magnetometer data
/Mag_Y	Output	IMU Magnetometer data
/Mag_Z	Output	IMU Magnetometer data
/Moodlight_request_ADAS	Input	Request to change moodlight value from ADAS
/Motor_current	Output	Current amp value of motor
/Name	Output	Name of the vehicle
/Pose_Azimuth	Output	IMU Azimuth data
/Pose_Heading	Output	IMU Heading data
/Pose_Roll	Output	IMU Pose data
/Power_Mode_Key_state	Output	Power mode key state
/Power_Mode_Request_ADAS	Input	Request to change power mode from ADAS
/Power_Mode_Request_CEM_dk	Input	Request to change power mode from CEM dk
/Power_Mode_Status	Output	Current power mode value
/Sonar1_Reading	Output	Sonar value
/Sonar2_Reading	Output	Sonar value
/Sonar3_Reading	Output	Sonar value
/Sonar4_Reading	Output	Sonar value
/Sonar5_Reading	Output	Sonar value
/Sonar6_Reading	Output	Sonar value
/Speed_Request_Speed	Input	Request to change Speed
/Speed_Request_Turn	Input	Request to change turn
/Speed_request_brake	Input	Request to change brake
/Wheel_Speed_FL	Output	Speed of Front Left wheel

Figure 18: Available ROS data topics in Go-Kart

7 Dissemination and publications

7.1 Dissemination

How has / is the project result to be used and disseminated?	Marked	Comments
Increase knowledge in a specific area	X	SMILE III improves the SMILE research environment with new researchers, improved current research competences of its partners and established a research infrastructure including open sources, process descriptions, and open datasets.

Be passed on to other advanced technological development projects	X	SMIRK is a research prototype under development that facilitates research on V&V of safety-critical systems embedding ML components. SMIRK responds to calls for a fully transparent ML-based ADAS to act as a system-under-test in research on trusted AI. SMIRK and its safety case is available on GitHub under an open source license.
Be passed on to product development projects		
Introduced to the market		
Used in investigations, regulations, permit matters/political decisions.		

7.2 Publications

SMILE III resulted in a number of publications, talks, tutorials as listed in the following subsections.

7.2.1 Scientific publications and conferences

- J. Henriksson, “On Improving Validity of Deep Neural Networks in Safety Critical Applications,” Licentiate thesis, Chalmers University of Technology, 2020. Accessed: Apr. 26, 2022. [Online]. Available: https://research.chalmers.se/en/publication/517219
- M. Borg, R. B. Abdessalem, S. Nejati, F.-X. Jegeden, and D. Shin, “Digital Twins Are Not Monozygotic – Cross-Replicating ADAS Testing in Two Industry-Grade Automotive Simulators,” in *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, Apr. 2021, pp. 383–393. doi: 10.1109/ICST49551.2021.00050.
- C. Englund, E. E. Aksoy, F. Alonso-Fernandez, M. D. Cooney, S. Pashami, and B. Åstrand, “AI Perspectives in Smart Cities and Communities to Enable Road Vehicle Automation and Smart Traffic Control,” *Smart Cities*, vol. 4, no. 2, pp. 783–802, 2021, doi: 10.3390/smartcities4020040.
- M. Borg *et al.*, “Exploring the Assessment List for Trustworthy AI in the Context of Advanced Driver-Assistance Systems, in *Proc. of the 2nd Workshop on Ethics in Software Engineering Research and Practice* (SEthics), 2021. Preprint: http://arxiv.org/abs/2103.09051
- H. Ebadi, M. H. Moghadam, M. Borg, G. Gay, A. Fontes, and K. Socha, “Efficient and Effective Generation of Test Cases for Pedestrian Detection - Search-based Software Testing of Baidu Apollo in SVL,” in *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, Aug. 2021, pp. 103–110. doi: 10.1109/AITEST52744.2021.00030.
- M. Borg *et al.*, “Ergo, SMIRK is Safe: A Safety Case for a Machine Learning Component in a Pedestrian Automatic Emergency Brake System,” *arXiv:2204.07874 [cs]*, Apr. 2022, Accessed: Apr. 26, 2022. [Online]. Available: http://arxiv.org/abs/2204.07874 (under journal review)
- J. Henriksson, C. Berger, M. Borg, L. Tornberg, S. R. Sathyamoorthy, and C. Englund, “Performance analysis of out-of-distribution detection on trained neural networks,” *Information and Software Technology*, vol. 130, p. 106409, Feb. 2021, doi: 10.1016/j.infsof.2020.106409.
- Q. Song, M. Borg, E. Engström, H. Ardö, and S. Rico, “Exploring ML testing in practice – Lessons learned from an interactive rapid review with Axis Communications.” To appear in Proc. of the 1st International Conference on AI Engineering – Software Engineering for AI, 2022. Preprint: https://arxiv.org/abs/2203.16225
- M. H. Moghadam *et al.*, “Performance Testing Using a Smart Reinforcement Learning-Driven Test Agent,” *2021 IEEE Congress on Evolutionary Computation*, Jul. 2021. Available: http://www.es.mdh.se/publications/6271-
- M. H. Moghadam, M. Borg, and S. J. Mousavirad, “Deeper at the SBST 2021 Tool Competition: ADAS Testing Using Multi-Objective Search,” *2021 IEEE/ACM 14th International*

Workshop on Search-Based Software Testing, May 2021. Available: http://www.es.mdh.se/publications/6270-
- M. H. Moghadam, M. Borg, M. Saadatmand, S. J. Mousavirad, M. Bohlin, and B. Lisper, "Machine Learning Testing in an ADAS Case Study Using Simulation-Integrated Bio-Inspired Search-Based Testing," Mar. 2022. [Online]. Available: http://www.es.mdh.se/publications/6398- (under journal review)
- M. H. Moghadam, M. Saadatmand, M. Borg, M. Bohlin, and B. Lisper, "An autonomous performance testing framework using self-adaptive fuzzy reinforcement learning," *Software Qual J*, vol. 30, no. 1, pp. 127–159, Mar. 2022, doi: 10.1007/s11219-020-09532-z.

7.2.2 *Talks and tutorials*

- Course "Trained, not coded – Toward Safe AI in the Automotive Domain", 6th International School on Software Engineering (ISE School 2020, the Free University of Bozen-Bolzano, Italy)
- Quest4Quality's free webinar (Infotiv)
- "Using Search-Based Software Testing to Guide the Strive for Robust Machine Learning Components - Lessons Learned Across Systems and Simulators in the Mobility Domain" Keynote at the 6th Int'l Workshop on Testing Extra-Functional and Quality Characteristics of Software Systems, Apr 4, 2022.
- Trained, Not Coded - Beauty in Software 2.0
- Beauty in Code Conference, Malmö, Mar 7, 2020.
- 9th Software Technology Exchange Workshop, Swedsoft, Virtual, Jan 22, 2021.
- Science! by Infotiv, Virtual, Apr 8, 2021.
- SMIRK, Scandinavian Conference on System and Software safety, 2022

8 Conclusions and future research

Safe and trustworthy AI is a key enabler to increase the level of vehicle automation. Several automotive standardization initiatives are ongoing to allow safety certification for DML in road vehicles, including e.g., ISO 21448 SOTIF. However, standards provide high-level requirements that must be operationalized in each development context. Unfortunately, there is a lack of publicly available ML-based automotive demonstrator systems that can be used to study safety case development.

Within this project, a major scientific contribution is SMIRK, a PAEB designed for operation in the industry-grade simulator ESI Pro-SiVIC, available on GitHub under an OSS license. SMIRK uses a radar sensor for object detection and an ML-based component relying on a DNN for pedestrian recognition. Originating in SMIRK's minimalistic ODD, we present a complete safety case for its ML-based component by following the AMLAS framework. To the best of our knowledge, this work constitutes the first complete application of AMLAS independent from its authors. We conclude that even for a very restricted ODD, the size of the ML safety case is considerable, i.e., there are many aspects of the AI engineering that must be clearly explained.

In the project, with its industry-academia collaboration, we report several lessons learned. First, using a simulator to create synthetic data sets for ML training particularly limits the validity of the negative examples. Second, the complexity of object detection evaluations necessitates internal training within the project team. Third, composing the fitness function used for model selection is a delicate engineering activity that forces explicit tradeoff decisions. Fourth, what parts of an image to send to an autoencoder for OOD detection is an open question – for SMIRK, we stretch the content of bounding boxes to a larger square.

The project results and findings (primarily represented by SMIRK) will enable future research in this direction:

- Study the efficient approaches to conduct safety assurance for extended ODDs. The open questions include how one can derive and agree with an open standard for ODD definition and how to build trustworthy ODDs as a base to communicate between smart traffic participants.
- Exploration of the use of SMIRK as test benchmark
- Further study the data specification and AI explainability in providing evidence to support the safety arguments

9 Participating parties and contact persons

Organization	Contact person	Logo
RISE Research Institutes of Sweden AB Lindholmspiren 3A 417 56 Göteborg	Thanh Hai Bui, Markus Borg	RI. SE
Semcon Lindholmsallén 2 417 80 Göteborg	Jens Henriksson	semcon
Infotiv AB Västra Hamngatan 8 411 17 Göteborg, Sweden	Martin Karsberg	INFOTIV INSTANT INNOVATION
ESI Nordics Bror Nilssons gata 16 SE-417 55 Göteborg	Erik Abenius	esi
Combitech AB Lindholmspiren 3A, 417 56 Göteborg	Gustaf Bengström	COMBITECH
QRTECH AB Flöjelbergsgatan 1C SE-431 35 Mölndal	Sankar Raman Sathyamoorthy	QRTECH an EMBRON Company

10 References

[1] M. Borg, "Explainability First! Cousteauing the Depths of Neural Networks to Argue Safety," *Explainable Software for Cyber-Physical Systems (ES4CPS), Report from the GI Dagstuhl Seminar 19023*, pp. 26–27, 2019.

[2] International Organization for Standardization, "ISO 26262-1: Road vehicles — Functional safety," *ISO*, 2018. http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/83/68383.html (accessed Nov. 18, 2019).

[3] International Organization for Standardization, "ISO/PAS 21448: Road Vehicles-Safety of the Intended Functionality," *ISO*, Jan. 2019. http://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/09/70939.html (accessed Nov. 12, 2019).

[4] K. R. Varshney, R. J. Prenger, T. L. Marlatt, B. Y. Chen, and W. G. Hanley, “Practical Ensemble Classification Error Bounds for Different Operating Points,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 11, pp. 2590–2601, Nov. 2013, doi: 10.1109/TKDE.2012.219.

[5] S. Ramos, S. Gehrig, P. Pinggera, U. Franke, and C. Rother, “Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1025–1032. doi: 10.1109/IVS.2017.7995849.

[6] S. Han *et al.*, “EIE: Efficient Inference Engine on Compressed Deep Neural Network,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, Jun. 2016, pp. 243–254. doi: 10.1109/ISCA.2016.30.

[7] K. Heckemann, M. Gesell, T. Pfister, K. Berns, K. Schneider, and M. Trapp, “Safe Automotive Software,” in *Knowledge-Based and Intelligent Information and Engineering Systems*, Berlin, Heidelberg, 2011, pp. 167–176. doi: 10.1007/978-3-642-23866-6_18.

[8] R. Adler, P. Feth, and D. Schneider, “Safety Engineering for Autonomous Vehicles,” in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W)*, Jun. 2016, pp. 200–205. doi: 10.1109/DSN-W.2016.30.

[9] A. Knauss, J. Schröder, C. Berger, and H. Eriksson, “Paving the roadway for safety of automated vehicles: An empirical study on testing challenges,” in *2017 IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2017, pp. 1873–1880. doi: 10.1109/IVS.2017.7995978.

[10] B. Huval *et al.*, “An Empirical Evaluation of Deep Learning on Highway Driving,” *arXiv:1504.01716 [cs]*, Apr. 2015, Accessed: Nov. 18, 2019. [Online]. Available: http://arxiv.org/abs/1504.01716

[11] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[12] B. Spanfelner, D. Richter, S. Ebel, U. Wilhelm, and C. Patz, “Challenges in applying the ISO 26262 for driver assistance systems,” 2012.

[13] A. Abdulkhaleq, S. Wagner, and N. Leveson, “A Comprehensive Safety Engineering Approach for Software-Intensive Systems Based on STPA,” *Procedia Engineering*, vol. 128, pp. 2–11, Jan. 2015, doi: 10.1016/j.proeng.2015.11.498.

[14] X. Huang, M. Kwiatkowska, S. Wang, and M. Wu, “Safety Verification of Deep Neural Networks,” in *Computer Aided Verification*, Cham, 2017, pp. 3–29. doi: 10.1007/978-3-319-63387-9_1.

[15] “VALU3S | Verification and Validation of Automated Systems’ Safety and Security.” https://valu3s.eu/ (accessed Dec. 05, 2020).

[16] “Assessment List for Trustworthy Artificial Intelligence (ALTAI) for self-assessment | Shaping Europe’s digital future.” https://digital-strategy.ec.europa.eu/en/library/assessment-list-trustworthy-artificial-intelligence-altai-self-assessment (accessed Oct. 31, 2021).

[17] M. Borg *et al.*, “Ergo, SMIRK is Safe: A Safety Case for a Machine Learning Component in a Pedestrian Automatic Emergency Brake System,” *arXiv:2204.07874 [cs]*, Apr. 2022, Accessed: Apr. 26, 2022. [Online]. Available: http://arxiv.org/abs/2204.07874

[18] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv:1804.02767 [cs]*, Apr. 2018, Accessed: May 10, 2019. [Online]. Available: http://arxiv.org/abs/1804.02767

[19] J. Henriksson, C. Berger, and S. Ursing, “Understanding the Impact of Edge Cases from Occluded Pedestrians for ML Systems.” arXiv, 2022. doi: 10.48550/ARXIV.2204.12402.

[20] A. Kendall and Y. Gal, “What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?,” *arXiv:1703.04977 [cs]*, Oct. 2017, Accessed: Apr. 25, 2022. [Online]. Available: http://arxiv.org/abs/1703.04977

[21] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning,” in *Proceedings of The 33rd International Conference on Machine Learning*, Jun. 2016, pp. 1050–1059. Accessed: Apr. 30, 2022. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html

[22] F. Kraus and K. Dietmayer, “Uncertainty Estimation in One-Stage Object Detection,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, Auckland, New Zealand, Oct. 2019, pp. 53–60. doi: 10.1109/ITSC.2019.8917494.

[23] J. Klaise, A. Van Looveren, C. Cox, G. Vacanti, and A. Coca, “Monitoring and explainability of models in production,” *arXiv:2007.06299 [cs, stat]*, Jul. 2020, Accessed: Apr. 30, 2022. [Online]. Available: http://arxiv.org/abs/2007.06299

[24] L. Deng, “The mnist database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
[25] X. Wang, K. He, C. Song, L. Wang, and J. E. Hopcroft, “AT-GAN: An Adversarial Generator Model for Non-constrained Adversarial Examples.” arXiv, 2019. doi: 10.48550/ARXIV.1904.07793.
[26] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved Training of Wasserstein GANs.” arXiv, 2017. doi: 10.48550/ARXIV.1704.00028.
[27] Y. Song, R. Shu, N. Kushman, and S. Ermon, “Constructing Unrestricted Adversarial Examples with Generative Models.” arXiv, 2018. doi: 10.48550/ARXIV.1805.07894.
[28] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, “Towards Deep Learning Models Resistant to Adversarial Attacks,” *arXiv:1706.06083 [cs, stat]*, Sep. 2019, Accessed: Apr. 27, 2022. [Online]. Available: http://arxiv.org/abs/1706.06083
[29] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, 2014. doi: 10.48550/ARXIV.1412.6980.
[30] S. Burton, L. Gauerhof, and C. Heinzemann, “Making the Case for Safety of Machine Learning in Highly Automated Driving,” in *Computer Safety, Reliability, and Security*, Cham, 2017, pp. 5–16. doi: 10.1007/978-3-319-66284-8_1.
[31] W. Ruan, X. Huang, and M. Kwiatkowska, “Reachability analysis of deep neural networks with provable guarantees,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, Jul. 2018, pp. 2651–2659.
[32] K. Pei, Y. Cao, J. Yang, and S. Jana, “Towards Practical Verification of Machine Learning: The Case of Computer Vision Systems,” *arXiv:1712.01785 [cs]*, Dec. 2017, Accessed: Apr. 29, 2022. [Online]. Available: http://arxiv.org/abs/1712.01785
[33] S. A. Seshia, D. Sadigh, and S. S. Sastry, “Towards Verified Artificial Intelligence,” *arXiv:1606.08514 [cs]*, Jul. 2020, Accessed: Feb. 25, 2022. [Online]. Available: http://arxiv.org/abs/1606.08514
[34] R. Salay, R. Queiroz, and K. Czarnecki, “An Analysis of ISO 26262: Machine Learning and Safety in Automotive Software,” Apr. 2018. doi: https://doi.org/10.4271/2018-01-1075.
[35] Y. Sun, X. Huang, D. Kroening, J. Sharp, M. Hill, and R. Ashmore, “Testing Deep Neural Networks,” *arXiv:1803.04792 [cs]*, Apr. 2019, Accessed: Apr. 29, 2022. [Online]. Available: http://arxiv.org/abs/1803.04792
[36] Y. Tian, K. Pei, S. Jana, and B. Ray, “DeepTest: automated testing of deep-neural-network-driven autonomous cars,” in *Proceedings of the 40th International Conference on Software Engineering*, New York, NY, USA, May 2018, pp. 303–314. doi: 10.1145/3180155.3180220.
[37] E. Rocklage, H. Kraft, A. Karatas, and J. Seewig, “Automated scenario generation for regression testing of autonomous vehicles,” in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct. 2017, pp. 476–483. doi: 10.1109/ITSC.2017.8317919.
[38] D. J. Fremont *et al.*, “Scenic: a language for scenario specification and data generation,” *Machine Learning*, Feb. 2022, doi: 10.1007/s10994-021-06120-5.
[39] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, “DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems,” in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2018, pp. 132–142. doi: 10.1145/3238147.3238187.
[40] Z. Q. Zhou and L. Sun, “Metamorphic testing of driverless cars,” *Commun. ACM*, vol. 62, no. 3, pp. 61–67, Feb. 2019, doi: 10.1145/3241979.
[41] J. Ding, X. Kang, and X.-H. Hu, “Validating a Deep Learning Framework by Metamorphic Testing,” in *2017 IEEE/ACM 2nd International Workshop on Metamorphic Testing (MET)*, May 2017, pp. 28–34. doi: 10.1109/MET.2017.2.
[42] A. Dwarakanath *et al.*, “Identifying implementation bugs in machine learning based image classifiers using metamorphic testing,” in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, New York, NY, USA, Jul. 2018, pp. 118–128. doi: 10.1145/3213846.3213858.
[43] P. Koopman and M. Wagner, “Challenges in Autonomous Vehicle Testing and Validation,” *SAE Int. J. Trans. Safety*, vol. 4, no. 1, Art. no. 2016-01–0128, Apr. 2016, doi: 10.4271/2016-01-0128.
[44] I. Goodfellow, P. McDaniel, and N. Papernot, “Making machine learning robust against adversarial inputs,” *Commun. ACM*, vol. 61, no. 7, pp. 56–66, Jun. 2018, doi: 10.1145/3134599.

[45] M. Wicker, X. Huang, and M. Kwiatkowska, “Feature-Guided Black-Box Safety Testing of Deep Neural Networks,” *arXiv:1710.07859 [cs]*, Feb. 2018, Accessed: Apr. 30, 2022. [Online]. Available: http://arxiv.org/abs/1710.07859

[46] S. Schelter, D. Lange, P. Schmidt, M. Celikel, F. Biessmann, and A. Grafberger, “Automating large-scale data quality verification,” *Proc. VLDB Endow.*, vol. 11, no. 12, pp. 1781–1794, Aug. 2018, doi: 10.14778/3229863.3229867.

[47] E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, “The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction,” 2017.

[48] R. Ashmore, R. Calinescu, and C. Paterson, “Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges,” *arXiv:1905.04223 [cs, stat]*, May 2019, Accessed: Jul. 08, 2020. [Online]. Available: http://arxiv.org/abs/1905.04223

[49] T. Dreossi, S. Ghosh, A. Sangiovanni-Vincentelli, and S. A. Seshia, “Systematic Testing of Convolutional Neural Networks for Autonomous Driving,” *arXiv:1708.03309 [cs]*, Aug. 2017, Accessed: Apr. 30, 2022. [Online]. Available: http://arxiv.org/abs/1708.03309

[50] R. Hawkins, C. Paterson, C. Picardi, Y. Jia, R. Calinescu, and I. Habli, “Guidance on the Assurance of Machine Learning in Autonomous Systems (AMLAS),” *arXiv:2102.01564 [cs]*, Feb. 2021, Accessed: Sep. 22, 2021. [Online]. Available: http://arxiv.org/abs/2102.01564

[51] Y. Roh, G. Heo, and S. E. Whang, “A Survey on Data Collection for Machine Learning: A Big Data - AI Integration Perspective,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1328–1347, Apr. 2021, doi: 10.1109/TKDE.2019.2946162.

[52] E. Schwalb, “Analysis of Safety of The Intended Use (SOTIF),” 2019.