SMIRK is a pedestrian automatic emergency braking system that facilitates research on safety-critical systems embedding machine learning components. As a fully transparent driver-assistance system, SMIRK can support future research on trustworthy AI systems, e.g., verification & validation, requirements engineering, and testing. SMIRK is implemented for the simulator ESI Pro-SiVIC with core components including a radar sensor, a mono camera, a YOLOv5 model, and an anomaly detector. ISO/PAS 21448 SOTIF guided the development, and we present a complete safety case for a restricted ODD using the AMLAS methodology. Finally, all training data used to train the perception system is publicly available.
publicly available training set for the ML model and a complete safety case for its ML component [3]. We posit that SMIRK can be used for various types of research on trustworthy AI as defined by the European Commission, i.e., AI systems that are lawful, ethical, and robust.

SMIRK is an ML-based Advanced Driver-Assistance System (ADAS) that provides Pedestrian Automatic Emergency Braking (PAEB) in the industry-grade simulator ESI Pro-SiVIC. SMIRK’s perception system uses input from a radar sensor and a forward-facing mono camera. As an ML-based AI system example, SMIRK interweaves source code and data into a software system [4]. We implemented the source code in Python and generated data to train (1) a YOLOv5 model for object detection and (2) an anomaly detector. SMIRK was developed as part of the Swedish research project SMILE III.1

2. SMIRK architecture

The development of SMIRK followed the process defined in SOTIF [2], i.e., iterative development and safety engineering toward acceptable risks. For a description of the engineering process, we refer readers to the publication describing the safety case [3]. This section presents the logical and process views of the SMIRK architecture and an overview of the ML components. Further details are available in the System Architecture Description on GitHub.

Fig. 1 presents the logical view. SMIRK interacts with three hardware components simulated in ESI Pro-SiVIC: a radar unit, a camera, and the simulated ego car. Moreover, SMIRK consists of standard Python source code and components trained using the PyTorch ML framework. A key safety mechanism in the safety argumentation is the “Safety Cage” in the center of the figure.

The overall flow in the SMIRK process view is as follows: (1) The radar detects an object. (2) The Radar Logic calculates the time-to-collision (TTC). If TTC < 4 s, the Perception Orchestrator (PO) is notified. (3) PO requests and forwards a camera image to the Pedestrian Recognizer (PR). (4) PR returns True if a pedestrian is found. (5) PO forwards information to the Uncertainty Manager (UM) in the safety cage. (6) UM forwards the image to the Anomaly Detector (AD). (7) AD analyses if the image is out-of-distribution (OOD), i.e., does not resemble the training data, and returns a verdict. (8) If True, UM rejects the input and does not propagate a brake signal. If False, the Rule Engine performs a sanity check based on laws of physics. (9) If UM remains confident that collision with a pedestrian is imminent, the signal to perform PAEB propagates to ego car.

The SMIRK architecture encompasses two OSS ML components. First, the PR uses the third-party framework YOLOv5 by Ultralytics implemented using PyTorch. YOLO is an established single-stage real-time object detection algorithm first presented by Redmon et al. [5], optimized for real-time applications through fast inference and the concept “You Only Look Once”. SMIRK’s PR uses the YOLOv5 architecture without any modifications. Second, the AD in the safety cage is an autoencoder provided by the third-party library AlibiDetect2 by SeldonIO. Autoencoders consist of an encoder part that maps input data to a latent space of fewer dimensions and a decoder that attempts to reconstruct the original data. Intuitively, if input that resembles the training data is processed by the autoencoder, the reconstruction error will be smaller than for outlier data — this approach, with a carefully selected threshold, is used for OOD detection in SMIRK.

3. Safety assurance

Demonstrating compliance to a safety standard such as SOTIF requires providing a safety case along with the software. A safety case is a structured argumentation that a system is safe, backed up by evidence. Independent safety assessors scrutinize safety cases before certifying that a system adheres to one or several standards. Providing holistic safety cases necessitates the creation of numerous artifacts, e.g., requirements, design documentation, test specification, inspection protocols, and test results. Moreover, providing traceability from the requirements to all downstream artifacts is vital to maintaining a chain of evidence.

The Hazard Analysis and Risk Assessment (HARA) for SMIRK revealed two categories of hazards. First, SMIRK might miss pedestrians and fail to commence emergency braking. Second, SMIRK might commence emergency braking when it should not, i.e., SMIRK might result in false positives. The first hazard can be fatal, but since SMIRK is an ADAS, the controllability is high, i.e., the driver can brake the car — false negatives are not safety-critical from the SOTIF perspective. The second hazard, however, involves no controllability, i.e., false positives can result in dangerous rear-end collisions. Mitigation of the second hazard included the introduction of a safety mechanism to minimize false positives, i.e., the safety cage providing OOD detection as described in Section 2.

SMIRK’s safety assurance adheres to the methodology Assurance of Machine Learning for use in Autonomous Systems (AMLAS) developed by the University of York [6]. AMLAS provides an overall process organized into six stages that mandate the creation of 34 individual artifacts that, together with a corresponding safety argumentation, constitutes the safety case. As prescribed by AMLAS, we use Goal Structuring Notation (GSN), a graphical argument, to document and present proof that SMIRK has achieved its safety goals. The following AMLAS artifacts are available in the GitHub repository,3 organized per type:

---

1. https://tinyurl.com/smileiiii

---

Fig. 1. The logical view of the SMIRK architecture.
• GSN argument patterns: {ML Assurance Scoping, ML Safety Requirements, ML Data Argument, ML Learning, ML Verification, ML Deployment} Argument Pattern
• Arguments: {ML Safety Assurance Scoping, ML Safety Requirements, ML Data, ML Learning, ML Verification, ML Deployment} Argument
• Requirements: System Safety Requirements, Description of Operating Environment of System, System Description, ML Component Description, Safety Requirements Allocated to ML Component, ML Safety Requirements, Data Requirements, Operational Scenarios
• Inspection results: ML Safety Requirements Validation Results, Data Requirements Justification Report
• Test results: {ML Data Validation, Internal Test, ML Verification, Integration Testing} Results
• Data sets: {Development, Internal Test, Verification} Data
• Logs: {Data Generation, Model Development, Verification, ErrorNeous Behaviour} Log
• Models: ML Model

Compared to conventional systems, a considerable difference in the safety engineering of ML-based systems is that data must be treated as a first-class software constituent. For SMIRK, we organize all data requirements according to the assurance-related desiderata proposed by Ashmore et al. [7], i.e., we ensure that the data set is relevant, complete, balanced, and accurate. Since SMIRK is developed for simulated environments, the SMIRK data collection relies on generating data and pixel-level semantic segmentation in ESI Pro-SiVIC. All data generation is script-based and fully reproducible. In total, we provide a data set corresponding to 4928 execution scenarios with pedestrians and 200 execution scenarios with OOD examples — corresponding to roughly 185 GB of image data, further described on AI Sweden’s dedicated hosting site.5

4. Impact overview

The SMIRK project goal is to assist the driver on country roads in rural areas by performing emergency braking in the case of an imminent collision with a pedestrian. The level of automation offered by SMIRK corresponds to SAE Level 1 — Driver Assistance, i.e., “the driving mode-specific execution by a driver assistance system of either steering or acceleration/deceleration”. We designed SMIRK with evolvability in mind; thus, future versions might include steering and thus correspond to SAE Level 2.

The project goal of the SMIRK development, as part of the research project SMILE III, was twofold. First, the project team benefited substantially from having a concrete example of ADAS development as a basis for discussions. We learned how challenging it is to perform safety case development for ML-based perception systems by practically doing it — nothing can substitute the experience of a hands-on engineering effort. Second, we wanted to provide a completely open research prototype that the community can use as a case under study in future research projects on AD engineering, automated driving, and safety engineering. While previous academic papers on automotive ML safety have provided vital contributions in requirements engineering, safety argumentation, and testing, we believe that SMIRK is the first to connect the fragmented pieces into a holistic demonstrator.

SMIRK has proved useful in communicating ML safety concepts. We have been invited to give talks at conferences and research groups in Sweden and abroad. Moreover, RISE has successfully used SMIRK as a complementary running example when providing external SOTIF training. We are confident that SMIRK can be used to facilitate research and education on various topics — a selection of five promising avenues for future projects follows.

ML testing is a rapidly moving field in software engineering that introduces new levels of testing [8]. Identifying systems that balance the needs of relevance and simplicity is an ever-present challenge in software testing research. SMIRK allows researchers to explore data testing, ML model testing, integration testing, and system testing since data sets, ML model architectures, and the source code are publicly available. For example, offline model testing can be compared to online system testing, as Haq et al. recently proved important [9]. Concrete test techniques that could be evaluated using SMIRK include search-based software testing, metamorphic testing, fuzz testing, neural network test adequacy assessments, and testing for explainable AI.

An equally active field concerns simulator development for the automotive industry. We have previously shown that cross-simulator experiments for ESI Pro-SiVIC and Siemens/TASS PreScan can lead to considerably different insights during system testing [10]. With an increasing number of industry-grade simulators available, e.g., CarMaker, RoadRunner, and Beam.NG, and increasingly mature OSS alternatives, e.g., CARLA and SVL, we call for additional comparative research. We believe that SMIRK can facilitate such research, as it should be fully portable to other simulators.

In safety engineering research, SMIRK can support research into several related aspects. The concept of dynamic safety cases to support the recertification of modified systems is a popular research topic. Along these lines, we believe that studies on evolving the SMIRK safety case after expanding the operational design domain beyond the currently simplistic restrictions described would be rewarding. For example, what would be needed to argue that SMIRK is safe for roads with curvature or different weather conditions? Other topics related to safety engineering that SMIRK facilitates include software traceability, functional modifications according to SOTIF, regression testing, change impact analysis, and compliance to other existing and emerging standards such as UL4600 and ISO 4804.

We developed SMIRK and its ML pipeline in tandem. The entangled nature of data and source code in ML-based systems necessitates ML pipeline automation, motivated by the CACE principle coined by Google researchers: “Changing Anything Changes Everything”. [11]. The industrial phenomenon of MLOps, adapting DevOps ideas and CI/CD for the ML era, is gradually becoming an academic research target. We argue that the current SMIRK pipeline can be used to study ML pipelines and the MLOps phenomenon. Furthermore, novel tools can be integrated into the pipeline as they become available, e.g., solutions for ML experiment tracking and data version control. The SMIRK ML pipeline is still under active development and will be included as a part of the next release.

Finally, SMIRK can be used to study and evaluate new ML models for computer vision. Many research studies evaluate object detection models on data sets containing independent single frames. Using SMIRK, researchers can study both single-frame accuracy, series of frames, and implications on the system level in the simulator. SMIRK’s YOLOv5 model can easily be replaced by other models and data scientists could explore feature engineering to tune models thanks to the ML pipeline. Analogously to the object detection model, alternative OOD detection approaches could be integrated into SMIRK for systematic evaluation [12].

5. Conclusion and future work

We present SMIRK, an ML-based PAEB developed for the simulator ESI Pro-SiVIC. The source code is available on GitHub under an OSS license, and the data set used to train and test the ML components is available. Furthermore, we present SMIRK with a fully transparent safety case for its ML component, including requirements, design documentation, test specifications, inspection protocols, and test results. Although the current version of SMIRK is designed for

5 https://www.ai.se/en/data-factory/datasets/smirk-synthetic-pedestrians-dataset
a highly restricted operational design domain in ESI Pro-SiVIC, our original software publication illustrates the variety of software artifacts needed to provide a holistic ML safety argumentation using the AMLAS methodology. The rich supplementing documentation makes SMIRK a helpful starting point for safety engineering research and education. The SMIRK system opens up avenues for research on software testing, cross-simulator experimentation, and computer vision. Finally, the ML pipeline that co-evolved with SMIRK can be used to facilitate MLOps research for cyber–physical systems.

CRediT authorship contribution statement

Kasper Socha: Software, Data curation, Investigation, Formal analysis, Validation, Writing – review & editing. Markus Borg: Conceptualization, Methodology, Validation, Resources, Writing – original draft, Supervision, Project administration, Funding acquisition. Jens Henriksson: Software, Investigation, Formal analysis, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Our thanks go to everyone in the SMILE III project who helped with developing SMIRK’s corresponding safety case. In particular we acknowledge Thanh Bui, Olof Lennartsson, Elias Sonnsoj Lönegren and Sankar Raman Sathyamoorthy. Furthermore, we thank François-Xavier Jegeden for providing advanced technical ESI Pro-SiVIC support. This work was carried out within the SMILE III project financed by Vinnova, FFI, Fordonstrategisk forskning och innovation under the grant number 2019-05871 and partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, Sweden. Finally, we thank AI Sweden for showcasing the SMIRK data set and helping interested users to download it.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.simpa.2022.100352.

Illustrative examples

Demonstrator video available as a supplementary file. SMIRK is in operation on the right side, braking for pedestrians and rejecting non-pedestrian objects.

References