EVS38
Göteborg, Sweden, June 15-18, 2025

AI Safety Assurance in Electric Vehicles: A Case Study on AI-Driven SOC Estimation

Martin Skoglund[1], Fredrik Warg[1], Aria Mirzai[1], Anders Thorsén[1], Karl Lundgren[1],
Peter Folkesson[1], Bastian Havers-Zulka[1]
[1]{*martin.skoglund, fredrik.warg, aria.mirzai, anders.thorsen, karl.lundgren, peter.folkesson, bastian.havers-zulka*}*@ri.se*
RISE Research Institutes of Sweden, Borås, Sweden

Executive Summary

Integrating Artificial Intelligence (AI) technology in electric vehicles (EV) introduces unique challenges for safety assurance, particularly within the framework of ISO 26262, which governs functional safety in the automotive domain. Traditional assessment methodologies are not geared toward evaluating AI-based functions and require evolving standards and practices. This paper explores how an independent assessment of an AI component in an EV can be achieved when combining ISO 26262 with the recently released ISO/PAS 8800, whose scope is AI safety for road vehicles. The AI-driven State of Charge (SOC) battery estimation exemplifies the process. Key features relevant to the independent assessment of this extended evaluation approach are identified. As part of the evaluation, robustness testing of the AI component is conducted using fault injection experiments, wherein perturbed sensor inputs are systematically introduced to assess the component's resilience to input variance.

1 Introduction

ISO 26262 [1] is the international standard for functional safety (FuSA) in road vehicles, focusing on reducing risks from system malfunctions through structured processes and safety mechanisms. For automated driving/advanced driver assistance systems (AD/ADAS) there is also the complementary ISO 21448 safety of the intended functionality (SOTIF) standard [2]. However, these lacks specific guidance for addressing AI safety, particularly in managing AI's black-box nature and data dependency. The black-box nature of AI limits transparency and traceability in decision-making, while its data dependency means that the quality and representativeness of training data significantly influence system behaviour. These characteristics create gaps in traditional safety assessments, requiring new approaches to evaluate and ensure the safety of AI-driven systems.

This paper explores expanding FuSA assessment evaluations using the new ISO/PAS 8800 [3], which discusses safety and artificial intelligence in road vehicles. This extension addresses gaps by identifying and mitigating systematic faults and insufficiencies in AI systems to ensure safety during development and beyond. To this end, one may conduct independent assessments on an assurance case on a heterogeneous set of evidence from various test environments that align with multi-pillar methodologies such as those described in NATM (new assessment/test method for automated driving) [4] , which are relevant for AD/ADAS systems.

The main challenges in assessing AI systems within a safety assurance framework include evaluating performance robustness and identifying performance insufficiencies. Performance robustness refers to the system's ability to maintain safe operation under expected variations and disturbances within the operating conditions. Performance insufficiency occurs when AI systems fail in unanticipated scenarios

due to technical limitations. Addressing these issues is critical in FuSA and SOTIF to ensure that systems meet essential performance standards, even in dynamic or unpredictable environments. Systematic faults in AI design further complicate safety assessments. AI systems trained on operating data are prone to subtle design flaws that may introduce hidden hazards, which may only become evident under specific conditions. Clear guidance is therefore needed to manage these risks, ensuring that AI systems remain robust throughout the design and deployment phases.
This work extends conventional FuSA assessment for AI-based state of charge (SOC) systems to address AI-specific challenges, including training data quality and validity. It identifies safety cages as key architectural elements aligned with ISO 26262 and proposes them as a natural interface for integrating ISO/PAS 8800 to address core AI-related measures. Robustness evaluation is conducted using fault injection experiments, where systematically perturbed sensor inputs are applied to investigate an AI component's failure characteristics. This can provide evidence of the AI components' behavior under abnormal and unforeseen conditions, thereby addressing some inherent uncertainty in assessing AI-based components.

2 Integrating AI in Safety Assurance

Several safety-related standards for electric/electronic (E/E) systems in road vehicles may apply to the context of AI systems. The primary safety standard, ISO 26262, covers functional safety, including systematic and random hardware faults for all E/E systems in road vehicles. For systems with complex sensors, such as cameras, lidar, or radar, which is especially relevant for ADAS and AD functions, ISO 21448 [2] additionally covers functional insufficiencies. These include performance limitations in technical abilities (e.g., sensor performance) or insufficiencies in the specification, when either of these insufficiencies can lead to hazardous behaviour under some relevant conditions (triggering conditions). The safety of AI systems is covered in ISO/PAS 8800, which is developed to be used in conjunction with the two aforementioned standards. Figure 1 illustrates how these standards are interrelated, i.e., depending on the function under development, using two or all three may be necessary. This paper and the SOC case study focus on ISO 26262 and ISO/PAS 8800, which are sufficient for a system without complex sensors. As mentioned above, a similar integration must include ISO 21448 for systems with complex sensors.

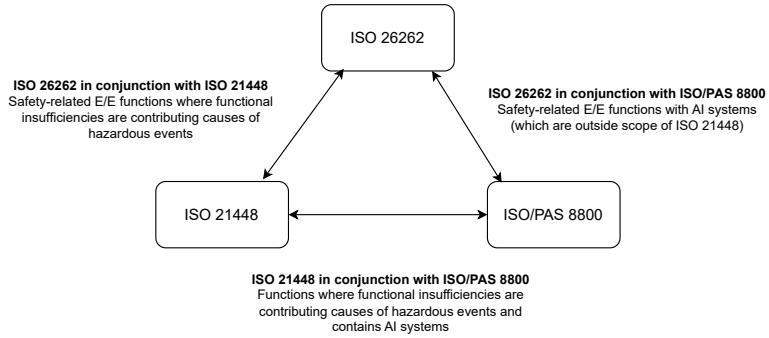


Figure 1: Relationship between road vehicle safety standards ISO 26262, ISO 21448, and ISO/PAS 8800 for use when developing functions with AI components.

A first step to integrate AI in safety assurance is to define for which parts ISO 26262 is still applicable and when it needs to be tailored to use ISO/PAS 8800 [3]. The latter defines an *AI system* as a top-level abstraction for AI-based functionality, an element containing one or more *AI components*. An AI component may be a pre- or post-processing component or an *AI model*, where the latter is a construct making inferences based on some input, e.g., a trained deep neural network with its weights and hyper-parameters. As illustrated in Figure 2, the overall AI system and AI models fall within the scope of ISO/PAS 8800. In contrast, the full *item*[1] and all non-AI system elements as well as AI components that are not AI models fall within the scope of ISO 26262. For instance, a pre-processing component that is not AI-based will be assessed under ISO 26262, even if it is also part of an AI system.
Based on this division, ISO/PAS 8800 proposes an AI reference lifecycle shown in Figure 3. It is based on the ISO 26262 development cycle, where safety requirements from the item under development are decomposed and allocated to the system during the system development phase. ISO/PAS 8800 tailors the ISO 26262 development cycle by adding that AI-related safety requirements are allocated to the AI system as illustrated on the left side of Figure 3.These requirements may also need to be adjusted during development and continuous assurance activities during operation in cases where the AI system cannot

[1] Item is the ISO 26262 term for a function at vehicle level that falls within its scope, i.e., a function containing E/E elements.

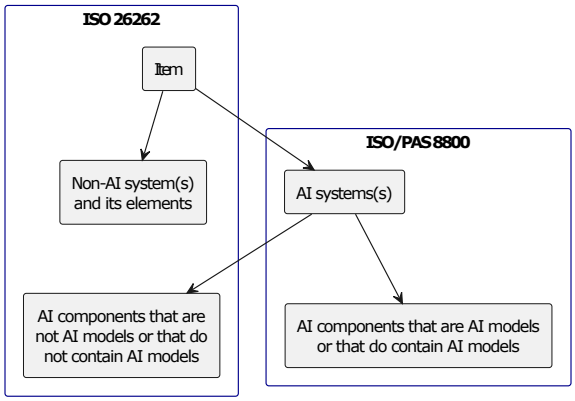


Figure 2: Illustration of applicability of ISO 26262 respectively ISO/PAS 8800 (based on [3]).

meet its safety requirements and related properties. Some of the reasons for this could be (i) difficulties in finding suitable training and testing data, (ii) limitations in the ability to generalize to new operating conditions, or (iii) insufficient evidence to demonstrate confidence in compliance with safety standards. These challenges require iterative feedback between the AI system and the encompassing system's safety concept and requirements.

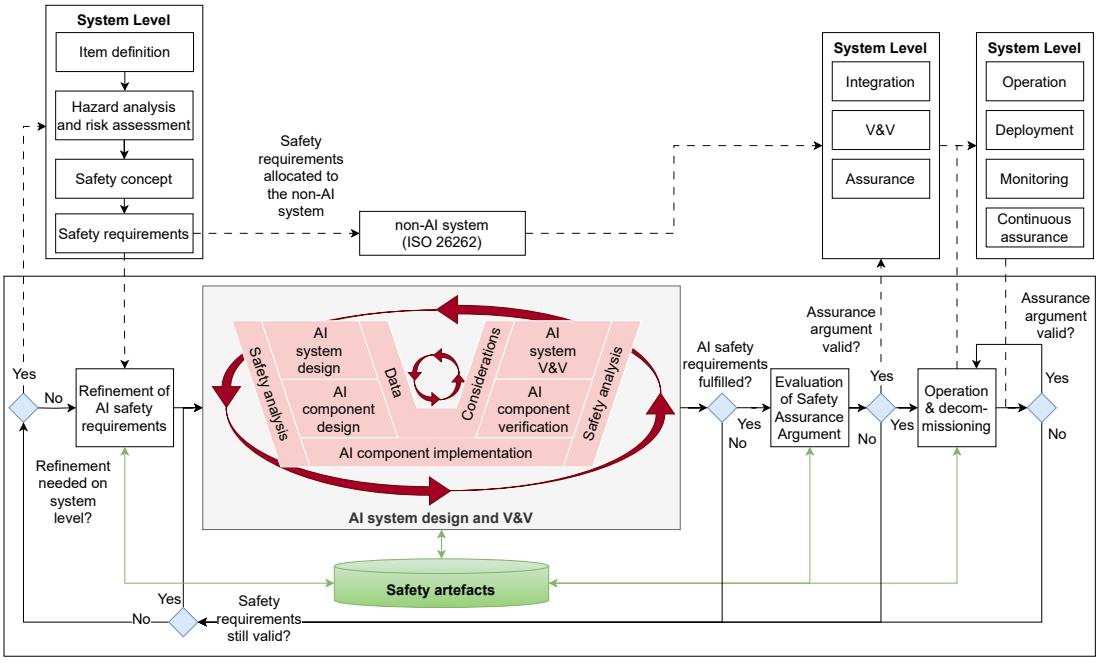


Figure 3: View of lifecycle for vehicle functions containing AI systems (based on [1, 3, 5]).

2.1 System-level design and verification

In the development lifecycle, ISO 26262 [1] would verify system-level artifacts such as technical safety concepts, safety requirements, architectural design, and architectural measures.
System-level design can also be adapted to increase the feasibility of fulfilling requirements allocated to AI systems. This can be achieved by, e.g., restricting the allowed operating conditions, incorporating diverse processing algorithms or sensing modalities, or implementing redundancy.

2.2 AI system verification

A key challenge in verification for most types of AI systems, including the most common type of machine learning (ML), is that the AI model is largely opaque, i.e., it is not possible to inspect the model or use

formal methods to determine if it fulfills the safety requirements. An inherent limitation of ML models, in particular, stems from their reliance on training data, where data quality issues or training method issues can result in unpredictable behaviour, especially in edge cases or rare conditions that deviate from the training data distribution. Addressing this issue necessitates appropriate verification methods but typically also robust fallback safety mechanisms capable of mitigating the potential consequences of such unexpected behaviour.
Thus, choosing appropriate combinations of test methods to address these challenges becomes important. Preferably, traditional safety-related test methods should be complemented with test methods specifically targeting AI systems. Some examples of such test methods are:

- Gradient-based search methods using analysis of the AI model to guide the generation of test cases.
- Statistical testing, i.e., evaluation of performance metrics measurable on the model under test, e.g., precision and recall, with the desired confidence interval.
- Test cases designed with expert domain knowledge or based on reviews of the used model and data set.
- Robustness testing, e.g., applying noise patterns or other disturbances in the input to measure the model's resilience to input variance.
- Use of explainability techniques for making the model's decisions semi-transparent. It can be used to understand how a model works and identify potential weaknesses or biases.
- Cross-validation, i.e., dividing the available dataset into several training/verification tuples to test if the model is robust for different training sets.
- Sampling-based methods that can guide testing to areas in the input space with higher error probability.

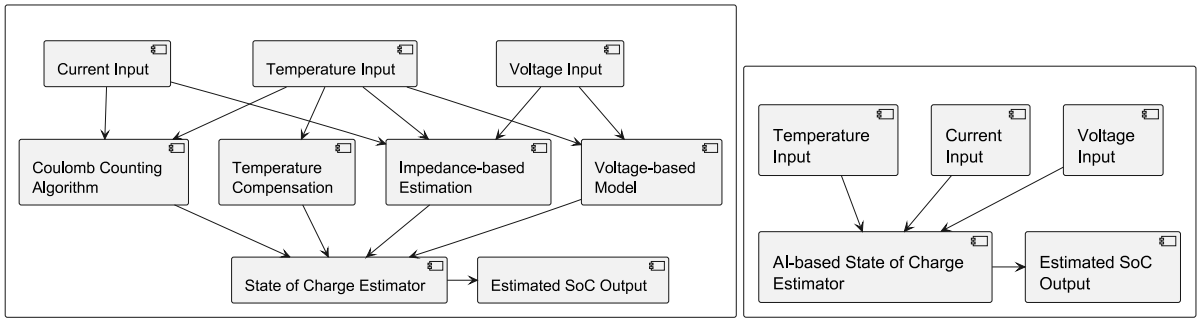
There may be challenges in the physical testing of AI systems operating in complex environments, particularly for achieving sufficient coverage of edge cases. In those cases, virtual testing can be used to complement physical testing. Virtual test platforms may also facilitate the generation of synthetic datasets to address the challenges of achieving adequate distribution and coverage of the inputs to AI systems. However, this requires the entire generation workflow to be validated and correlation with real data to be made.

2.3 AI system validation

In addition to field testing, virtual testing techniques used for verification may also be used for validation once integrated into the encompassing system, where they can be used to explore relevant scenarios systematically and identify corner cases or abnormal situations. Methods for detecting out-of-distribution data, i.e., input data that is not similar to the training data, can aid the evaluation [5].

2.4 AI system safety analysis

For the safety analysis of AI systems, the aim is to provide confidence that the risk of violating the AI safety requirements at the AI system level due to AI errors is sufficiently low. The safety analysis techniques should adequately identify hazards and their potential causes. Some off-the-shelf techniques may be reused or enhanced to analyse AI systems. Examples of such techniques include fault-tree analysis (FTA) [6], failure mode and effects analysis (FMEA) [7], and hazard and operability analysis (HAZOP) [8]. While these techniques analyse systems with certain underlying assumptions, other state-of-the-art techniques have been introduced with stronger assumptions to model AI systems [9, 10, 11].
If AI errors are identified as a result of testing, analyses are performed to evaluate their impact. Typically, the analysis activities include risk evaluation, root-cause analysis, and risk mitigation. Risk evaluation involves assessing the risk of a failed test to estimate the impact on safety. Root-cause analysis involves identifying the underlying issues for the AI errors, which may be related to AI safety requirements, datasets, or AI model design. After the risk evaluation and root-cause analysis have been performed, the risks are mitigated through the definition of prevention, detection, and control measures for the identified root causes. Thus, depending on the root cause, the mitigations may involve changes to the AI safety requirements, AI model, dataset, or the AI development process.



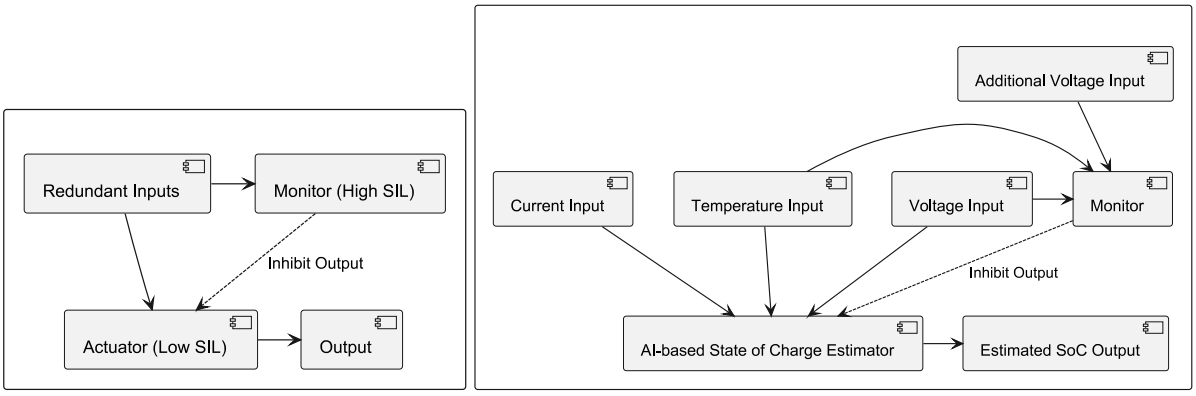
(a) (b)

Figure 4: SOC estimation methods: Traditional estimation (4a) and AI-based estimation (4b).

3 Case study: AI-based SOC estimator for EV battery

The SOC (state of charge) measures the remaining charge in a battery, typically expressed as a percentage of its total capacity. In EVs, SOC provides critical information for range estimation and safety functions, such as preventing overcharging and deep discharging in battery management systems (BMS). Failure to accurately measure SOC can result in overcharging, potentially causing excessive heat generation, electrolyte decomposition, and, in extreme cases, thermal runaway. Estimating SOC in batteries is challenging due to their nonlinear behavior and dependency on operating conditions such as temperature, aging, and discharge rates [12]. Figure 4a shows a traditional SOC estimation method that relies on coulomb counting, which measures the charge entering and leaving the battery; open-circuit voltage analysis, which maps the battery's voltage at rest to its SOC; and physics-based electrochemical models. Recently, however, AI-based SOC methods have gained traction due to their ability to model batteries' complex and nonlinear behaviour; such an estimator is illustrated in Figure 4b.

3.1 AI-based SOC estimator with monitor



(a) (b)

Figure 5: Actuator/Monitor Architecture safety patterns: General (5a, where SIL = safety integrity level) and applied to AI SOC estimation (5b) from Figure 4b.

A well-known safety pattern is to encapsulate a more complex function, where ensuring safety is difficult, by adding a simpler monitor to restrict the output of the complex function to safe ranges. The complex component will, in normal operation, provide better performance. In comparison, the monitor intervenes when the complex component fails but provides lower performance or keeps the system safe. Figure 5a illustrates this pattern. Input signals go to the complex component (actuator) and the monitor. There are some variations of the pattern, but in this example, the monitor can inhibit the actuator's output if the output calculated by the actuator component is outside the safe range. The advantage of the pattern is that the safety requirements can be allocated to simpler components that are feasible to develop with a high safety integrity level (SIL). However, it is important to realize that the monitor must be able to

intervene in all failure modes of the complex component. If not, there will be a shared responsibility where the complex component will still be assigned some safety requirements.
In AI, this pattern is sometimes called safety envelope or safety cage [13] and means there is a non-AI component monitoring the AI component. Figure 5b shows our case study SOC estimator using this pattern to handle some of the fault modes; a monitor uses voltage and temperature to be able to inhibit the SOC output such that thermal runway, the most critical consequence of erroneous SOC estimation, can be avoided.

3.2 AI-Specific Safety Assessment Features for SOC

The SOC estimator functions as a sensor and is subject to the same typical failure modes as conventional sensors. A detailed analysis is necessary to ensure its safety and proper functionality. As outlined in ISO 26262-5 Table D.9, the typical failure modes for sensors include **out-of-range signals, offset errors, signals remaining stuck within a valid range, and oscillatory behaviour**.
In addition to these standard failure modes, AI-based SOC estimation systems introduce at least two additional assessment considerations due to their potential contribution to the abovementioned failure modes.
Training Data Quality and Relevance: The performance of AI systems heavily depends on the quality of their training data. The data must be relevant, sufficiently representative, and as complete and error-free as possible. Inadequate or biased training data can lead to poor generalisation and incorrect SOC estimates.
Validity of Training Data Over Time: Training data is inherently limited to the conditions under which it was collected and cannot guarantee universal validity in future operational scenarios. To mitigate this, in-service monitoring must continuously verify that the assumptions and assertions made during training remain valid in the operational environment.
Given the fault modes and unique characteristics of AI systems, ensuring the safety of an AI-based SOC estimation system requires a careful assessment to confirm the existence of appropriate safety measures and evidence of their effectiveness, i.e., test results, that address both traditional and AI-specific challenges. The selection of test methods (Table 1) for each safety mechanism is based on their specific objectives in ensuring system safety. Methods for obtaining evidence of the correct implementation of functional and technical safety requirements at the system level are selected from ISO 26262-4, Table 9. Similarly, methods for validating correct functional performance, accuracy, failure mode coverage, and the timing of safety mechanisms at the system level are chosen from ISO 26262-4, Table 10.

Table 1: Safety Mechanisms, traditional Test Methods, and Assessment Aims adapted from ISO 26262-5 Table D.9 — Sensors

Measure Name	**Test Methods**	**Assessment Aim**
Input comparison/voting	Fault Injection Test, Error Guessing Test	Assess ability to detect discrepancies across redundant inputs or models, e.g., offset errors and signals stuck within a valid range.
Sensor correlation	Performance Test, Fault Injection Test	Assess ability to detect inconsistencies between sensors and mitigate sensor drifts in SOC estimation, e.g., offset errors and oscillatory behaviour.
Sensor rationality checks	Error Guessing Tests derived from Field Experience	Assesses the ability to detect implausible outputs and maintain SOC plausibility using diverse inputs, e.g., offset errors and out-of-range signals.

When mapping safety measures for an SOC sensor implemented according to the architectural pattern depicted in Fig 5a, some responsibilities fall to the monitor component and can, therefore, be addressed using traditional functional safety measures and test techniques listed in Table 1. However, as listed below, certain sensor-related AI-concerns (AIC) pertain specifically to the AI-based SOC estimation. Thus, they require the integration of an ISO/PAS 8800 tailored process, depicted in Figure 6, and test techniques selected from Section 2.2 to provide evidence of safety objectives fulfillment.

AIC1 Detecting failures through *online* monitoring, emphasizing evaluating the system's ability to identify deviations in behaviour during normal operation and to manage anomalies in the AI model. This includes detecting conditions such as **signals remaining stuck within a valid range** and **oscillatory behaviour**.

AIC2 Assessing the AI model's capability to detect *static* failures and deviations by employing test patterns that compare the AI's output against expected behaviours. Relevant fault modes include **signals remaining stuck within a valid range** and **offset errors**.

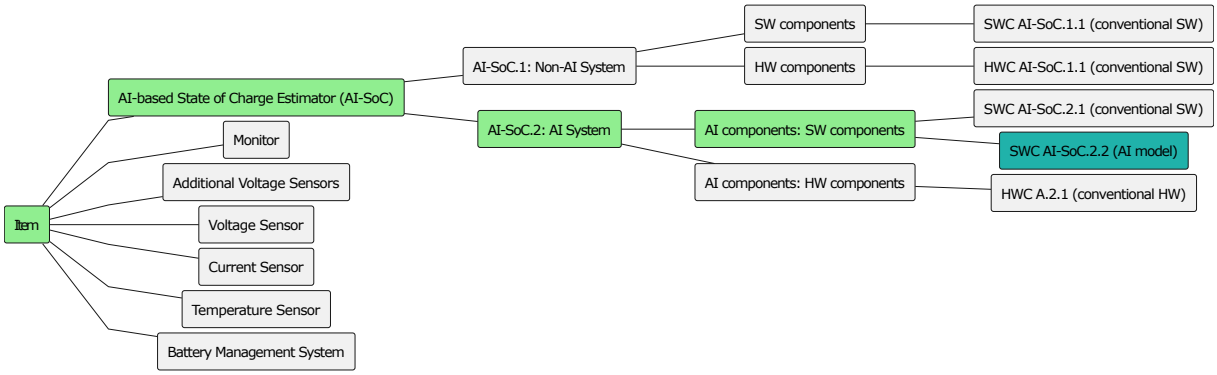


Figure 6: Hierarchical decomposition of the AI SOC estimator in Figure 5b. The green colour indicates elements subject to the ISO/PAS 8800 tailored process.

AIC3 Validating sensor input ranges by verifying the system's ability to detect and appropriately respond to **out-of-range signals**, ensuring that SOC estimation remains reliable despite invalid sensor data.

The list of concerns was compiled through a systematic assignment of each measure specified in ISO 26262-5 Table D.9 to either the monitor or the actuator based on their functional relevance. It subsequently identifies the measures and fault modes that the AI-based component of the sensor should address. In this context, particular attention is given to fault modes that may result in underestimating the state of charge. This condition is closely associated with the most severe hazard, battery outgassing in the passenger compartment.

3.3 Experiments

Robustness testing, described in Section 2.2, such as applying noise patterns or other disturbances to the inputs to evaluate the model's resilience to input variance as described in **AIC2**, constitutes an appropriate approach to provide evidence for the mitigation of failures related to the underestimation hazard. An example of such testing is investigated in the following Sections.

3.3.1 Experimental method

Fault injection test is a common method for evaluating safety measures, which involves accelerating the occurrences of faults for evaluating the dependability and cybersecurity properties of systems [14]. Fault injection is performed by inserting artificial faults or errors into the system, often using simple fault models such as *stuck-at-0* and *stuck-at-1* for permanent faults, which set the logic values to 0 or 1 respectively, and *bit-flips* which are typically used for transient faults, where the logic values are inverted. The injected faults typically correspond to operational faults such as shortcuts, breaks, electromagnetic disturbances, radiation particle strikes, etc. Still, they may also correspond to development faults, such as programmers' mistakes or flaws in semiconductor devices. In the case of evaluating cybersecurity properties of systems, fault injection may also be referred to as attack injection [15].
Fault injection may be performed at many different abstraction levels and design stages, depending on the availability of system models or physical prototypes. Common fault injection techniques include simulation-based and physical techniques. Simulation-based techniques are typically used at early development stages where faults may be injected into hardware-, software- or system models, i.e., model-level fault injection. Physical fault injection is used to inject faults at the hardware level at later development stages when the actual physical system or prototype is available. Common physical techniques include pin-level- and radiation-based fault injection and other methods, e.g., using debug/test logic, EMI, or power supply disturbances. Additional software for injecting faults is typically referred to as software-implemented fault injection (SWIFI). SWIFI is an attractive technique commonly used for its flexibility and cost-effectiveness compared to other techniques. Two main approaches are used: *Runtime injection*, which injects faults during system operation, and *pre-runtime injection*, which injects faults before system operation.
The stuck-at fault model is commonly used in fault injection experiments and is often applied on a single logic value (or bit), which is permanently set to 0 (stuck-at-0) or 1 (stuck-at-1) in the target system for each experiment. Apart from being a common fault model used for emulating the effects of permanent hardware faults, stuck-at faults are also part of the failure modes for sensors outlined in ISO 26262-5 Table D.9. A survey of papers from five major conferences on dependability (DSN, ISSRE, SafeComp,

PRDC and EDCC) published during the last 6 years (2019-2024)[2] reveals that SWIFI is commonly used for evaluating AI-based systems and that that the stuck-at fault model is often used, e.g., [16, 17, 18].

Conducted experiment Following the above paragraph, we chose to perform our fault injection experiments on the AI-driven SOC estimation system using pre-runtime SWIFI with the stuck-at-fault model applied to multivariate input signals of an AI SOC estimation system. To detect aberrations, we compare predictions for the SOC from original data with those from data corrupted via stuck-at fault injection using absolute deviation and Root Mean Squared Error (RMSE).

3.3.2 System under test

The experiments investigate how pre-runtime injection of stuck-at faults into the test data affects the model's SOC estimations. The injected bits of the test data are part of `Float64` values representing "Voltage," "Current," or "Temperature" inputs, which the model receives at each time instance (referred to as "steps").
The specific model used for the experiments in this paper is a Recurrent Neural Network (RNN) with Long Short-Term Memory (LSTM) cells, as presented in [19]. This model generates a SOC percentage prediction by processing the information from a fixed number (N) of preceding steps. Consequently, one prediction is generated every Nth step, independent of other predictions made. Model parameter values shown to provide good performance with uncorrupted data were selected for the experimental setup. The model performs well with room-temperature data (25°C) and when trained to process 300 steps of preceding information for each SOC prediction. A pre-trained model with these specifications, along with a Python implementation for SOC estimation, is provided by the authors of [19] and utilised in this paper. The implementation normalises input (Voltage, Current, Temperature) values from the dataset to be within the range of zero to one before the model receives them. Said model has been trained and tested using the "LG 18650HG2 Li-ion Battery" dataset, available at [20]. Although the training data only consists of six mixed discharge cycles, we believe that the results obtained with the available dataset are illustrative for cycles involving charging as well. In any event, incorrect SOC estimations may have potential safety implications, e.g., if the estimated SOC is too low at the end of a discharge cycle when the charge cycle begins.
Single stuck-at-0 or stuck-at-1 faults are injected at the start of the entire discharge cycle for each experiment. The faults are systematically injected into bits 3 to 64 for each 64-bit floating point value of the inputs (Voltage, Current, Temperature). The initial two bits are exempted, as the resulting value of the injected float may be large enough to trigger exceptions in the programming code rather than affecting the model's SOC output.

3.3.3 Results

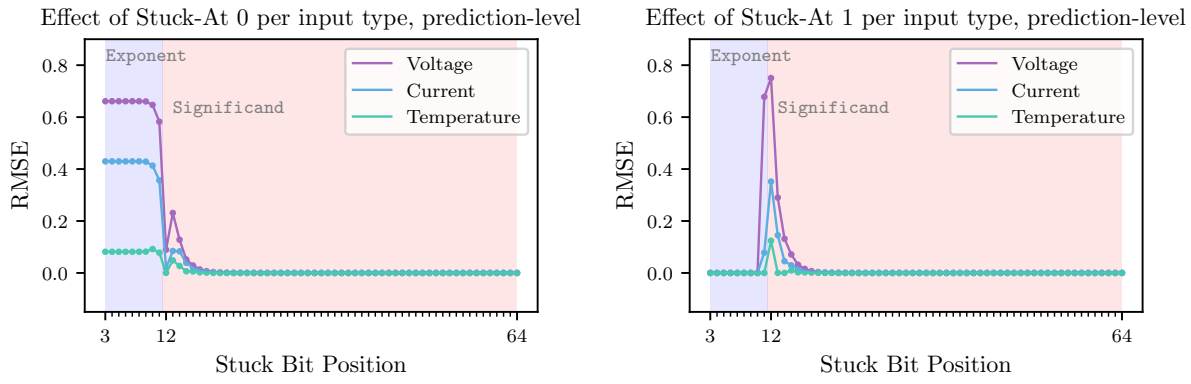


Figure 7: RMSE between SOC predictions from original and corrupted data for stuck-at faults injected into bits 3-64 of input type sensor, which are in turn a series of `Float64`. 7a: stuck-at 0; 7b: stuck-at 1.

Figures 7a (stuck-at-0) and 7b (stuck-at-1) displays the results for each input type, where the experiment consisted of injecting a stuck-at fault into a single bit in the very beginning of the discharge cycle, and comparing the resulting SOC predictions with those from the uncorrupted data. Each point in the plot

[2]The survey in question is not yet published.

represents the RMSE value for the whole discharge cycle. Whether the flipped bit was in the *exponent* or *significand* of the `Float64` is highlighted in this plot, since the former has a greater impact on the floating point's value[3]. In this experiment, it appears that the sensors differ in how much they affect the model's predictions, with fault injections in "Voltage" causing the largest errors, followed by "Current" and finally "Temperature". Furthermore, bits in the significand have a starkly decreasing (with bit index) influence on the results. As the data was normalised in [19] to the range $[0; 1]$, bits 3 to 10 all have the value '1' for all sensors and at every step. Thus, stuck-at 1 has no effect for these bits, and effects only become visible for bits 11 and higher. For stuck-at 0, we see the reverse effect, as now making the first bits in the exponent stuck at '0' leads to significant RSME of the predictions.

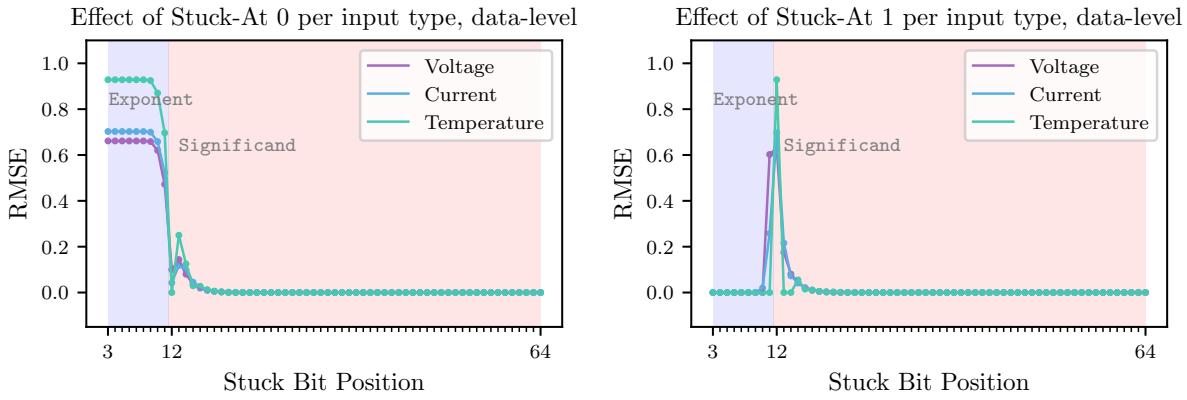


Figure 8: RMSE between original and corrupted data for stuck-at faults injected into bits 3-64 of each input type, which are in turn a series of `Float64`. 8a: stuck-at 0; 8b: stuck-at 1.

Figures 8a and 8b show the RSME deviations not on the level of the model's predictions, but on the level of deviations between the original data and the corrupted data. These results mirror those from the predictions above and show that large deviations in the predictions are tied to large deviations in the data. Furthermore, it appears that even though the effect of stuck-at-0 on temperature is the largest on a data level, on a prediction level it is the smallest (see Figure 7a). This suggest that an internal (and opaque) weighting of inputs by an AI model can make it difficult to predict behaviour from only looking at the input data.

Figure 9 illustrates the Absolute Deviation for each prediction of SOC throughout the discharge cycle for stuck-at-1 faults. This visualisation enables the analysis of the model's behaviour at various stages of the discharge cycle. For clarity, the figure displays only the effects on exponent bits (bits 3 to 12).

As previously mentioned, bits 3 to 10 already have the value 1 at the start and are unaffected by the injected stuck-at-1 faults; thus, no sensitivity is observed for these bits as a function of SOC percentage. However, significant influences from the Voltage input begin at bit 11 and below 44% charge. For Current, the influence of bit 11 is less pronounced but more spread out throughout the discharge cycle. For Temperature, only bit 12 has an effect.

Additionally, the largest zone of impact appears at different locations for each input type. For Voltage, this occurs at bit 11 below approximately 44% charge. For Current, it occurs at bit 12 above roughly 72.6% charge, and for Temperature, at bit 12 below 32.9% charge.

The behaviour observed in Figure 9 aligns with the results presented earlier in Figure 8b, where the RMSE value increases only when a high exponent bit (11 or 12) is stuck-at-1. The mechanism through which this affects the model's predictions is challenging to determine without a thorough understanding of its inner workings. However, as previously stated, the different input types (Voltage, Current, Temperature) may be subjected to various internal weighting. Both Figures 7 and 9 demonstrate that significant deviations from the original predictions can occur if the exponent bits of the model's inputs are altered.

To understand the varying impact of stuck-at faults on the Absolute Deviation at different locations for each input type in Figure 9, one needs to examine the changes in `Float64` values throughout the discharge cycle. Although all model inputs are normalised within the range $[0; 1]$, as stated earlier, they can still differ in how much of this interval they utilise. For `Float64` in the range $(1; 0)$, it is true that bits 3-10 are all 1, while bits 11 and 12 are 10_2 for values in $(1; 0.5]$ and 01_2 for values in $(0.5; 0)$. During our tests, we observed that Temperature remains relatively stable throughout the discharge cycle with a value close to 1. In contrast, Voltage and Current fall from values close to 1 to values below

[3]The `Float64` standard is specified in IEEE 754: Bit 1 is the sign, bits 2-12 are the exponent, and the remaining bits compose the significand or mantissa

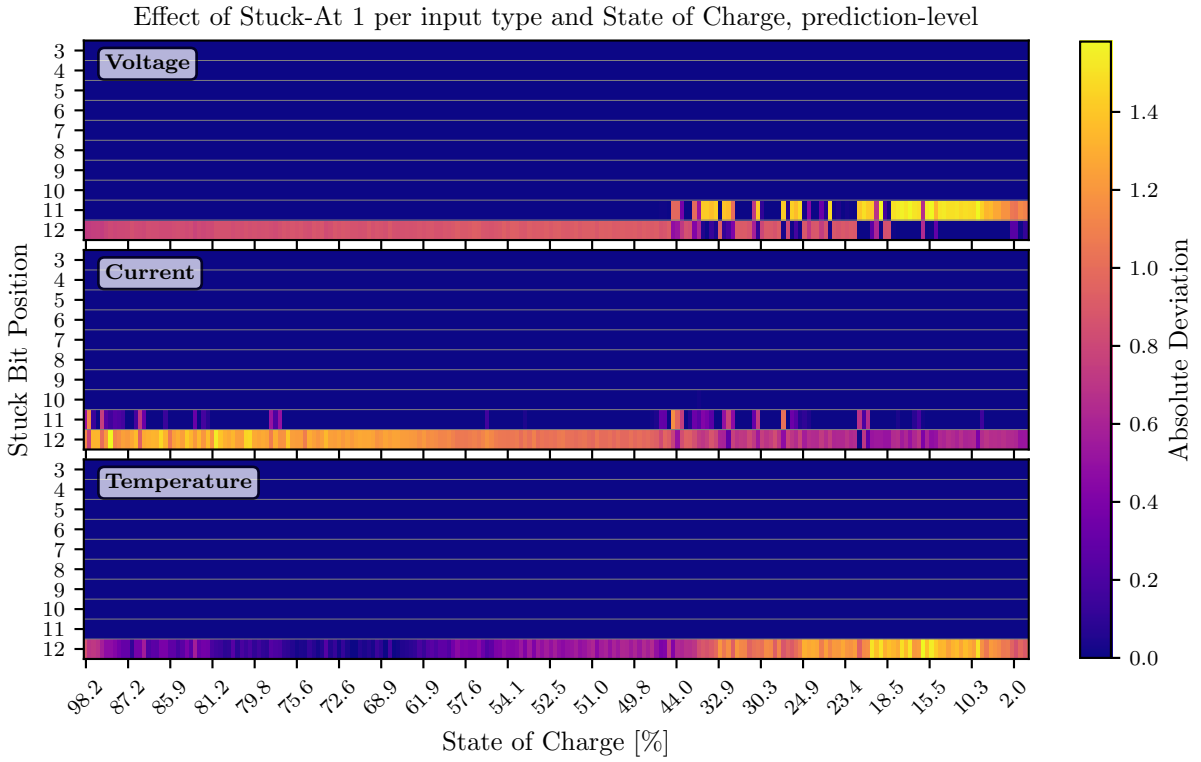


Figure 9: Absolute Deviation between SOC predictions from original and corrupted data throughout the discharge cycle. Stuck-at 1 faults injected into bits 3-12 of each input type, which are in turn a series of `Float64`.

0.5 during the discharge cycle. Thus, the sensitivity to stuck-at-1 faults changes depending on the data value, explaining the patterns seen in Figure 9: Temperature is only sensitive in bit 12, while Voltage and Current are sensitive even in bit 11 if their value falls in the interval $(0.5, 0)$.

Summary The experiments underline that stuck-at faults can cause significant deviations from the AI component's original predictions, potentially resulting in undesired behaviour, such as markedly inaccurate SOC predictions. These findings highlight the critical importance of detecting errors like stuck-at faults in the input data using the monitor system proposed in Section 3.1.

4 Conclusions and future work

This work addresses the challenges of ensuring the safety of AI-based systems, focusing on SOC estimation. The main contribution, besides an introduction for integrating AI in the V-Model for safety assessments, is the identification of the architectural element of the safety cage as a good candidate for demarcation and interface between traditional, AI-independent measures aligned with ISO 26262 and AI-dependent measures requiring alignment with emerging standards like ISO/PAS 8800. Our experiments underline how sensitive an AI SOC prediction model can be to common faults such as *stuck-at*. Future work will enhance the multi-concern assessment framework [21] by integrating new AI findings and refining methods to address data quality, in-service monitoring, and fallback strategies for AI-based systems. Furthermore, we will analyse the effects of additional failure modes from **AIC1**, **AIC2**, **AIC3** on AI SOC prediction, and investigate the quality and relevance of using training data with faults already applied in order to improve the robustness of the AI-driven SOC estimations. Finally, it will evaluate the effectiveness of the safety monitor itself.

Acknowledgments

We acknowledge the support of the Swedish Knowledge Foundation via the industrial doctoral school RELIANT, grant nr: 20220130. This research was carried out within the SUNRISE project and is funded by the European Union's Horizon Europe Research and Innovation Actions under grant agreement No.

101069573. However, views and opinions expressed are those of the author(s) only and do not necessarily reflect those of the European Union or the European Union's Horizon Europe Research and Innovation Actions.

References

[1] International Organization for Standardization (ISO), "ISO 26262:2018 Road Vehicles – Functional Safety."

[2] ——, "ISO 21448:2022 Road vehicles — Safety of the intended functionality."

[3] ——. ISO/PAS 8800 Road vehicles — Safety and artificial intelligence. ISO. [Online]. Available: https://www.iso.org/standard/83303.html

[4] ECE/TRANS/WP.29/2021/61, "(GRVA) New Assessment/Test Method for Automated Driving (NATM) - Master Document — UNECE," World Forum for Harmonization of Vehicle Regulations.

[5] J. Henriksson, S. Ursing, M. Erdogan, F. Warg, A. Thorsén, J. Jaxing, O. Orsmark, and M. O. Toftås, "Out-of-distribution detection as support for autonomous driving safety lifecycle," in *Requirements Engineering: Foundation for Software Quality*, A. Ferrari and B. Penzenstadler, Eds. Springer Nature Switzerland, pp. 233–242.

[6] N. Ali, M. Hussain, and J.-E. Hong, "Analyzing Safety of Collaborative Cyber-Physical Systems Considering Variability," *IEEE Access*, vol. 8, pp. 162 701–162 713, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9186018/

[7] R. Salay, M. Angus, and K. Czarnecki, "A Safety Analysis Method for Perceptual Components in Automated Driving," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. Berlin, Germany: IEEE, Oct. 2019, pp. 24–34. [Online]. Available: https://ieeexplore.ieee.org/document/8987559/

[8] Y. Qi, P. R. Conmy, W. Huang, X. Zhao, and X. Huang, "A Hierarchical HAZOP-Like Safety Analysis for Learning-Enabled Systems," 2022, version Number: 1. [Online]. Available: https://arxiv.org/abs/2206.10216

[9] A. Adee, R. Gansch, and P. Liggesmeyer, "Systematic Modeling Approach for Environmental Perception Limitations in Automated Driving," in *2021 17th European Dependable Computing Conference (EDCC)*. Munich, Germany: IEEE, Sep. 2021, pp. 103–110. [Online]. Available: https://ieeexplore.ieee.org/document/9603704/

[10] A. Adee, R. Gansch, P. Liggesmeyer, C. Glaeser, and F. Drews, "Discovery of Perception Performance Limiting Triggering Conditions in Automated Driving," in *2021 5th International Conference on System Reliability and Safety (ICSRS)*. Palermo, Italy: IEEE, Nov. 2021, pp. 248–257. [Online]. Available: https://ieeexplore.ieee.org/document/9660641/

[11] M. Berk, O. Schubert, H.-M. Kroll, B. Buschardt, and D. Straub, "Assessing the Safety of Environment Perception in Automated Driving Vehicles," *SAE International Journal of Transportation Safety*, vol. 8, no. 1, pp. 49–74, 2020, publisher: SAE International. [Online]. Available: https://www.jstor.org/stable/27034112

[12] O. Demirci, "Review of battery state estimation methods for electric vehicles - part i: SOC estimation."

[13] A. V. S. Neto, J. B. Camargo, J. R. Almeida, and P. S. Cugnasca, "Safety Assurance of Artificial Intelligence-Based Systems: A Systematic Literature Review on the State of the Art and Guidelines for Future Work," *IEEE Access*, vol. 10, pp. 130 733–130 770, 2022.

[14] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[15] B. Sangchoolie, P. Folkesson, and J. Vinter, "A study of the interplay between safety and security using model-implemented fault injection," in *2018 14th European Dependable Computing Conference (EDCC)*. IEEE, 2018, pp. 41–48.

[16] S. Qutub, F. Geissler, Y. Peng, R. Gräfe, M. Paulitsch, G. Hinz, and A. Knoll, "Hardware faults that matter: Understanding and estimating the safety impact of hardware faults on object detection DNNs," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13414 LNCS, 2022, pp. 298–318.

[17] M. Beyer, J. Borrmann, A. Guntoro, and H. Blume, "Online quantization adaptation for fault-tolerant neural network inference," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 14181 LNCS, 2023, pp. 243–256.

[18] J. Ruiz, D. De Andres, L. Saiz-Adalid, and J. Gracia-Moran, "Zero-space in-weight and in-bias protection for floating-point-based CNNs," in *Proceedings - 2024 19th European Dependable Computing Conference, EDCC 2024*, 2024, pp. 89–96.

[19] K. L. Wong, M. Bosello, R. Tse, C. Falcomer, C. Rossi, and G. Pau, "Li-Ion Batteries State-of-Charge Estimation Using Deep LSTM at Various Battery Specifications and Discharge Cycles," in *Proceedings of the Conference on Information Technology for Social Good*, ser. GoodIT '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 85–90. [Online]. Available: https://doi.org/10.1145/3462203.3475878

[20] P. Kollmeyer, C. Vidal, M. Naguib, and M. Skells. (2020) LG 18650HG2 Li-ion Battery Data and Example Deep Neural Network xEV SOC Estimator Script. Version 3. [Online]. Available: https://data.mendeley.com/datasets/cp3473x7xv/3

[21] M. Skoglund, F. Warg, A. Thorsén, and M. Bergman, "Enhancing safety assessment of automated driving systems with key enabling technology assessment templates," *Vehicles*, vol. 5, no. 4, pp. 1818–1843, 2023. [Online]. Available: https://www.mdpi.com/2624-8921/5/4/98

Presenter Biography

Martin Skoglund works within the Dependable Transport Systems unit at RISE Research Institutes of Sweden in Borås, focusing on safety assurance for connected and automated systems through advancing methods for evaluating safety and security-informed safety. His expertise encompasses functional safety, cybersecurity, safety of the intended functionality, and artificial intelligence, with research aimed at improving the efficiency and effectiveness of safety assessments to ensure safe and correct operation under all relevant conditions.