White Paper

Safe AI

How is this possible?

Munich Center for Trustworthy AI:

Fraunhofer IKS

fortiss

Safe AI — How is this possible?

Authors:

Dr. Harald Rueß

fortiss GmbH
Guerickestr. 25
80805 Munich
ruess@fortiss.org

Prof. Dr. Simon Burton

Fraunhofer Institute for Cognitive Systems IKS
Hansastraße 32
80686 Munich
simon.burton@iks.fraunhofer.de

Safe AI— How is this possible?[1]

Dr. Harald Rueß

fortiss
Research Institute of the Free State of
Bavaria for software-intensive Systems

ruess@fortiss.org

www.fortiss.org

Prof. Dr. Simon Burton

Fraunhofer IKS
Fraunhofer Institute for
Cognitive Systems

simon.burton@iks.fraunhofer.de

www.iks.fraunhofer.de

Munich, 4th February 2022

"As we know,
there are known knowns.
There are things we know
we know.
We also know
there are known unknowns.
That is to say,
we know there are
some things we do not know.
But there are also unknown unknowns,
the ones we don't know,
we don't know."

Donald Rumsfeld, Feb 2002, US DoD news briefing

[1] This work is funded by the Bavarian Ministry for Economic Affairs, Regional Development and Energy as part of the fortiss AI Center and a project to support the thematic development of the Fraunhofer Institute for Cognitive Systems. We are also grateful to Carmen Cârlan and Henrik Putzer for their thorough remarks and suggestions for improvement; in particular, Figure 2 is due to Carmen.

Table of Content

1. Introduction

A new generation of cyber-physical systems (CPSs) with cognitive capabilities is being developed for real-world control applications. Examples are self-driving vehicles, flexible production plants, automated surgery robots, smart grids, and cognitive networks. These systems are artificial intelligence (AI)-based in that they leverage techniques from the field of AI to flexibly cope with imprecision, inconsistency, and incompleteness, to have an inherent ability to learn from experience, and to adapt according to changing and even unforeseen situations. This extra flexibility of AI, however, makes its behavior more difficult to predict, and the challenge is to construct AI-based systems without incurring the frailties of "AI-like" behavior [1].

In addition, cyber-physical AI systems usually are safety-critical in that they may be causing real harm in (and to) the real world. As a consequence, the central safe AI objective is to handle or even overcome the dichotomy between safety and the largely unpredictable behavior of complex AI systems.

Consider, for example, an automated emergency braking system for a car that continually senses the operational context based on machine learning (ML), assesses the current situation via an AI decision module based on models of the operational context (and itself), and initiates a maneuver for emergency braking by overriding the human driver, when necessary. The intent of this emergency maneuver is, of course, to prevent accidents in time-critical situations that the human operator may not be able to control anymore. The emergency braking maneuver itself is also safety-related, as wrongful execution might cause severe harm.

The safe AI challenge is not exactly new [2] and may well be traced back to Turing himself in the early 1950s. Still, it has recently become all-important because of the euphoric mood about AI, as the acceptance and the success of AI techniques for real-world applications hinge on a meaningful, dependable, and safe control. Ongoing discussions about the responsible deployment of AI in the real world range from human-centered social norms and values[2] to its robust and safe realization [3] [4].

In this thought outline, however, we restrict ourselves to the technical design and engineering principles of safe AI systems as a necessary step for the responsible deployment of mission- and safety-critical AI systems into our very societal fabric. Moreover, although we are concentrating only on safety aspects in this thought outline, we believe that the suggested approach also fruitfully intersects with related dependability attributes of AI systems, such as security, privacy, inverse privacy, fairness, and transparency.

[2] https://www.ai4europe.eu/ethics

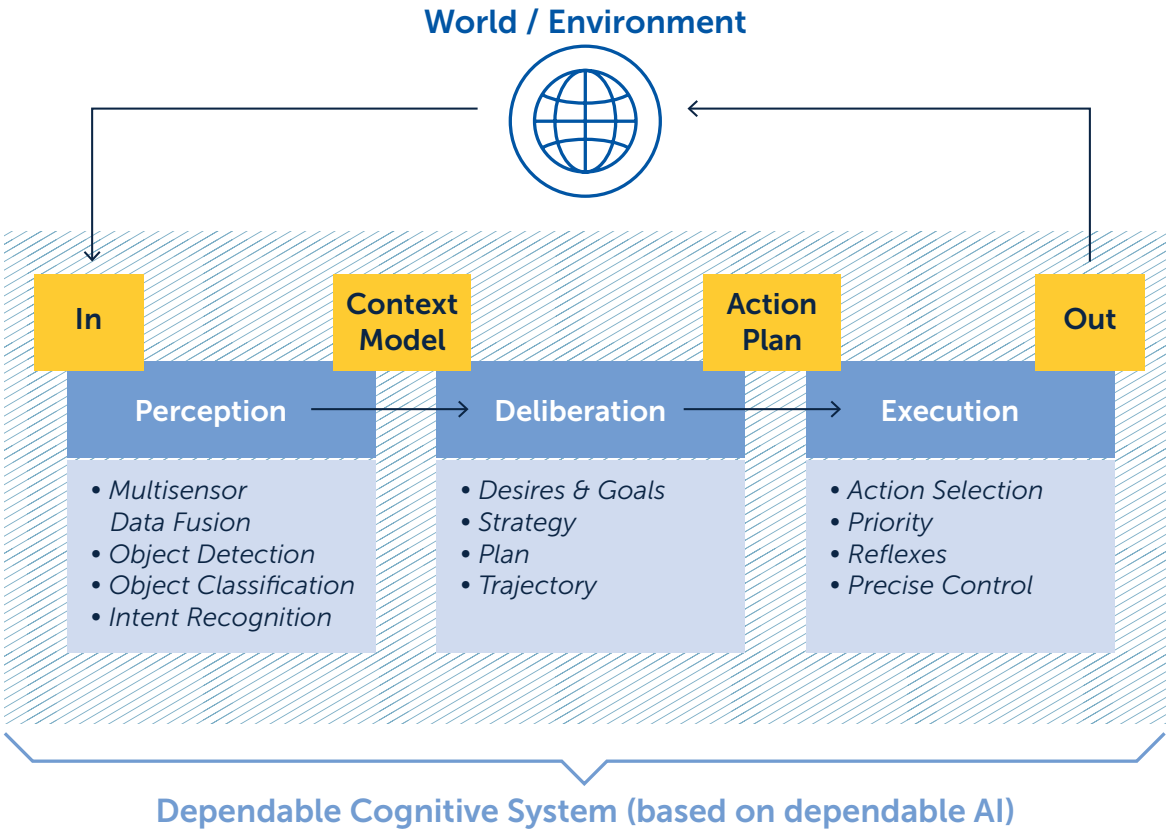

Figure 1. Sense-plan-act loop of a cognitive system.

The starting point of our considerations is cognitive systems, which are software-intensive technical systems that imitate cognitive capabilities, such as perception, model-building, and reasoning. More specifically, the basic sense-plan-act control loop of the cognitive AI system[3] in Figure 1 is based on monitored observations of the operational environment (including the controlled plant), perception, and interacting commands from human operators. Functionally automated driving systems, such as the emergency braking example, may easily be considered instances of this sense-plan-act loop, where the ego car is the plant to be controlled.

The cognitive system in Figure 1 is conceptually a function taking sensing inputs and generating corresponding output actions, which is usually also based on the internal state. Although this loop may be used as the conceptual specification of a reactive CPS [5], it is also the central technical concept of the AI field, which is concerned principally with designing the internals of stream-transforming controls for mapping from a stream of raw perceptual data to a stream of actions.

Behavior generation for the sense-plan-act loop is decomposed into successive stages for situational awareness, followed by deliberate, goal-oriented planning and by execution of selected actions in the real world. Sensing functionalities, in particular, are currently often realized through data-driven ML methods, such as artificial neural networks (ANNs). Behavioral planning capabilities, on the other hand, are usually realized by more traditional software-based control methods but also through probabilistic and reinforcement-based synthesis of control strategies. This conceptual separation into sensing and deliberate planning is supported, among others, by the global workspace theory, which categorizes cognitive capabilities into fast and slow

[3] For our purposes, we use the terms "AI system", "cognitive AI system", and "cognitive system" largely interchangeably.

modes of operation: System 1 operates rapidly, intuitively, and effortlessly, whereas System 2 requires concentration, motivation, and the application of learned rules, and it allows us to grasp the right ones.[4] In other words, System 1 means snap judgments that seduce us with the wrong answers, and System 2 means thinking twice [6].

The context model of our running example, automated emergency braking, might consist of the positions, bounding boxes, and motion vectors of surrounding environmental objects, such as cars or cyclists.
The sensing stage constructs and updates faithful models,[5] based on perceived inputs and other knowledge sources, of the exogenous operating environment and the endogenous self. One can easily imagine scenarios in which failure of detection, misclassification, or imprecision in models causes an accident.
The main challenge, therefore, is to provide a convincing argument that an AI system is sufficiently safe as determined through applicable risk and safety analysis. As usual, this notion of sufficiently safe heavily depends on the specific societal context and, correspondingly, acceptable risks.

For our purposes, automated emergency braking (EB) is intuitively said to be safe if its activation prevents, at least up to some tolerable quantity, accidents in prescribed situations. Assuming we can identify a corresponding subset S of "known" safe states of the operating context, the safety envelope, then the safety challenge for EB reduces to verifying the safety invariant $EB(S) \subseteq S$. In this way, EB, when initiated in a potentially unsafe and uncontrollable (for the driver) state in S, produces safe control actions, in that the ego car is always maneuvered toward a safe and controllable state, possibly a fail-safe state, and as the basis for a possible handover to the driver. As with most CPSs, ensuring safe control involves a rather complex interaction of uncertain sensing, discrete/probabilistic computation, physical motion, and real-time combination with other systems (including humans). We are arguing that traditional safety engineering techniques for embedded systems and CPSs are, for the multitude of heterogeneous sources of uncertainty, not applicable to learning-enabled cognitive systems, which are acting increasingly autonomous in open environments.

We identify central specification, uncertainty, assurance, design, analysis, and maintenance challenges for realizing this rigorous design of safe AI, all based on the notion of managing uncertainty to acceptable levels.[6] An overview of these challenges is provided in Table 1 — without any claim of completeness.

In addition, notice that because safe AI engineering is in its infancy, at times this exposition may seem to be rather sketchy and speculative, and clearly, many of our claims and hypotheses need further substantiation or disproval. In this sense, this thought outline should be provocative and thought-inspiring. It is also intended to be a living document, which needs to be updated and concretized as we gain more experience and increase our theoretical understanding of the rigorous design for safe AI — as the basis for the responsible and safe deployment of AI in our economic and societal fabric.

[4] Notice, however, that this suggested separation-of-means has exceptions, such as Nvidia's end-to-end-control for an experimental self-driving system [119].

[5] So-called digital twins.

[6] In analogy to the "as low as reasonably possible" (ALARP) risk-based criterion, we might call this the "as certain/confident as reasonably possibly" (ACARP) principle.

2. Challenges

Uncertainty and Complexity

The cognitive capabilities of CPSs are enabled by advances in AI, in particular, ML, as well as the large-scale availability of training and validation data through an increasing number of sensing channels and connectivity. As motivated above, the deployment of such systems is leading to substantial challenges in safety assurance, including existential questions, such as can AI systems ever be considered safe enough? We now look at some of the legitimate reasons for these doubts before focusing on AI-specific topics in later sections.

Previously, safety-critical electric/electronic (E/E) systems were assured by considering the impact of malfunctions caused predominantly by random hardware failures or system design faults, including but not exclusive to software bugs. This scrutiny allowed for a model-based approach to understanding the failure modes of individual components and how faults in individual components propagate through a system, leading to hazardous actions. However, introducing safety-critical cognitive systems requires a broader consideration of safety and potential causes of hazards. Many of these challenges can be related to the increasing complexity and uncertainty within a system and its environment.

Uncertainty. A particular challenge is that an AI system contains a multitude of sources of entangled uncertainty. The inductive capability of ML for extracting models from data is inseparably connected with uncertainty, but there is also uncertainty regarding the operating context, there is uncertainty regarding the models of the operating context and the "self," there is behavioral uncertainty due to the approximate nature of heuristic learning algorithms, there is uncertainty due to probabilistic and non-deterministic components, there is uncertainty regarding safety hazards,[7] there is uncertainty regarding safety envelopes in uncertain operating contexts, there is uncertainty in a meaningful fallback to a responsible human operator and, finally, there is uncertainty in self-learning systems concerning their emergent behavior in time. Possibly the only thing that is certain about an AI system is that it is uncertain and largely unpredictable.

As an example, let's investigate the sources of uncertainty of ML components, such as ANNs, in more detail. The input-output behavior of ANNs heavily relies on selecting "complete" and "correct" (with respect to the ground truth) sets of training and support data to faithfully specify relevant operating contexts (input) and their intended internal representation (output). Another source of uncertainty for these ML algorithms is the use of stochastic search heuristics, which may lead to incorrect recall even for inputs from the training data, and the largely unpredictable capability of generalizing from the given data points. Uncertainty regarding the faithfulness of the training data representing operating contexts and uncertainty regarding the correctness and generalizability of training also combine in a, well, uncertain manner. The consequences of these accumulated uncertainties are profound. Particularly, ANNs are usually not robust with unseen inputs, as there is also quite some uncertainty in their behavior for even small input changes.[8]
Adequate approaches are needed to measure the (un)certainty in the input-output behavior of an ANN with respect to the real world. For example, how certain are we that a given ANN correctly classifies certain classes of homeomorphic images of a "cat"? How certain should an ego vehicle be that there will be no surprises, such as undetected or misclassified vehicles, before initiating an emergency brake? Based on these certainty measures, internal models of the operating context should be equipped with confidence levels or, more generally, confidence intervals or distributions. The EB assistant, for example,

[7] For instance, dynamic hazards, such as the sudden occurrence of objects on a road, which may lead to catastrophic failure.
[8] For instance, "one-pixel attacks" for fooling deep neural networks [22].

may, as the basis for selecting appropriate action, assign confidence levels for the position and mobility vectors of all relevant objects, possibly together with a level of confidence that objects have been correctly identified and classified and that no "ghost" objects are in the context model.

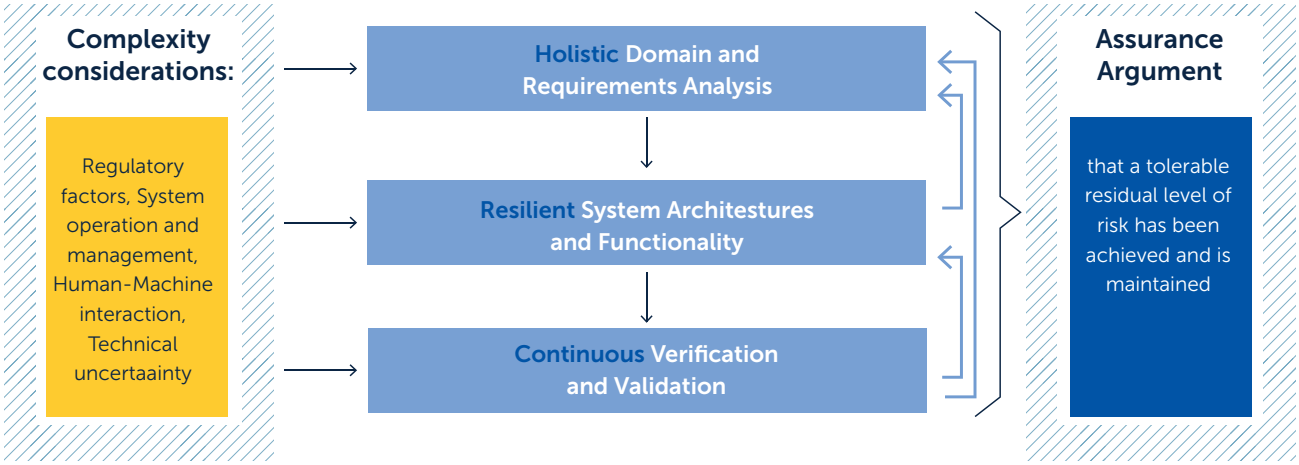

Figure 2. Complexity-aware systems safety engineering.

Complexity. We refer to complexity in terms of systems theory, where a system is defined as complex if the interaction between its parts leads to behavior that could not be predicted by considering the individual parts and their interactions alone. Complexity can manifest itself within different levels of a system:

- **Increasing complexity within the E/E architecture.** This complexity is caused by not only the increasing number of technical components within a system but also the heterogeneity and technical implementation of these components, the use of components and software of unknown pedigree, and changes in the system after release due to software updates or the integration of additional services (e.g., via cloud connectivity). One impact of system complexity is nonlinearity, mode transitions, and tipping points where the system may respond in unpredictable ways depending on its current state or context.

- **Complex behavioral interactions between systems, self-organization, and ad hoc systems-of-systems.** Interactions between a system and its environment may be difficult to predict, particularly when human agents are involved in the interactions. Consider the range of behaviors that must be considered by a self-driving vehicle navigating heavy traffic consisting of human-driven vehicles and automated vehicles from other manufacturers acting according to unharmonized norms of behavior. Such interactions may lead to ad hoc systems-of-systems forming, over which individual manufacturers have little or no control and thus call into question whether the system scope under consideration for safety is appropriate and what an appropriate scope of analysis should be.

Increasing complexity increases the difficulty of determining the (potential) causes of failures in the system and effective risk control measures. The impact of complexity in the systems and their environment on our ability to deliver convincing arguments for safety has been discussed in more detail within the scope of automated driving [7]. The concept of uncertainty is closely related to the topic of increasing complexity. Again, uncertainty can manifest itself in several ways that make the safety assurance of safety-critical cognitive systems more challenging.

1. **Scope and unpredictability of the operational domain.** Many highly automated CPSs can be said to operate within an open context, that is, an environment that cannot be fully specified in a way that desirable system behavior can be defined for each possible set of conditions. Such environments are typified by the presence of edge cases or "black swans," corresponding to previously unknown or even unknowable conditions. Furthermore, the operational domain can shift over time, leading to new sets of conditions that were neglected during design. This neglect inevitably leads to insufficiencies in the resulting specification of the system under development, which are referred to as "ontological" uncertainties [8].

2. **Inaccuracies and noise in sensors and signal processing.** This complex, unpredictable environment is measured using a combination of inevitably imperfect sensors providing a noisy, incomplete view of the environment. In addition to general inaccuracies in the measurements, such sensors themselves can be "fooled" by physical properties of the environment, such as a lens flare distorting a video image or manhole covers leading to spurious radar reflections.

3. **Uncertainties in the perception and decision-making functions.** The complex, incomplete, and noisy inputs to the system are often the motivation for using AI and ML techniques in the first place. However, as we will explain in further chapters, these algorithms themselves introduce additional uncertainty within the system and rarely deliver precise results. Therefore, in an attempt to solve the problem of uncertainties in the inputs to the system, yet another class of uncertainties is introduced.

The complexity of the system and associated uncertainties lead to semantic gaps [9], which are defined as a discrepancy between the intended and specified functionality and can be caused by the complexity and unpredictability of the operational domain, the complexity and unpredictability of the system itself, as well as the increasing transfer of decision functions to the system, which would otherwise require non-specifiable properties, such as human intuition or ethical judgment. These semantic gaps lead to insufficiencies in the definition of appropriate safety acceptance criteria as well as a lack of confidence that statements made in a safety assurance case reflect the actually achieved safety of the system.

The above discourse illustrates the manifold challenges we face when developing safe cognitive systems. It also allows us to better delimit discussions around "safe AI." To derive an adequate set of safety assurance methods for such systems, we must be clear about which problems we are addressing. These problems can be roughly separated into the following categories:

1. **Safety challenges caused by the inherent difficulty of the task to be solved.** This category includes the systematic complexity of the function to be implemented using AI components based on the complexity and unpredictability of the input domain and the resulting impact on semantic gaps, which may restrict our ability to define an adequate specification of the required performance of the AI-based function. These factors are independent of the actual AI or ML techniques used and are better referred to as cognitive system safety engineering activities. These activities include the application of suitable system safety assurance methods, including the definition of socially and legally tolerable risk acceptance criteria, as the development of an overall system design that is resilient to previously unknown or changing properties of the environment. A "complexity-aware" system safety engineering approach is summarized in Figure 2.

2. **Safety challenges caused by the use of specific AI/ML techniques.** This category includes performance limitations and properties of the specific AI/ML techniques used. For example, statistical modeling and linear regression-based models exhibit different sets of properties related to the explainability and predictability of their results as deep neural networks but may differ greatly regarding their accuracy for certain tasks. An example of how the properties of a specific ML technique can support a safety

assurance case can be found in [10]. The AI-technology specific challenges therefore involve ensuring that the specific performance requirements allocated to the AI-based function within the system context are fulfilled with a level of confidence commensurate to the overall level of system risk.

When discussing "safe AI" and the associated challenges, we should therefore be clear about which scope we are referring to. Are we referring to the safety of cognitive systems operating within an open context or to whether specific properties of a trained model remain within certain bounds of uncertainty for a given set of inputs? These two topics are closely interrelated. For example, when applying ML techniques that deliver a high level of prediction uncertainty or sensitivity to small changes in the inputs, such as deep neural networks, the cognitive systems engineering task must ensure that tolerances on uncertainties within the trained model must be carefully defined and aligned with other system components.

Safety Engineering

Traditional safety engineering ultimately is based on fallback mechanisms to a responsible human operator, the deterministic behavior of a cyber-physical system as the basis for its testability, and well-defined operating contexts. In addition, current safety certification regimes require correct and complete specifications before operation. These basic assumptions of safety engineering no longer pertain to AI systems for the following reasons:

1. With increased autonomy, a fallback mechanism to a human is often not possible anymore. Indeed, the EB system needs to perform without any human intervention, as the required reaction times are well below the capabilities of human beings.
2. AI systems make their own knowledge-based judgments and decisions. While added flexibility, resilience, elasticity, and robustness of cognitive AI systems are clearly important, the gains in these dimensions come at the loss of testability due to the admittance of nondeterminism.[9] This disadvantage is costly because systematic testing and simulation are still the single most used technique for verifying the correct functioning of software-intensive systems.
3. AI systems need to cope with operating environments in which comprehensive monitoring and controlling is impossible and in which unpredictable events may occur. In fact, AI systems are mainly used for situations where the full details of the operating context cannot be known in advance. Risk estimation is therefore difficult to perform for AI systems using conventional techniques.

For all these reasons, well-established and successful safety standards for software-intensive systems, including DO 178C in the aerospace industry and ISO 26262 in the automotive industry, cannot readily be applied to AI systems. Indeed, these safety standards barely heed autonomy and the particularly advanced software technologies for system autonomy [11].

Many industrial initiatives for development are in progress, such as lower levels of automated driving[10] and certified AI algorithms in the medical domain.[11] These endeavors, however, are incomplete in that

[9] A NASA study of a software-based control of vehicle acceleration, for example, revealed, among other things, potential race conditions in sensor readings due to asynchronous access by a multiplicity of threads. This study concluded that the software was "untestable", preventing the possibility of eliminating unsafe control actions [124].

[10] Safety first guidance for potential methods and considerations with the intention of developing safe L3-L4 automated driving functionality, including an ANN [105].

[11] https://www.fda.gov/medical-devices/software-medical-device-samd/artificial-intelligence-and-machine-learning-aiml-enabled-medical-devices#resources

they are based on prescriptive safety standards.[12] However, we still need to determine adequate methodologies and end-to-end verification technology to assure safe autonomy. In addition, we need to gain more practical experience with these approaches before prescribing them as "good" or even "best" practices in industrial standards. Current safety engineering standards are also based on the idea that the correct behavior must be completely specified and verified before operation. It is therefore unclear if and how these safety standards may apply to learning-enabled systems, which are continuously self-adapting and optimizing their behavior to ever-changing contextual conditions and demands, based on their experience in the field.

If the current safety engineering methodology is not directly applicable to AI systems, then we might ask ourselves if we can at least reduce the problems of safe and learning-enabled control on a case-by-case basis, including the following examples:

- Depending on the application context, safety engineers can restrict AI-based functionality with the intent of increasing controllability or decreasing severity/exposure, thereby decreasing associated safety risks.

- Uncertainty due to open-ended operating contexts and the safe control thereof is dramatically reduced in current automotive practice by collecting all types of possible driving scenarios using real and virtualized global ecosystems of vehicles.[13] This approach basically tries to close off the set of possible driving scenarios as the basis for constructing the equivalent of a "digital rail." Because of the many possible anomalies ("black swans"),[14] however, this approach is rather resource-intensive, and it is unclear how to determine that enough scenarios have been collected to sufficiently cover the space of all driving scenarios.

- AI-based functionality is complemented by a safe control channel, thereby effectively combining the intended performance of AI-based systems with the safety of more traditional control.[15] The crucial element in this safety architecture is a switch between the performant AI-based channel and the safe channel, which is based on runtime monitoring of crucial safety specifications. A pervasive runtime monitor, for example, checks that the proposed action of the EB yields safe behavior. This pervasive runtime monitor, together with the switching logic between the two channels, is developed with tradi-tional safety engineering methods, thereby effectively removing the AI component from a safety-critical path.

- An engineer may also decide to discontinue initial AI-based proof-of-concepts by reverting to well-understood control techniques altogether, for example, if an end-to-end safety concept for a given AI-based functionality is too costly or not possible and if a sufficiently performant and safe system may be achieved by more traditional means. The underlying phenomenon of technical debt for data-driven systems in real-life engineering has already been described [12].

Based on these types of engineering design decisions for reducing the safety relevance and increasing the determinacy of AI systems using traditional prescriptive safety engineering techniques for AI may

[12] "If you do … then the system is safe."
[13] For example, https://www.pegasusprojekt.de/en/home.

[14] A well-known example is the fatal crash of a car in autonomous mode that resulted from a very rare four-factor combination of a white truck against a brightly lit sky, along with the truck height and angle versus the car (https://www.tesla.com/blog/tragicloss), (https://www.ntsb.gov/news/press-releases/Pages/PR20170912.aspx).

[15] This combination is sometimes called a "Simplex" architecture.

sometimes also be possible. However, this reductionist approach is restricted to a rather small class of functionally automated systems with added machine learning-based capabilities, which do not adequately support the key concepts of AI systems, namely, autonomy and self-learning. Methods for reducing safe AI problems to the currently prevailing prescriptive safety engineering standards therefore are not future proof,[16] because prescribed and fixed verification and validation process activities, criteria, and metrics do not work well for assuring the safety of AI systems [13].
Overarching properties [14] have recently been proposed as a product-based alternative to prescriptive safety engineering standards such as DO 178C. Informally, a system is safe for operation if and only if the CIA conditions hold:

- the system does what it should do under foreseeable operating conditions (Correctness);
- what the system should do is properly captured (Intent); and
- the system does not cause unacceptable harm (Acceptability).

An assessment of whether a system possesses these properties might be based on an explicit assurance case.[17] Overarching properties are therefore flexible enough to be adapted to developing a justified belief of system safety with learning-enabled components. Currently, however, the overarching CIA properties do not seem to have been adopted for safety certification on a larger scale.
Taken together, traditional safety engineering is reaching a turning point, moving from deterministic, non-evolving systems operating in well-defined contexts to self-adaptive and self-learning systems that are acting increasingly autonomous and in largely unpredictable operating contexts. However, we currently do not have an adequate safety engineering framework for designing this upcoming generation of safety-related AI systems.[18]

In the following section, we therefore outline a novel approach to safe AI engineering. This approach is based on uncertainty quantification for the multitude of sources of uncertainties in AI systems.
The overarching goal of managing uncertainties is to minimize uncertainty in the system behavior, thereby increasing confidence up to tolerable levels in the safe behavior of the AI system.
The underlying idea is to generalize the notion of determinacy in traditional safety engineering to uncertainty. As a special case, if some system behaves with no uncertainty, then it is fully predictable and deterministic. Deterministic parts of an AI system can (and probably should) therefore still be developed with well-proven design and verification techniques for establishing correctness and possible perfection.

Instead of fallback to responsible (human) operators, uncertainty measures are used by autonomous control strategies to minimize surprises and safely explore largely unknown territory. The EB assistant, for example, might be rather uncertain about the precise location of some relevant car, and it therefore initiates additional perceptive capabilities with the intent of decreasing uncertainty, thereby increasing its confidence, in the location of the respective car to a sufficient level, as the basis for deciding on a safe sequence of EB actions. Moreover, some indirect cues [15] cause the system to hypothesize the presence of a relevant car, which needs to be confirmed by additional actions before initiating emergency breaking. These examples demonstrate that uncertainty is not only a design but also an essential runtime artifact for the situational generation of safe control behavior.

Uncertainties in the proposed engineering framework are explicitly managed through safety cases. These structured arguments are supported by a body of evidence that provides a compelling, comprehensible, and valid case that a system is safe for a given application and operating environment.[19] In contrast to

[16] The applicability and limits of prescriptive safety standards to autonomous AI systems are also discussed in [126].
[17] See also the Subsection entitled Assurance-based Uncertainty Estimat.
[18] See, for example, [129] for a survey of the safety certification of systems with learning-enabled components.
[19] Def Stan 00-56 Issue 3, Safety Management Requirements for Defense Systems, Ministry of Defense, 2007.

largely process-based traditional prescriptive approaches, a safety argument based on safety cases is largely product-based, as it involves presenting evidence that the actually developed system is safe, as opposed to merely showing that it was developed using normatively prescribed "good" practice. Recent quantitative extensions to safety and assurance cases provide the basis for assigning and combining uncertainties for the central ingredients, such as evidence, arguments, assumptions, and conclusions. Figure 3, for example, illustrates the top-level plan for constructing a safety case for an autonomous GNC,[20] which is built upon a traditional 3-level autonomous architecture. The modular construction of this safety case is based, among others, on evidence from traditional verification of planning components, verification of the correctness or quasi-predictability of neural network components for perceptive tasks, and runtime monitoring for central safety properties. This GNC also includes an FDIR[21] component for detecting and recovering from unforeseen and potentially hazardous events. The overall goal is to develop an autonomous spacecraft, say, for landing on an asteroid, together with a complete safety case. In addition, the safety case is also used to generate safe landing behavior and safely handle unforeseeable events.

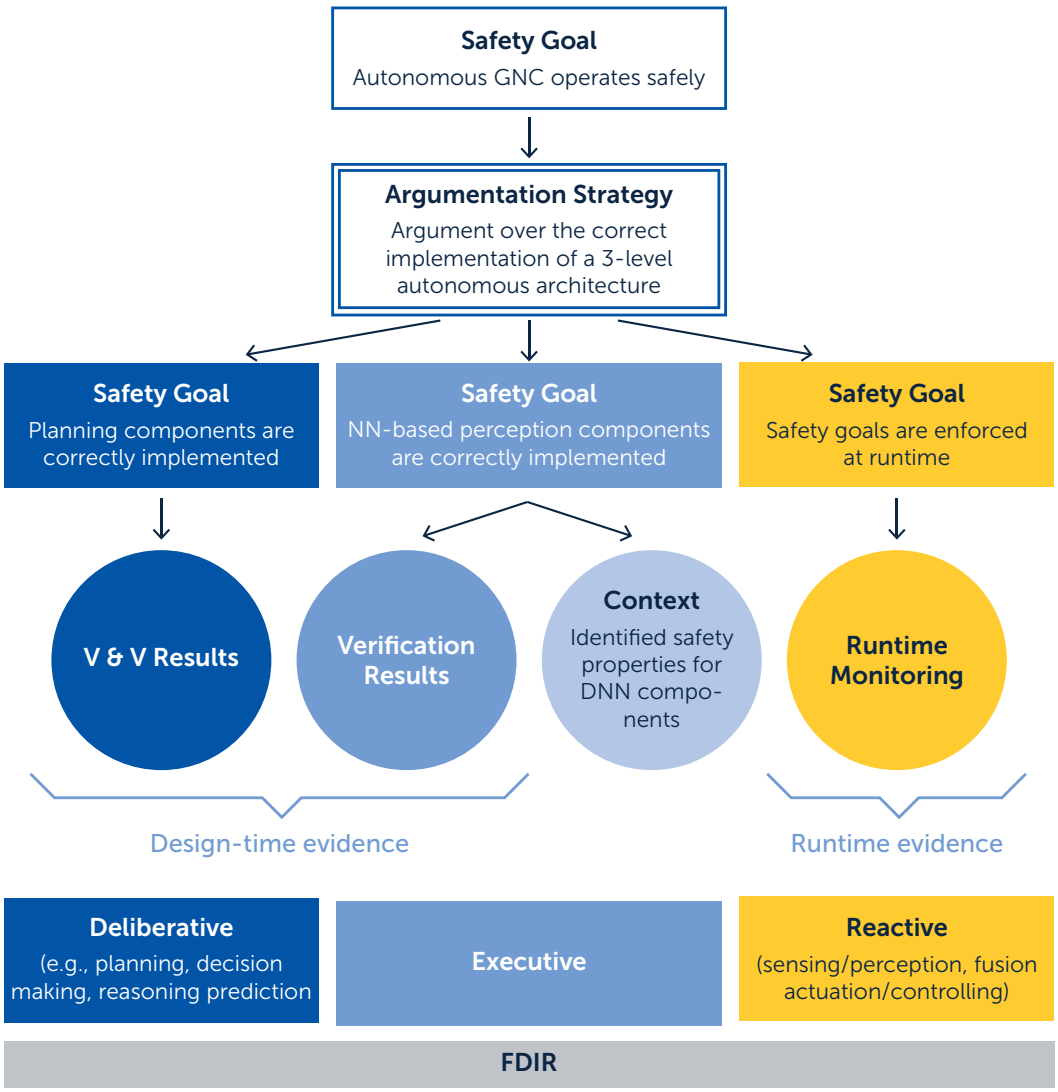

Figure 3. Safety case from a 3-layer architecture of a cognitive system.

[20] Guidance, Navigation, and Control.
[21] Failure Detection, Isolation, and Recovery.

3. Specification

In the first step, we need to express in precise terms when an AI system may be considered safe. This step sounds deceptively easy because data-driven AI is particularly successful in application contexts in which obtaining concise specifications (say, translation of natural language) is difficult, if not impossible.
In addition, the operating contexts of AI systems typically are complex, uncertain, and largely unpredictable. Safety hazards are subject to change during operation, and the presence of human operators and their interaction with the sense-plan-act control loop further complicates matters in that even well-intended human interaction may lead to unsafe behavior. In summary, the adaptive, flexible, and context-sensitive behavior of AI systems causes unpredictability and emergent unsafe behavior, which was not necessarily specified, intended, or consciously implemented during system design.

In the following section, we distinguish between the safety specifications of AI systems and the derived specifications of the learning-enabled components of these systems. In addition, we discuss ways to systematically derive requirements for individual components of a system from overall safety requirements.

System Safety Specification

The overall system's safety specification is often described in terms of safety envelopes. These envelopes may be considered an underapproximation of the states (or scenarios) of possible operating contexts that are sufficiently safe.[22] In a slightly more general setting, we also quantify the uncertainty of environment states.

A common approach for specifying safety envelopes is based on maximizing underapproximations, thereby also maximizing the number of known safe behaviors. In other words, the operating context is partitioned into the known safe (the safety envelope), the known unsafe, the unknown safe, and the unknown unsafe, and the goal is to maximize the known safe areas by minimizing the known unsafe areas and discovering as many new unsafe scenarios as possible for a given level of effort [16].[23] In addition, safety envelopes must be continually adapted to ever-changing operating contexts, safety hazards, and safety requirements.

The safety objective of an automated EB system, for example, is to maintain a minimum safe distance between the ego car and environmental objects. More generally, the responsibility-sensitive safety (RSS) model offers provable safety for vehicle behaviors, such as the minimum safe following distance [17].

Safety envelopes such as RSS have also been extended to address worst-case variability and uncertainty [18]. Safety envelopes usually are highly nonlinear and context dependent, as is the case of Kamm's friction ellipses.

In these cases, ML techniques based on minor component analysis are promising for synthesizing safety envelopes from safe behaviors [19]. These techniques open the possibility of self-learning and safe maintenance of safety envelopes through cautious[24] and safe exploration.

We still have very little knowledge on how to systematically construct safety envelopes. So the challenge is to construct and maintain safety envelopes that are known safe states of the operational context and to maximize the safety envelope for a given level of effort. The safety of certain states contains uncertainty.

[22] According to a risk and safety analysis of the system under consideration.
[23] A newer version is available at https://www.beuth.de/de/norm-entwurf/iso-dis-21448/335355102.
[24] Cautious behavior might be realized using minimizing surprises, which may be realized by minimizing free energy or maximizing predictive information.

Operational contexts, system behavior, and the notion of acceptable risk are constantly evolving. Therefore, safety envelopes must be continuously adapted to these ever-changing conditions and requirements. Depending on the degree of autonomy, such AI systems need to self-maintain corresponding safety envelopes, possibly including online risk analysis.

Component Safety Specifications

The behavior of ML components is specified using data. In the supervised learning of ANNs, for example, a set of interpolation points is used to specify the input-output behavior of the intended function. Inputs may be observations of the operational context, and outputs are corresponding context models.

The central challenge for data-driven requirements engineering is to represent the operational context using selected interpolation points as faithfully as possible for a given level of effort. This task can be accomplished by sampling input scenarios from the (assumed) distribution of the operational context or by discretizing the operational context according to n features into 2^n cells and sampling inputs from these cells. Both approaches are prohibitively expensive for many interesting operational contexts.
A series of polynomial-time approximations, for example, has been developed to make feature-based discretization feasible in practical applications [20], and clustering and unsupervised learning techniques are used to identify a finite number of representative classes of scenarios [21]. Also of interest would be constructing sets of interpolation points that, when used to train a perception ANN, are sufficient for establishing the invariance of the AI system with respect to a given safety envelope.

Given a set of hopefully representative interpolation points and an initial neural network architecture, an ANN is constructed by heuristically searching, usually based on hill-climbing, for a configuration to minimize the error between actual and specified outputs. This type of heuristic search might get stuck in local minima, thereby leading to suboptimal solutions, or it might not terminate. Moreover, the resulting ANN might be incorrect in that its input-output behavior does not coincide with the specifying interpolation points, and the ANN might not be resilient in that slight variations of inputs lead to completely different output behavior, because many ANNs tend to "overfit" without further precautions in training. Consequently, ANN output can often be altered by adding relatively small perturbations to the input vector [22].

These types of uncertainties motivate the need for requirements of ANNs beyond data, such as resilience. For a fixed input x and a metric d on the input space, an ANN is locally ε-resilient if ANN(x') is equal to or similar to the output ANN(x) for all small perturbations $x'$with $d(x, x') < \varepsilon$; if the ANN is locally ε-resilient for all possible inputs, then it is also globally ε-resilient [23] [24]. The inherent uncertainty of ANN input-output behavior is considerably reduced by establishing strong resilience properties.

The additional desired behavior of perception components usually comes with their intended functionality. For example, an object classifier may be expected to correctly classify certain affine or homeomorphic images of training inputs, such as stretching, squeezing, rotational, and translational images. Sequences of context models for modeling traffic flow, for example, should also obey the fundamental laws of physics.
The challenge is to identify and maintain desired properties and potential defects of ANN-based perception components, which lead to undesired behavior. Moreover, a much better understanding of the contribution of these properties and defects in the overall system safety is needed. This type of knowledge should enable the development engineer, for example, to compute precise bounds on the required resilience of the perception ANN for arguing overall system safety.

Deriving Component Safety Specifications

Given a safety specification S of the AI system depicted in Figure 1, we derive the corresponding safety constraints on the possible behavior of the ANN-based perception component. The control, consisting of the deliberation and the execution units, needs to ensure that the output (the changed environment) is safe, that is, in S. In an engineered system, we can compute the weakest precondition, say, $\mathrm{wp}(\mathrm{controller})(S)$, which now serves as the postcondition for the perception unit. Assuming that the input x to the perception unit is in S, that is, that this state is safe, we obtain a pre/postcondition specification

$$(\forall\, x \in S)\ \mathrm{perception}(x) \in \mathrm{wp}(\mathrm{controller})(S)$$

of the perception unit that is sufficient to establish the safety of the overall AI system loop.[25] Adequate domain abstractions and corresponding abstract interpretation techniques are needed to make this approach feasible.

Indeed, researchers have taken the first step in this direction and identified special cases of pre/post-condition pairs for neural networks [25] [26]. Logical specifications θ may also be incorporated into the training purpose of an ANN by constructing, for example, a corresponding differentiable loss function $L(\theta)$, such that x (logically) satisfies θ whenever $L(\theta)(x) = 0$ or by incorporating constraints such that θ will be satisfied by the model even on unseen data [27]. More generally, in the case of mutual dependencies between the perception unit and the controller for realizing active perception or in the case of learning-enabled controllers, sufficiently strong preconditions for these two components can be synthesized based on, for example, a combination of traditional assume-guarantee reasoning and ML [28] [29]. Instead of using sets of states as properties and state transformers between these properties, one can also build uncertainty directly into the computational model of an AI system. In these cases, the behavior of AI systems may, at least partially and when necessary, be based on probabilistic sets, where states belong to a set with a certain probability only, and probabilistic transformers between probabilistic sets. The classical notions of weakest precondition and strongest postcondition generalize to probabilistic set transformers.

Whenever a few interactions occur between the perception and the control unit, as is expected in many real-time systems, the weakest precondition approach above is applied to the unrolled system. These types of pre/postcondition specifications for the perception unit are the basis for largely decoupling perception development from the control unit. For example, as long as the controller adapts in time such that $\mathrm{wp}(\mathrm{controller}')(S) \subseteq \mathrm{wp}(\mathrm{controller})(S)$, where controller' is the updated controller, component-wise safety analyses will still compose to a system-level safety argument; otherwise, the challenge is to identify corresponding minimal sets of changes for the perception unit and its analysis. The perception specification can also be used as additional input to train the perception an ANN or as the basis for verifying this component, for example, for systematically deriving test cases. These initial ideas for systematically deriving component safety, particularly for learning-enabled components from overall systems safety requirements, clearly need to be further developed and stress-tested on challenging real-world AI systems.

[25] Notice that the precondition $x \in S$ is in the language of environment inputs, whereas the postcondition is in the language of the context models.

Component Safety Verification

Furthermore, one may compute the weakest precondition of the perception ANN. For example, computing the weakest preconditions of ReLu networks with their rather simple node activation functions is, in principle, straightforward [30]. Now, given a safety envelope S, the safety verification problem for an AI system (perception; control) may be stated as

$$\mathrm{wp}(\mathrm{perception})\big(\mathrm{wp}(\mathrm{controller})(S)\big) \subseteq S$$

This fundamental safety invariant immediately reduces to the local constraint for the perception unit:

$$\mathrm{perception}(S) \subseteq \mathrm{wp}(\mathrm{controller})(S)$$

These types of constraints are statically analyzed based on symbolic verification techniques [31] used for test case generation or dynamically checked using runtime verification (see Section 5).

In addition, the perception component may now be trained with the additional knowledge that its precondition is S and the postcondition is $\mathrm{wp}(\mathrm{controller})(S)$. Logical constraints can also be interpreted in a more general quantitative logic to obtain a differentiable objective function as needed for hill-climbing-based training. Such quantitative interpretation may, for instance, be based on probabilistic sets and probabilistic transformers for modeling.
If we manage to train a "correct" ANN, then we obtain a safety-by-design method for constructing safe AI systems. Indeed, as mentioned above, an ANN may be trained to obey some given logical safety property by constructing a corresponding differentiable loss function for the satisfiability of this formula. Nonetheless, the safe behavior and input-output still contain uncertainty due to the incorrectness of underlying learning algorithms. A new generation of knowledge-enhanced ML [32] techniques is tackling such real-world challenges for ML algorithms.

4. Uncertainty Quantification

Learning in the sense of replacing specific observations with general models is an inductive process. Such models are never provably correct but only hypothetical and therefore uncertain, and the same holds for the predictions produced by a model.

In addition to the uncertainty inherent in inductive inference, other sources of uncertainty exist, including incorrect model assumptions and noisy or imprecise data. Correspondingly, one usually distinguishes between aleatoric and epistemic sources of uncertainty [33] [34]. Whereas aleatoric[26] uncertainty refers to the variability in the outcome of an experiment that is due to inherently random effects, epistemic[27] uncertainty refers to uncertainty caused by a lack of knowledge. In other words, epistemic uncertainty refers to the ignorance of an actor, and hence to its epistemic state, and can in principle therefore be reduced with additional information. Various approaches toward robustness are taken based on reducing uncertainty [33]. Uncertainty reduction also plays a key role in active learning [35] and learning algorithms such as decision tree induction [36].

Indeed, sources of uncertainty in the design of safe AI systems are multitudinous [37]. There is, among other things, uncertainty about the operational context, about hazards and risks, about the correctness and generalizability of learning-enabled components, about safety envelopes, there is uncertainty due to noise in sensing, controller uncertainty due to nondeterminism and/or probabilistic control algorithms, uncertainty on the internal models of the controller, and, last but not least, uncertainty about the actions of human operators and their possible interaction with the AI-based control system.

Rigorous approaches for safe AI need to manage the multitude and heterogeneity of sources of uncertainty. We therefore propose an engineering approach based on the principle of uncertainty reduction, thereby increasing the predictability (up to tolerable quantities) of the AI system. The crucial steps are as follows:

- Identify all[28] relevant sources of uncertainty.
- Quantify and estimate the uncertainty,[29] including the certainty thereof.
- Forward and inverse propagation of uncertainty along chains[30] of computation.
- Modular composition of uncertainties along the architectural decomposition[31] of the AI system.
- Design operators to mitigate the overall system uncertainty below a certain level as determined by a risk and safety analysis,[32] including

    - a combination of offline and online accumulation of relevant knowledge for managing epistemic sources of uncertainty, and
    - incremental change in uncertainty reasoning due to self-learning or even self-modification capabilities of an AI system.

[26] AKA statistical, experimental, or "known unknown".
[27] AKA systematic, structural, or "unknown unknown".
[28] In a defeasible manner.
[29] Uncertainty quantification is the science of quantitative characterization and reduction of uncertainties in computational and real-world applications. Among others, it tries to determine how likely certain outcomes are if some aspects of a system are not exactly known.
[30] Including recursive chains.
[31] Both horizontal and vertical.
[32] For example, less than one hazardous behavior for 10^9 operational time.

Clearly, these tasks for managing the multitude of heterogeneous sources of the uncertainty in AI systems are fundamental in any rigorous and transparent engineering process. We currently do not have, however, a comprehensive set of methods and tools for supporting application engineers in managing uncertainties.

Environmental Uncertainty

The operational environment of AI systems can be rather complex,[33] with considerable uncertainty even about the number and type of objects and agents, human and robotic, that are in the environment, let alone about their intentions, behaviors, and strategies [38]. An AI system therefore must act without relying on a correct and complete model of the operating environment. The models at hand usually do not faithfully reflect the real-world operational context,[34] and it is simply not possible, and possibly not even desirable, to model everything. To address modeling errors, AI systems may make distributional assumptions about the operational environment. However, exactly ascertaining the underlying distribution can be difficult.

As an alternative to explicitly modeling the operational environment, this environment is commonly specified using a set of scenarios, which should be sampled with respect to the underlying distribution of the environment. These scenarios are analyzed and labeled with their respective interpretation of the context model to obtain training data for an ANN-based perception unit. Selecting "good" scenarios is a major challenge. These scenarios should significantly reduce the difference between the assumed, underlying distribution of the operating environment and the distribution of the selected training set. For example, collecting scenarios by driving around for five hours on a highway in Alaska does not contribute as well to the approximation of real-world driving as collecting driving scenarios at the Gate of India. Another concern is evolving operating scenarios and how to correspondingly adapt the set of specifying scenarios.

The challenge is to quantify and measure uncertainty between the operating environment and its specifying set of scenarios, identify "good" scenarios for reducing uncertainty to tolerable levels, provide sufficient conditions on the uncertainty of scenario sets for overall system safety (up to quantifiable tolerances as identified through safety risk assessment), and adapt the specifying scenario set to the evolving operating environment.

Behavioral Uncertainty

We restrict our considerations on learning-enabled components to a widely popular class of ANNs. This ANN is a deterministic function. Because of nonlinear activation functions, however, its input-output behavior contains considerable uncertainty: Training instances may or may not be represented correctly by the ANN, and it is usually unclear how, and how much, the input-output behavior of an ANN generalizes from training instances. The success of one-pixel attacks serves as a reminder of the limited generalizability and resilience of some machine-learned models. Establishing the resilience [23] or invariance properties — for example, invariance with respect to certain affine or homeomorphic transformations — of an ANN is an important means of reducing uncertainty in the input-output behavior. Some uncertainty about outcomes, however, remains. A systematic framework for analyzing different sources of uncertainty for ANNs is described in [39].

[33] Operational design domains may be specified following standards such as PAS 1883 (https://www.bsigroup.com/en-GB/CAV/pas-1883).
[34] Again, the old saying applies: All models are wrong, but some might be useful.

Measuring behavioral uncertainty. Entropy may be used to quantify the uncertainty of a neural network. Indeed, under mild assumptions on uncertainty, entropy is the only possible definition of uncertainty [40], at least in its aleatoric interpretation. Behavioral uncertainty has a multitude of indicators. The work in [41], for example, proposes using the distance between neuron activations observed during training and the activation pattern for the current input to estimate input-output behavior uncertainty.

Training-based estimation of behavioral uncertainty. Ensembles of neural networks, for example, estimate predictive uncertainty by training a certain number of NNs from different initializations and sometimes on differing versions of the dataset. The variance in the ensemble's predictions is interpreted as its epistemic uncertainty. Instances of ensemble learning techniques, such as Bayesian neural networks (BNNs) [42], measure epistemic uncertainty $P(\theta|D)$ on model parameters θ and the aleatoric uncertainty $P(Y|X, \theta)$.
In fact, the predicted uncertainty of BNNs is often more consistent with observed errors compared to classical neural networks. The out-of-training distribution points of a BNN lead to high epistemic uncertainty. The uncertainty $P(\theta|D)$ can be reduced with more data. BNNs are also an interesting approach to active learning, as one can interpret the model predictions and see if, for a given input, different probable parametrizations lead to different predictions. In the latter case, the labeling of this input will effectively reduce the epistemic uncertainty.

Uncertainty Propagation

What we really should care about is not freedom from faults but an absence of failure [43]. Particularly, if a perception unit fails to meet its safety specification, then we call this unit faulty, and if the overall cognitive system loop fails to act safely, then a system failure occurs. Using the corresponding random variables Faulty and $\mathrm{Failure}$, we are interested in the probability that the system is safe, that is, $\mathrm{P}(\mathrm{not\ Failure})$; using Bayes' rule, we obtain:

$$\mathrm{P}(\mathrm{Failure} \mid \mathrm{Faulty}) * \mathrm{P}(\mathrm{Faulty}) = \mathrm{P}(\mathrm{Faulty} \mid \mathrm{Failure}) * \mathrm{P}(\mathrm{Failure})$$

Provably worst-case distributions [44] are used to estimate the posterior probability $\mathrm{P}(\mathrm{Failure} \mid \mathrm{Faulty})$ of faulty behaviors leading to safety violations.[35] The probability $\mathrm{P}(\mathrm{Faulty})$ that the perception unit is faulty is approximated, for instance, using a training-based estimate of behavioral uncertainty (as described above) or from an assurance-based estimate of uncertainty (as described below). Now, assuming that all but the perception unit are possibly perfect and that the faulty perception unit is the only possible cause of failure, $\mathrm{P}(\mathrm{Faulty}|\mathrm{Failure}) = 1$. Consequently, we can estimate $\mathrm{P}(\mathrm{not\ Failure}) = 1 - \mathrm{P}(\mathrm{Failure})$ using Bayesian inference.
This short exposition of the propagation of component faults to system safety failures is intended to demonstrate a possible style of Bayesian inference for establishing safety results. The underlying methodology, however, should also be applicable for more general mutually recursive system architectures.

[35] Failure and Faulty are random variables, and the conditional probability P(Failure | Faulty) measures the uncertainty that a system is unsafe (Failure) given that the perception unit violates its specification (Faulty).

Assurance-based Uncertainty Estimation

The goal of rigorous design is to gain sufficient confidence that failures, in our case safety violations, are very rare up to tolerable quantities. However, sufficient confidence cannot be constructed by considering failures only.
Instead, assurance constructs a convincing case that failures are rare. One widely quoted definition of the corresponding notion of a safety case comes from [45]: "A safety case is a structured argument, supported by a body of evidence that provides a compelling, comprehensible, and valid case that a system is safe for a given application in a given operating environment." An assurance case is simply the generalization of a safety case to properties other than safety.

An assurance case, therefore, is a comprehensive, defensible, and valid justification of the safety of a system for a given application in a defined operating context. It is based on a structured argument of safety considerations across the system lifecycle, which can assist in convincing the various stakeholders that the system is acceptably safe.

The purpose is, broadly, to demonstrate that the safety-related risks associated with specific system concerns[36] have been identified, are well-understood, and have been appropriately mitigated and that mechanisms are in place to monitor the effectiveness of safety-related mitigations. In this sense, an assurance case is a structured argument for linking safety-related claims through a chain of arguments to a body of the appropriate evidence. One of the main benefits of structured arguments in assurance cases is to explicitly capture the causal dependencies between claims and the substantiating evidence.

Altogether, assurance cases are the basis for judging that a technical system is acceptable for widespread use. Assurance cases also determine the level of scrutiny needed to develop and operate acceptably safe systems. More specifically, assurance cases determine constraints on the design, implementation, verification, and training strategies, and they demonstrate the contributions of corresponding artifacts and activities to the overall system safety.

One may be confident in this assurance based on "the quality or state of being certain that the assurance case is appropriately and effectively structured, and correct" [46]. A necessary aspect of gaining confidence in an assurance case is addressing uncertainty, which, as we have seen above, may have several sources. Uncertainty, often impossible to eliminate, nevertheless undermines confidence and must therefore be sufficiently bounded.

Recent extensions of assurance cases for reasoning about confidence and uncertainty [47] are a good starting point for estimating and managing aleatoric and epistemic uncertainties for safe AI systems. In particular, probability theory has been proposed for quantifying confidence and uncertainty [48], and epistemic uncertainty is quantified through the Dempster–Shafer theory of beliefs or Bayesian analysis [49], the use of Bayesian belief networks [50] [51] [52], Josang's opinion triangle [47], evidential reasoning [53], and weighted averages [54].

However, a slight problem arises with quantifying confidence in assurance case arguments, as proposed methods on Bayesian belief networks, Dempster–Shafer theory, and similar forms of evidential reasoning can deliver implausible results [55, 46]. Without strong evidence that the quantified confidence assessments are indeed trustworthy, there is no plausible justification for relying on any of these techniques in

[36] Including safety and security but also applying to all the other attributes of trustworthiness.

safety engineering. Alternatively, one may also look toward a value for the probability of perfection — based on extreme scrutiny of development, artifacts, and code — which is then related to confidence [56] [57]. Qualitative approaches toward uncertainty, on the other hand, focus on the reasoning and rationale behind any confidence by constructing an explicit confidence argument. For example, eliminative induction is increasing confidence in assurance cases by removing sources of doubt and using Baconian[37] probability to represent confidence [58]. Eliminative induction first identifies potential sources of doubt, so-called defeaters, and then works toward removing them or proving their irrelevancy.

The search for defeaters, and their possible defeat, should be systematized and documented as essential parts of the case [59]. One systematic approach is through construction and dialectical consideration of counterclaims and countercases. Counterclaims are natural in confirmation measures as studied in Bayesian confirmation theory, and countercases are assurance cases for negated claims.

Assurance cases have successfully been applied to many safety-critical systems, and they are also flexible enough to be adopted in systems with learning-enabled components. An overall assurance framework for AI systems with an emphasis on quantitative aspects, e.g., breaking down system-level safety targets to component-level requirements and supporting claims stated in reliability metrics, has recently been out-lined [60]. Requirements on assurance cases for autonomously acting vehicles with learning-enabled components are addressed, for example, by UL 4600.[38]

A mixture of requirements and data-centric metrics together with corresponding verification techniques, static and dynamic [61], is needed to establish the safety of AI systems with ML components. A successful element in a successful deployment of safety assurance for AI systems is a library of pre-validated argument steps [62] [63, 64] together with adequate operators for instantiating and composing specific system-specific assurance cases from these pre-validated structured arguments. We also hypothesize that because of the multitude of sources of uncertainty, assurance arguments for increasingly autonomous AI systems need to (1) stress rigor in assessing the evidence and reasoning employed and (2) systematize and automate the search for defeaters, the construction of cases and counter cases, and the management and representation of dialectical examination.

Increased rigor and automation in building and maintaining assurance cases should enable productive interaction with tools for logical and probabilistic reasoning and formal argumentation. Using frameworks such as STPA [65] to better capture and examine a component's control actions in relation to the larger system-level safety contexts may be beneficial. How the influence of learning-enabled components is captured and reasoned within the AI control structure is of particular interest. Finally, rigorous assurance cases open new possibilities for online self-adaptation of safety arguments for determining safe behavior when operating in uncertain contexts because they can be adapted, quickly and efficiently, to the ever-changing safety considerations of AI systems.

[37] https://ntrs.nasa.gov/api/citations/20160013333/downloads/20160013333.pdf
[38] https://ul.org/UL4600

5. Analysis

A key issue for AI systems is rigorous safety analysis, which is based on a mixture of well-known verification and validation techniques, with safety verification of learning-enabled components. Here, we focus on novel aspects of analyzing AI systems with ANN-based perception units only.

What do we need to verify about ANN components to support AI system safety? Our starting point here is the component requirements as obtained by breaking down application-specific systems safety requirements to verification and validation requirements for the individual components of AI systems.

Because of mounting concerns about using ANNs for safety-related applications, new techniques have emerged to increase the trustworthiness of ANNs [66] [67] [68] [64]. A survey of this ever-growing number of methods and technologies is well beyond the scope of these notes. Indeed, individual methods are not lacking, but the safety relevance of many ANN analysis techniques (such as adversarial analysis) is questionable, particularly when the impact of the overall system within which the ANN is used is unclear [69].

What is needed is a systematic evaluation of individual analysis techniques. A central challenge is to adequately measure and quantify how well and under which circumstances they improve confidence in the safe system behavior. A first step in this direction is provided in [70], which develops a safety pattern for choosing and composing analysis techniques based on how they contribute to identifying and mitigating systematic faults known to affect system safety. More generally, given an ANN and desired properties, we therefore define the goal of ANN analysis to improve confidence or, dually, reduce uncertainty, if the desired properties hold up to tolerable quantities on the ANN.

Testing

The goal of ANN testing is to generate a set of test cases that can demonstrate confidence in an ANN's performance, when passed, such that the ANN can support an assurance case. Usually, test case generation is guided by structural and nonstructural coverage metrics [71].

Traditional structural coverage criteria from software testing usually cannot be applied directly to ANNs. For example, neuron coverage is trivially fulfilled in ANNs using a single test case. Moreover, MC/DC, when applied to ANNs, may lead to an exponential (in the number of neurons) number of branches to be investigated and therefore is not practical, as typical ANNs comprise millions of neurons. As usual in testing, the balance between the ability to find bugs and the computational cost of test case generation is essential for the effectiveness of a test method [72].

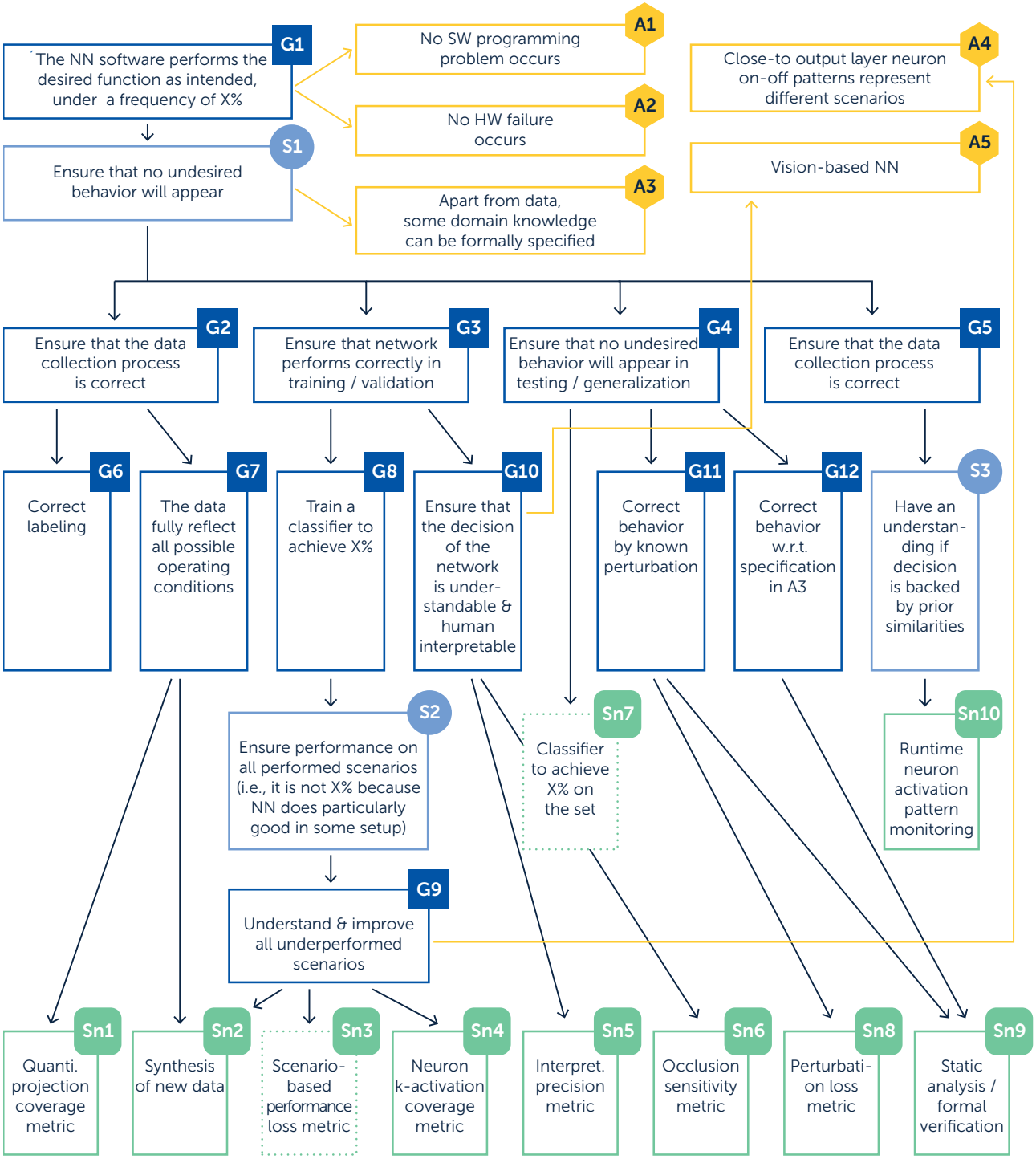

Figure 4. NNDK-based assurance case.

Generating falsifying/adversarial test cases is generally performed using search heuristics based on gradient descent or evolutionary algorithms [73, 74] [75] [76]. These approaches may be able to find falsifying examples efficiently, but they usually do not provide an explicit level of confidence about the nonexistence of adversarial examples when the algorithm fails to find one.

The work in [20] developed ANN-specific nonstructural test coverage criteria for the robustness, interpretability, completeness, and correctness of an ANN. A scenario coverage metric, for example, partitions the possible input space according to N attributes (e.g., snow, rain, ...), and proposes, based on the existing work on combinatorial testing, efficient k-projection (for k = 0,...,N–1) coverage metrics as approximations of the exponential number of input partitions. In principle, a "complete" (with respect to the available input data) set of attributes may be obtained through unsupervised learning or clustering methods. These coverage metrics are implemented in the NNDK testing toolkit for ANNs [77].

In [78], coverage is enforced to finite partitions of the input space, relying on predefined sets of application-specific scenario attributes. In a similar vein, the "boxing clever" technique focuses on distributing training data and divides the input domain into a series of representative boxes.

Many traditional test case generation techniques, such as fuzzing [79, 80, 75] [81], symbolic execution [82], concolic testing [83], mutation testing [84], and metamorphic testing [85], have been extended to support the verification of ANNs. Despite their effectiveness in discovering various defects of ANNs together with their data-centric requirement specifications, it is not exactly clear how testing-based approaches can be efficiently integrated into the construction of convincing safety argumentations for AI systems. A possible step in this direction, however, is the NNDK-based safety case in Figure 4, which makes the contribution and the rationale behind individual test metrics in establishing safety goals more explicit.

Altogether, testing methods seem to be effective at discovering the defects of ANNs. It is unclear, however, how to measure the effectiveness of test coverage metrics in constructing sufficient confidence — or dually, raising doubts — in a convincing assurance case. In addition, most testing-based approaches assume a fixed ANN. However, ANNs are learning-enabled and trained continuously on new data/scenarios. The challenge is to invent methodologies for efficiently — and depending on the application context, in real time — retesting safety requirements for continuously evolving ANNs. This retesting methodology could be based on adapting corresponding assurance cases.

Instead of validating individual learning-enabled components, the idea of scenario-based testing is to (1) automatically or manually identify a reasonably small set of relevant dynamic situations or scenario types; (2) check if the set of scenario types is complete; and then (3) derive system-specific tests for each scenario type. The need for a test-ending criterion immediately arises based on the following question: did we test all scenario types? In addition, did we sufficiently test each type with specific instances? The general approach to scenario-based testing is outlined in Figure 5. It is based on automated clustering of real driving data and completeness checks for the clusters thus obtained [86].

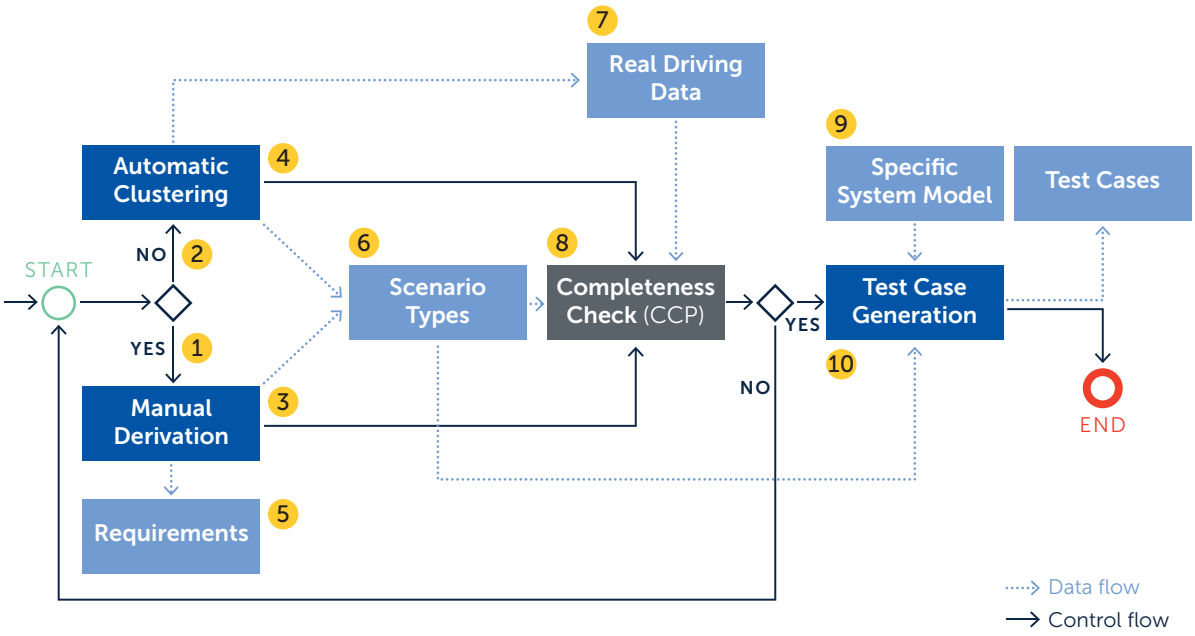

Figure 5. Scenario-based testing.[39]

Symbolic Verification

Safety verification problems for ANNs can be reduced to constraint solving problems, such as satisfiability in propositional logic [87] [88], satisfiability modulo theories [89] [90] [91] [92], and mixed-integer linear programming [23]. These approaches typically do not scale up to the size of real-world ANNs with millions of neurons. Approximation techniques are applied to improve efficiency but usually at the expense of precision. Recent approaches based on global optimization potentially can address larger networks [93]. Compositional verification techniques for scaling up ANN safety verification are largely missing. For the assume-guarantee style of reasoning applied to verifying an ANN-based automotive safety controller, however, see [94].

Because symbolic safety verification technologies work on a model of an ANN, they might have certain defects due to implementation issues (for example, rational numbers vs. IEEE floating-point implementations). In addition, how to efficiently apply these techniques to continuously changing ANNs is unclear.

Runtime Verification

In runtime verification, a monitor observes the concrete execution of the system in question and checks for violations of stipulated properties. When the monitor detects a violation of a property, it notifies a command module, which then isolates the cause of the violation and attempts to recover from the violation. In this way, runtime verification is a central element of FDIR-based[40] fault-tolerant systems.

[39] Adapted from: https://doi.org/10.1109/ITSC.2019.8917326

[40] Fault Detection, Isolation, and Recovery.

For the multitude of sources of uncertainty in AI systems, stringent real-time requirements, and ever-changing learning-enabled components, runtime verification is an essential element for the safety verification of AI systems.

System requirements of the form "the system must perform action a within n seconds of event e" are common in the runtime monitoring of autonomous systems [95]. These types of properties are expressible in suitable sub-logics of metric temporal logic, such as GXW [96] [97, 52] and the timed extensions thereof [98]. These types of specifications are compiled into (timed) synchronous dataflows as the basis for efficient runtime monitors. A dynamic programming and rewriting-based algorithm for monitoring MTL formulas is described in [99]. Moreover, architectural design principles for monitoring distributed systems are needed to ensure that monitoring does not perturb the system (at least, not too much) [74]. In particular, the tutorial [100] discusses the challenges of instrumenting real-time systems so that the timing constraints of the system are respected. A recent tutorial describes state-of-the-practice technology for generating runtime monitors that capture the safe operational environment of systems with AI/ML components [101].

Altogether, runtime verification is an essential and attractive technique of any verification strategy for safe AI. Unlike static verification techniques, such as testing or symbolic verification, adaptation to learning-based components, such as ANNs, is unnecessary. In this way, runtime monitoring is an enabling verification technology for continuous assurance, based on the MAPE-K[41] loop from autonomic computing.
The main challenge in deploying runtime monitoring, as with any other cyber-physical system, is to embed monitors in an efficient (for example, energy-efficient) way without perturbing the behavior of the AI system too much.

Runtime monitoring may also be used to measure uncertainties in the input-output behavior of ANNs. For example, if an input is out-of-distribution of the training set, then one may conclude that the "correctness" of the corresponding ANN output may be doubtful. Such information about the uncertainty of a perception result may be useful input for planning in the deliberation stage. Uncertainty information about the perception unit is also used in Simplex architectures to switch to a safe(r) perception channel whenever the ANN output is doubtful. Clearly, the distance (in some given metric) of the input to the training input set may serve as a measure of the uncertainty of the input-output behavior of an ANN. However, this measure returns zero uncertainty even for the "incorrect" behavior of the ANN on training inputs. Alternatively, [102] proposes to monitor the neuronal activation pattern of some input and to compare it with neuronal activation patterns as learned during the ANN training phase. This measure of the input-output behavior certainty of an ANN is part of the assurance case for the ANN in Figure 4. In addition, applicable background knowledge and physical laws may also be used to monitor the plausibility of the input-output behavior of an ANN.

In summary, because of the multitude of sources of uncertainty, the complexity of AI-based systems and the environments in which they operate, even if all the challenges for specification and verification are solved, one will likely be unable to prove unconditional safe and correct operation. Situations in which we do not have a provable guarantee of correctness will always arise. Therefore, techniques for achieving fault tolerance and error resilience at run time must play a crucial role. There is, however, not yet a systematic understanding of what can be achieved at design time, how the design process can contribute to the safe and correct operation of the AI system at run time, and how the design-time and runtime techniques can interoperate effectively.

[41] Measure, Analyze, Plan, Execute; the K stands for knowledge.

6. Safety-by-Design

Validation and verification activities are usually complemented with safety-by-construction design steps. We briefly describe some of the main challenges and initial approaches toward safety-by-design, namely, property-driven synthesis of learning-enabled components, compositional construction of AI subsystems and systems, and safety architectures for AI systems. The goal in this respect is a fundamental set of building blocks together with composition and incremental change operators for safety-by-construction design and continual assurance of large classes of AI systems.

Property-driven Synthesis

Instead of using a posteriori verification of desirable properties of ANNs via static or dynamic verification technologies as outlined above, can we design, from scratch, a ML component that provably satisfies (possible in a robust interpretation) given formal specifications? For example, given the pre- and post-conditions of an ANN, as obtained from breaking system safety envelopes down to individual learning-enabled components, can an ANN that satisfies the given safety specification be trained? Given a property expressed in logic, for example, one constructs a corresponding differentiable loss function for property-driven training of the ANN. In this way, property-driven synthesis needs to, among other things, design an appropriate training set, set up the initial structure of the ANN, and choose and adjust appropriate hyper-parameters for training. The selection of training sets and training is then guided by reducing an adequate measure of the uncertainty so that the ANN indeed satisfies the given specification.

Progress is needed along all these fronts. Techniques of neuro-symbolic computation [103] [104] may be a good starting point, as they also try to integrate high-level reasoning with low-level perception such that neuro-symbolic methods have the pure neural, logical, and probabilistic methods as special cases. A short history and perspectives of knowledge-augmented ML are described in [32].

Compositional System Design

The triad of perception, deliberation, and execution, as depicted in Figure 1, is the simplest possible architecture of an AI system. Often, deliberation and execution units are complex and mutually dependent for realizing a fine-grained control; perception may also depend on deliberation, say, in AI systems with active perception. Moreover, each stage of an AI system triad is usually decomposed into any number of functional units, including monitors and safe channels. For example, deliberation may include functionalities for modeling AI capabilities such as interpretation and prediction, model building, derivation of knowledge, goal management, or planning, and perception is decomposed into a pipeline of tasks for, say, internal and external state estimation, sensor fusion, object recognition, and object classification. This real-world architecture for realizing an autonomous driving function can be found, for example, in [105].[42]

Traditional Simplex architectures [106] are used to address the performance and safety requirements of many automated and autonomous systems [107] [108] [109] [110] by leveraging runtime assurance, where the results of design-time verification are used to build a system that monitors itself and its environment

[42] This publication advocates the use of state-of-the-practice dependability and safety engineering methodologies as prescribed in current industrial safety standards[42] for saving SAE L3 and L4 automated driving capabilities.

at run time. More precisely, a Simplex architecture comprises (1) a performant controller under nominal operating conditions, which is designed to achieve high performance, but it is not provably safe, (2) a safe controller that can be pre-certified to be safe, and (3) a decision module that is pre-certified (or safe-by-design) to monitor the state of the controlled system and its operational environment to check whether desired system safety specifications can be violated. If so, the decision module switches control from the nominal monitor to the safe monitor. A provably safe composition of Simplex architectures is developed in the context of Soter [111], which also allows for switching to nominal control to minimize performance penalties while retaining strong safety guarantees.

Although compositional design operators have been developed for digital circuits and embedded systems, we do not yet have such comprehensive theories for AI systems. For example, if two ANNs are used for perception on two types of sensors, say LiDAR and a camera, and individually satisfy their specifications under certain assumptions, under what conditions can they be used together to decrease perception uncertainty? More generally, how can we compositionally design safe and predictable perception pipe-lines? How can one design planning and deliberation components for overcoming the inherent limitations of their ANN-based perception component? How can one design execution components for minimizing surprises in uncertain environments? Additionally, how can these components interact in a safe and quasi-predictable[43] manner?

[43] That is, predictable up to acceptable levels.

Table 1. Safe AI Engineering Challenges.

Specification Challenge	• Provide the means for constructing (and maintaining) safety envelopes, either deductively from safety analysis or inductively from safe nominal behavior • Provide the means for minimizing uncertainties related to safety envelopes with a given level of effort • Provide the means for deriving safety requirements for learning-enabled components, which are sufficient for establishing AI system safety • Provide the means for reducing specification uncertainty using deriving data requirements for learning-enabled components
Uncertainty Challenge	• Identify all relevant sources of uncertainty for an AI system • Provide adequate means for measuring uncertainty • Calculate forward propagation of uncertainty, where the various sources of uncertainty are propagated through the model to predict overall uncertainty in the system response • Identify and solve the relevant inverse[44] uncertainty quantification problems for safe AI • Predict (up to tolerable quantities) unsafe behavior of AI systems operating in uncertain environments
Assurance Challenge	• Provide adequate measures of uncertainty for assuring AI system safety • Construct and maintain evidence-based arguments for supporting the certainty and for rebutting the uncertainty of safety claims • Identify useful safety case patterns[45] for safe AI systems and identify corresponding operators for instantiating and composing these patterns
Design Challenge	• Develop safety case patterns for different architectural designs of AI systems[46] • Compositionally construct safe and quasi-predictable AI systems together with their safety cases
Analysis Challenge	• Provide adequate means for measuring and reducing uncertainty in the input-output behavior of learning-enabled components • Define and measure the respective contribution of static and dynamic analysis techniques for learning-enabled systems to reduce safety-related uncertainty to tolerable levels
Maintenance Challenge	• Identify incremental change operators for maintaining the uncertainty and safety assurance of self-learning AI systems • Safely adapt and optimize the situational behavior of an AI system (together with its safety cases based on the principle of minimizing uncertainty)

[44] That is, calculating from a set of observations the causal factors that produced them.
[45] Cmp. AMLAS
[46] In analogy to, say, Mils separation kernel protection profile.

7. Conclusions

We have been arguing that traditional safety engineering is unsuited for developing and operating AI systems. On the basis of this insight, we outlined a safety engineering methodology for AI that is centered around managing and assuring uncertainty to acceptable levels [112, 76] as the basis for predictable (up to acceptable tolerances) and safe AI systems.

The proposed rigorous design methodology for safe AI is based on the central notion of a safety case for managing uncertainties. Our proposals are compatible with the emergent standard UL4600[47] on required properties for safety cases. In some sense, the depicted design methodology may also be considered an uncertainty-based amalgam of the paradigms of data — with model-driven design.

The main contribution lies in identifying the core challenges and possible research directions for the specification, design, analysis, assurance, and maintenance of safe AI (for a summary, see Table 1). This list, however, is incomplete, as we have omitted, for instance, all-important systems challenges due to interactive control between human operators and machine-based control.

The identified challenges for safe AI, as listed in Table 1, do not seem insurmountable. The overarching challenge rather lies in integrating individual methods into a coherent and comprehensive engineering framework for systematically managing and reducing uncertainty to tolerable quantities and demonstrating its relative merits in real-world AI systems.

We have been working toward AI safety engineering, among others, with Fasten ([113, 20] for checkable safety cases, evidential transactions in Evidentia/CyberGSN for continual assurance and compliance [114], the neural network dependability kit [77] for analyzing ANNs, and risk-based safety envelopes for autonomous vehicles under perception uncertainty [115]. We are also currently working on concrete safe AI use cases to integrate these individual engineering nuggets and to elaborate on a generally useful approach for safe AI engineering. We hypothesize that uncertainty quantification also plays an increasingly prominent role in analyzing and certifying complex software systems because traditional notions of system-level correctness are becoming less applicable for heterogeneous and ever-evolving software landscapes.

There are related ideas on uncertainty quantification in engineering [80] for certifying that, with high probability, a real-valued response function of a given physical system does not exceed a given safety threshold. Uncertainty quantification also plays a pivotal role in minimizing uncertainties for ANNs [116]. We expect these types of techniques to provide a mathematical underpinning of a design calculus for safe AI.

The ultimate goal in this respect is a rigorous engineering framework based on pre-certified parameterized components, corresponding assurance arguments, and system composition operators (for example, for watchdogs, monitors, and redundant channels) from which complete systems and corresponding assurance cases are constructed in a property-guided, traceable, and optimized manner.
In addition, onboard management of uncertainty is used to design safe exploration strategies of unknown territory based on the principle of managing uncertainty and to minimize surprises. This type of safe exploration of an AI system might even be complemented with an online risk and safety assessment, together with corresponding online updates of safety cases and the uncertainty quantifications thereof.

[47] https://edge-case-research.com/ul4600/

References

[1] Rushby, "Quality measures and assurance for AI software," in NASA Contractor Report 4187, 1988.

[2] Rodd, Safe AI - is this possible?, Elsevier, 1994.

[3] Amodei, Olah, Steinhardt, Christiano, Schulman and Mané, Concrete problems in AI safety, 2016.

[4] Bhattacharyya, Cofer, Musliner, Mueller and Engstrom, Certification considerations for adaptive systems, IEEE, 2015.

[5] Geisberger and Broy, Living in a networked world: Integrated research agenda Cyber-Physical Systems (agendaCPS), Herbert Utz Verlag, 2015.

[6] Pinker, "Rationality," 2021.

[7] S. Burton, J. McDermid, P. Garnet and R. Weaver, "Safety, Complexity, and Automated Driving: Holistic Perspectives on Safety Assurance," IEEE Computer, vol. 54, no. 8, pp. 22-32, 2021.

[8] R. Gansch and A. Adee, "System theoretic view on uncertainties," in Design, Automation & Test in Europe Conference & Exhibition (DATE), 2020.

[9] S. Burton, I. Habli, T. Lawton, J. McDermid, P. Morgan and Z. Porter, "Mind the gaps: Assuring the safety of autonomous systems from an engineering, ethical, and legal perspective," Artificial Intelligence, vol. 279, 2020.

[10] S. Burton, I. Kurzidem, A. Schwaiger, P. Schleiß, M. Unterreiner, T. Graeber and P. Becker, "Safety Assurance of Machine Learning for Chassis Control Functions," in International Conference on Computer Safety, Reliability, and Security, York, U.K., 2021.

[11] Blanquart, Fleury, Hernek, Honvault, Ingrand, Poncet, Powell, Strady-Lécubin and Thévenod, Software Product Assurance for Autonomy On-Board Spacecraft, ESA SP-532, 2003.

[12] Sculley, Holt, Golovin, Davydov, Phillips, Ebner and Young, Machine learning: The high interest credit card of technical debt, http://research.google/pubs/pub43146/, 2014.

[13] Alves, Bhatt, Hall, Driscoll, Murugesan and Rushby, Considerations in assuring safety of increasingly autonomous systems, NASA, 2018.

[14] Holloway, "Understanding the Overarching Properties: First Steps," 2018.

[15] Björkman, Internal cue theory: Calibration and resolution of confidence in general knowledge, 1994.

[16] SOTIF, ISO/PAS 21448:2018, Road Vehicles – Safety of the Intended Functionality, Draft, 2017.

[17] Shalev-Schwartz, Shammah and Shashua, Vision Zero: Can Roadway Accidents be Eliminated without Compromising Traffic Throughput, https://export.arxiv.org/abs/1901.05022, 2018.

[18] Koopman, Osyk and Weast, Autonomous vehicles meet the physical world: Rss, variability, uncertainty, and proving safety, Springer, 2019.

[19] Tiwari, Dutertre, Jovanović, d. Candia, Lincoln, Rushby and Seshia, Safety Envelope for Security, 2014.

[20] Cheng, Huang, Nührenberg and Ruess, "Towards dependability metrics for neural networks," 16th ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE), 2018.

[21] Hauer, Gerostathopoulos, Schmidt and Pretschner, Clustering traffic scenarios using mental models as little as possible, IEEE, 2020.

[22] Su, Vargas and Sakurai, One pixel attack for fooling deep neural networks, 2019.

[23] Cheng, Nührenberg and Rueß, Maximum resilience of artificial neural networks, Springer, 2017.

[24] Seshia, Desai, Dreossi, Fremont, Ghosh, Kim and Yue, Formal Specification for Deep Neural Networks, Springer, 2018.

[25] Dutta, Jha, Sanakaranarayanan and Tiwari, Output range analysis for deep neural networks, arXiv:1709.09130, 2017.

[26] Dvijotham, Stanforth, Gowal, Mann and Kohli, A dual approach to scalable verification of deep networks, arXiv:1803.06567, 2018.

[27] Goyal, Dumancic and Blockeel, "SaDe: Learning Models that Provably Satisfy Domain Constraints," in arXiv:2112.00552, 2021.

[28] Giannakopoluou, Păsăreanu and Barringer, Assumption generation for software component verification, IEEE, 2002.

[29] Păsăreanu, Gopinath and Yu, Compositional verification for autonomous systems with deep learning components., Springer, 2019.

[30] Sotoudeh and Thakur, A Symbolic neural network representation and its application to understanding, verifying, and patching networks, 2019.

[31] Gopinath, Converse, Pasareanu and Taly, Property Inference for Deep Neural Networks, IEEE, 2019.

[32] Sagel, Sahu, Matthes, Pfeifer, Qiu, Rueß, Shen and Wörmann, Knowledge as Invariance -- History and Perspectives of Knowledge-augmented Machine Learning, 2020.

[33] T. Dietterich, Steps Toward Robust Artificial Intelligence, 2017.

[34] Hüllermeier and Waegeman, Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods,, 2021.

[35] Aggarwal, Kong, Gu, Han and Philip, Active learning: A survey, CRC Press, 2014.

[36] Mitchell, The need for biases in learning generalizations, 1980.

[37] Weyns, Bencomo, Calinescu and et.al., "Perpetual Assurances for Self-Adaptive Systems".

[38] Seshia, Sadigh and Sastry, Towards Verified Artificial Intelligence, 2016.

[39] Czarnecki and Salay, Towards a framework for managing perception uncertainty for safe automated driving, Springer, 2018.

[40] Robinson, Entropy and Uncertainty, 2008.

[41] Cheng, Nührenberg and Yasuoka, "Runtime monitoring neuron activation patterns," 2019.

[42] Jospin, Buntine, Boussaid, Laga and Bennamoun, Hands-on bayesian neural networks - a tutorial for deep learning users, 2020.

[43] Rushby, The Indefeasibility Criterion for Assurance Cases, 2020.

[44] Zhao, Littlewood, Povyakalo, Strigini and Wright, Modelling the probability of failure on demand (pfd) of a 1-out-of-2 system in which one channel is “quasi-perfect”, 2017.

[45] UK Ministry of Defence:, Defence Standard 00-56, Issue 4: Safety Management Requirements for Defence Systems. Part 1: Requirements., 2007.

[46] Grigorova and Maibaum, Taking a page from the law books: Considering evidence weight in evaluating assurance case confidence, IEEE, 2013.

[47] Duan, Rayadurgam, Heimdahl, Ayoub, Sokolsky and Lee, Reasoning About confidence and uncertainty in assurance cases: A survey, 2014.

[48] Bloomfield, Littlewood and Wright, Confidence: its role independability cases for risk assessment., 2007.

[49] Swiler, Paez and Mayes, Epistemic uncertaintyquantification tutorial, 2009.

[50] Littlewood and Wright, The use of multilegged arguments to increase confidence in safety claims for software-based systems: A study based on a BBN analysis of an idealized example., IEEE, 2007.

[51] Zhao, Zhang, Lu and Zeng, A new approach to assessment of confidence in assurance cases., Springer, 2012.

[52] Dennney, Pai and Habli, Towards measurement of uncertainty in safety cases, IEEE, 2011.

[53] Nair, Walkinshaw, Kelly and d. l. Vara, An evidential reasoning approach for assessing confidence in safety evidence, 2015.

[54] Yamamoto, Assuring security through attribute GSN, 2015.

[55] Graydon and Holloway, An investigation of proposed techniques for quantifying confidence in assurance arguments, 2017.

[56] Rushby, Formalism in safety cases., 2010.

[57] Rushby, Logic and epistemology in safety cases., 2013.

[58] Goodenough, Weinstock and Klein, Toward a theory ofassurance case confidence., Carnegie-Mellon University, 2012.

[59] Bloomfield and Rushby, Assurance 2.0: A manifesto, 2020.

[60] Zhao, Huang, Bharti, Dong, Cox, Banks, Wang, Schewe and Huang, "Reliability assessment and safety arguments for machine learning components in assuring learning-enabled autonomous systems," in arXiv:2112.00646, 2021.

[61] McDermid, Jia and Habli, Towards a framework for safety assurance of autonomous systems, 2019.

[62] Bloomfield and Netkachova, Building blocks for assurance cases, IEEE, 2014.

[63] Hawkins, Paterson, Picardi, Jia, Calinescu and Habli, Guidance on the assurance of machine learning in autonomous systems (AMLAS), 2021.

[64] Houben, Abrecht, Akila, Bär, Brockherde, Feifel and Woehrle, Inspect, understand, overcome: A survey of practical methods for AI safety, 2021.

[65] Abdulkhaleq, Wagner and Leveson, "A comprehensive safety engineering approach for software-intensive systems based on STPA," in Procedia Engineering, 2015.

[66] Huang, A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability, arXiv:1812.08342v5, 2020.

[67] Xiang, Musau, Wild, Lopez, Hamilton, Yang and Johnson, Verification for machine learning, autonomy, and neural networks survey, 2018.

[68] Schwalbe and Schels, A Survey on Methods for the Safety Assurance of Machine Learning Based Systems, 2020.

[69] Dreossi, Jha and Seshia, Semantic adversarial deep learning, Springer, 2018.

[70] Cârlan, Gallina, Kacianka and Breu, Arguing on software-level verification techniques appropriateness, Springer, 2017.

[71] Chen, Yan, Wang, Kang and Wu, Deep neural network test coverage: How far are we?, arXiv:2010.04946v2, 2014.

[72] Sun, Huang, Kroening, Sharp, Hill and Ashmore, Testing deep neural networks, 2019.

[73] Goodfellow, Shlens and Szegedy, Explaining and harnessing adversarial examples, 2014.

[74] Goodloe and Pike, Monitoring distributed real-time systems: a survey and future directions, National Aeronautics and Space Administration, Langley Research Center, 2010.

[75] Papernot, McDaniel, Jha, Fredrikson, Celik and Swami, The limitations of deep learning in adversarial settings, IEEE, 2016.

[76] Carlini and Wagner, Towards evaluating the robustness of neural networks, IEEE, 2017.

[77] Sahu, Vállez, Rodríguez-Bobada, Alhaddad, Moured and Neugschwandtner, Application of the neural network dependability kit in real-world environments, 2020.

[78] Cheng, Huang and Yasuoka, Quantitative projection coverage for testing ml-enabled autonomous systems, Springer, 2018.

[79] Odena, Olsson, Andersen and Goodfellow, TensorFuzz: Debugging neural networks with coverage-guided fuzzing, PMLR, 2018.

[80] Owhadi, Scovel, Sullivan, McKerns and Ortiz, Optimal uncertainty quantification, 2013.

[81] Xie, Ma, Juefei-Xu, Chen, Xue, Li and See, Coverage-guided fuzzing for deep neural networks, 2018.

[82] Gopinath, Wang, Zhang, Pasareanu and Khurshid, Symbolic execution for deep neural networks, arXiv:1807.10439, 2018.

[83] Sun, Wu, Ruan, Huang, Kwiatkowska and Kroening, Concolic testing for deep neural networks, IEEE, 2018.

[84] Shen, Wan and Chen, MuNN: Mutation analysis of neural networks, IEEE, 2018.

[85] Zhang, Zhang, Liu and Khurshid, Deep-Road: GAN-based metamorphic autonomous driving system testing, IEEE, 2018.

[86] Hauer, Schmidt, Holzmüller and Pretschner, Did we test all scenarios for automated and autonomous driving systems?, IEEE, 2019.

[87] Cheng, Nührenberg, Huang and Ruess, Verification of binarized neural networks via inter-neuron factoring, Springer, 2018.

[88] Narodytska, Kasiviswanathan, Ryzhyk, Sagiv and Walsh, Verifying properties of binarized deep neural networks, 2018.

[89] Huang, Kwiatkowska, Wang and Wu, Safety verification of deep neural networks, Springer, 2017.

[90] T. Pulina, An Abstraction-Refinement approach to verification of artificial neural networks, 2010.

[91] Katz, Barrett, Dill, Julian and Kochendoerfer, Reluplex: An efficient SMT solver for verifying deep neural networks, Springer, 2017.

[92] Tuncali, Ito, Kapinski and Deshmukh, Reasoning about safety of learning-enabled components in autonomous cyber-physical systems, 2018.

[93] Ruan, Huang and Kwiatkowska, Reachability analysis of deep neural networks with provable guarantess, 2018.

[94] Cheng, Huang, Brunner and Hashemi, Towards safety verification of direct perception neural networks, IEEE, 2020.

[95] Kane, Chowdhury, Datta and Koopman, A case study on runtime monitoring of an autonomous research vehicle (ARV) system, Springer, LNCS, 2015.

[96] Cheng, Hamza and Rueß, Structural Synthesis for GXW Specifications, Springer, LNCS, 2016.

[97] Cheng, Lee and Rueß, Autocode4: Structural controller synthesis, Springer, LNCS, 2017.

[98] Xin and Rueß, Actor-based Synthesis for Timed GXW, 2021.

[99] Thati and Roşu, Monitoring algorithms for metric temporal logic specifications., 2005.

[100] Bonakdarpour and Fischmeister, Runtime-monitoring of time-sensitive systems, Springer, 2011.

[101] Torfah, Junges, Fremont and Seshia, "Formal Analysis of AI-Based Autonomy: From Modeling to Runtime Assurance," in International Conference on Runtime Verification, pp. 311-330..

[102] Cheng, Nührenberg and Yasuoka, Runtime monitoring neuron activation patterns, IEEE, 2019.

[103] d. Raedt, Manhaeve and Dumancic, Neuro-Symbolic = Neural + Logical + Probabilistic, 2019.

[104] Riegel and et.al., "Logical neural networks," arXiv:2006.13155, 2020.

[105] Aptiv, Audi, Baidu, BMW, Continental, Daimler, FCA, HERE, Infineon, Intel and Volkswagen, Safety first for automated driving, https://www.daimler.com/documents/innovation/other/safety-first-for-automated-driving.pdf, 2019.

[106] Sha, Using simplicity to control complexity, IEEE, 2001.

[107] Schiemann, DeVore, Richards, Gandhi, Cooper, Horneman, Stoller and Smolka, Runtime assurance framework development for highly adaptive flight control systems,, Charlottesville: Barron Associates Inc., 2001.

[108] Bak, Manamcheri, Mitra and Caccamo, Sandboxing controllers for cyber-physical systems,, IEEE, 2011.

[109] Phan, Yang, Clark, Grosu, Schierman, Smolka and Stoller, A component-based simplex architecture for high-assurance cyber-physical systems, arXiv:1704.04759, 2017.

[110] Aswani, Bouffard and Tomlin, Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter, IEEE, 2012.

[111] Desai, Ghosh, Seshia, Shankar and Tiwari, SOTER: a runtime assurance framework for programming safe robotics systems, IEEE, 2019.

[112] Chechik, Salay, Viger, Kokali and Rahimi, Software Assurance in an Uncertain World, Springer, 2019.

[113] Cârlan and Ratiu, FASTEN.Safe: A model-driven engineering tool to experiment with checkable assurance cases, Springer, International Conference on Computer Safety, Reliability, and Security.

[114] Beyene and Carlan, CyberGSN: A semi-formal language for specifying safetyc, IEEE, 2021.

[115] Bernhard, Hart, Sahu, Schöller and Cancimance, Risk-Based Safety Envelopes for Autonomous Vehicles Under Perception Uncertainty, 2021.

[116] Abdar, Pourpanah, Hussain, Rezazadegan, Liu, Ghavamzadeh and Nahavandi, A review of uncertainty quantification in deep learning: Techniques, Applications and Challenges, 2021.

[117] W. Putzer, Trustworthy Autonomous/Cognitive Systems: A Structured Approach, ISSN 2699-1217, 2020.

[118] Balta, Sellami, Kuhn, Schöpp, Buchinger and Baracaldo, Accountable Federated Machine Learning in Government: Engineering and Management Insights, Springer, 2021.

[119] Bojarski, d. Testa, Dworakowski, Firner, Flepp, Goyal and Zieba, "End-to-end learning for self-driving cars," arXiv:1604.07316, 2016.

[120] Amini, Azari, Bhaskaran, Beauchamp, Castillo-Rogez, Castano and Wyatt, Advancing the scientific frontier with increasingly autonomous systems, 2020.

[121] Dreossi, Ghosh, Sangiovanni-Vincentelli and Seshia, A formalization of robustness for deep neural networks, 2019.

[122] C. Hammerschmidt, ISO 26262 is not perfectly designed for Artificial Intelligence, https://www.eenewsautomotive.com/news/iso-26262-not-perfectly-designed-artificial-intelligence, 2019.

[123] LeCun, Bottou, Bengio and Haffner, Gradient-based learning applied to document recognition, IEEE, 1998.

[124] NASA Engineering and Safety Center, National highway traffic safety administration Toyota unintended acceleration investigation., 2011.

[125] Nguyen, Destercke and Hüllermeier, Epistemic uncertainty sampling, Springer, 2019.

[126] Salay and Czarnecki, Using machine learning safely in automotive software: An assessment and adaption of software process requirements in ISO 26262, 2018.

[127] Seshia, Desai, Dreossi, Fremont, Ghosh, Kim and Yue, "Formal Specification for Deep Neural Networks," International Symposium on Automated Technology for Verification and Analysis (ATVA), pp. 20-34, 2018.

[128] Rushby, "Assurance for Increasingly Autonomous Safety Critical Systems," 2018.

[129] Tambon, Laberge, An, Nikanjam and et.al., "How to Certify Machine Learning Based Safety-critical Systems?," in Arxiv 2107.12045, 2021.

Imprint

Published by
fortiss GmbH
Guerickestraße 25
80805 München

Layout/Gestaltung
fortiss GmbH

ISSN Print
2699-1217

ISSN Online
2700-2977

2nd issue
January 2023

Photo credits
Title: AdobeStock_434584931
Page16/28: AdobeStock
Page 36: Fortiss GmbH @ Kathrin Kahle

Find here more
fortiss White Paper

Safe Intelligence — this forms the core brand of the **Fraunhofer Institute for Cognitive Systems IKS**. Connected cognitive systems drive innovation in many sectors, for example in autonomous vehicles, medical devices or intelligent automation within industry. They should always take full advantage of the potential offered by artificial intelligence, while remaining demonstrably safe and reliable at the same time. This is why Fraunhofer IKS researches both artificial intelligence and software engineering — we consider resilience and intelligence as part of the same process.

fortiss is the **Free State of Bavaria Research Institute for software-intensive systems** based in Munich. The institute's scientists work on research, development and transfer projects together with universities and technology companies in Bavaria and other parts of Germany, as well as across Europe. The research activities focus on state-of-the-art methods, techniques and tools used in software development and systems & service engineering and their application with cognitive cyber-physical systems such as the Internet of Things (IoT).
fortiss is legally structured as a non-profit limited liability company (GmbH). The shareholders are the Free State of Bavaria (majority shareholder) and the Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.

fortiss GmbH
Guerickestraße 25
80805 München
Deutschland
www.fortiss.org
Tel: +49 89 3603522 0
E-Mail: info@fortiss.org

fortiss