What, Indeed, is an Achievable Provable Guarantee for Learning-Enabled Safety-Critical Systems

Saddek Bensalem[1], Chih-Hong Cheng[2], Wei Huang[3], Xiaowei Huang[4],
Changshun Wu[1], and Xingyu Zhao[5]

[1] University Grenoble Alpes, VERIMAG, Grenoble, France
{saddek.bensalem,changshun.wu}@univ-grenoble-alpes.fr
[2] Technical University of Munich, Garching, Germany
chih-hong.cheng@tum.de
[3] Purple Mountain Laboratories, Nanjing, China
huangwei@pmlabs.com.cn
[4] Department of Computer Science, University of Liverpool, Liverpool, UK
xiaowei.huang@liverpool.ac.uk
[5] WMG, University of Warwick, Coventry, UK
xingyu.zhao@warwick.ac.uk

Abstract. Machine learning has made remarkable advancements, but confidently utilising learning-enabled components in safety-critical domains still poses challenges. Among the challenges, it is known that a rigorous, yet practical, way of achieving safety guarantees is one of the most prominent. In this paper, we first discuss the engineering and research challenges associated with the design and verification of such systems. Then, based on the observation that existing works cannot actually achieve provable guarantees, we promote a two-step verification method for the ultimate achievement of provable statistical guarantees.

Keywords: Safety-critical systems · learning-enabled components · statistical guarantees.

1 Introduction

From the studies of Leibniz [1] to the philosophical view, the human mind and brain have been perceived as an information processing system and thinking as a form of computing. Over three centuries ago, two dreams were mingled, the philosopher's and the engineer's: the philosopher's ideal to have a sound method to reason correctly, and the engineer's dream to have a machine to calculate efficiently and without error. Any attempt to assimilate the human brain into a mechanical or computer machine necessarily negates the autonomy of thought. The latter is not the result of chance or indeterminacy but instead of a possibility of choice according to the reasoning based on rules and principles. By its organization, the human brain allows the emergence of cognitive autonomy. Of course, suppose we accept the idea of a level of existence proper to the cognitive processes. In that case, the philosophical dream becomes, more modestly, that of understanding the diversity of human cognitive functions. The development of the general theory of automata and the formalization of the construction of complex machines by

Von Neumann allowed the pursuit of the engineer's dream. A central turning point took place around the 1960s, with the design of machines on the one hand and progress in artificial intelligence (AI) and cognitive science on the other hand. Significant successes have been achieved, for example, in natural language processing.

Our world today is witnessing the genesis of a significant shift in how advanced technologies operate. We are beginning to see increasingly independent and autonomous systems in this emerging wave of available automation. The degree of interactions between these systems and human operators is gradually being reduced and pushed further away. These autonomous systems are inherently sophisticated and operate in complex, unpredictable environments. Unfortunately, they still face deployment concerns in safety-critical applications (e.g., transportation, healthcare, etc.) due to a lack of trust, behavioural uncertainty, and technology compatibility with safe and secure system development methods. In particular, Urban Autonomous Driving and Intelligent Medical Devices are considered to be the most complex problem in autonomy; existing development of autonomous vehicles naturally includes the AI part (e.g., machine-learning for perception), as well as the CPS part (e.g., for vehicle control or decision making via infrastructure support). However, there are significant challenges in ensuring the quality of the overall system.

To ensure the safety of autonomous systems that incorporate AI components, we consider it mandatory for the overall engineering process to understand the safety performance of AI components while considering their impact on the overall system. Guaranteeing safety in critical systems that incorporate AI components, however, is not a straightforward process. Several constituent elements of safety cover all the dimensions of an AI system. The criteria catalog we can find in the literature to improve safety in critical applications can be summarized as the following:

- All algorithms based on decision-making shall be explainable [2, 3, 4, 5];
- The functionality of algorithms shall be analyzed and validated using formal verification methods before use [6, 7, 8, 9, 10];
- Statistical validation is necessary, mainly in cases where formal verification is unsuitable for specific application scenarios due to scalability issues [11, 12];
- The inherent uncertainty of neural network decisions shall also be quantified [13, 14, 15, 16];
- Systems must be observed during operation, for example, by using online monitoring processes [17, 18, 19, 20, 21].

In this paper, we promote an approach founded on a two-step integration. The first step involves a system-level analysis and testing, rather than solely focusing on the AI component in isolation. It recognizes the interconnected nature of the system and considers the integration and interactions of various components. By examining the system as a whole, potential risks and vulnerabilities can be identified, allowing for comprehensive safety assurance. The second step involves a detailed analysis of the AI components themselves, without considering their impact on the overall system. While this step provides insights into the specific AI algorithms and models, in itself it may overlook potential risks arising from the interactions between the components and the broader system context. This two-step integration of verification processes is to assess the safety performance of AI components while also considering their impact on the overall system. This entails examining not only their individual performance but also their interactions within the broader system context. In addition to formal analysis, we can also conduct studies on statistical guarantees and how these guarantees propagate throughout the system.

The rest of the paper is organized as follows. Sections 2 and 3 discuss the challenges of designing reliable and trustworthy AI critical systems from an engineering and research perspective, respectively. In Section 4, we present our methodology and proposed solutions to tackle these challenges. Finally, Section 5 provides a summary of the conclusions and highlights avenues for future work.

2 Challenges in Engineering Safety-critical Systems integrating Learning-enabled Components

The engineering of safety-critical systems has been a mature paradigm with the support of safety standards such as IEC 61508, ISO 26262, or DO-178c. The rigorous method implied by the process focuses on hazards caused by the malfunctioning behavior of E/E safety-related systems, including the interaction of these systems. Nevertheless, even in the absence of system malfunctioning, functional insufficiencies caused by performance limitations and incomplete/improper specification can also be the source of hazards, where standards such as ISO 21448 are introduced to address these issues.

To ensure the necessary level of safety and reliability, a learning-enabled component must also meet the identical functional safety standards encompassing reliability, applicability, maintenance, and safety (RAMS) as any other system. Moreover, it should mitigate the impacts of malfunctions to fulfill the essential safety and reliability prerequisites. On the other hand, properly ensuring the safety of the intended functionality (SOTIF) is the crucial gap in embracing the legitimate use of learning-enabled components. In the following, we enumerate some of the key limiting factors.

1. The introduction of learning-enabled components comes with the practical motivation where the operational environment is ***open*** and ***dynamic*** (e.g., urban autonomous driving), thereby inherently making rigorous analysis complicated [22, 23, 10].
2. ***Data*** has played a central role in learning-enabled systems [24, 25, 26]. Under the slogan “data is the new specification”, it is crucial to have a systematic approach to performing data collection, cleaning, labeling, as well as managing the data to incorporate adjustment of the operational domain.
3. Learning implies translating the implicit knowledge embedded in the data to a model. Despite the mathematical optimization nature of learning model parameters being transparent, the ***uncertainty*** [13, 14, 15, 16] caused by the model training or the data can lead to fundamental concerns about the validity of the prediction.
4. Classical techniques for software ***verification*** encounter scalability issues [10, 21]. Learning models such as deep neural networks create highly non-linear functions to perform classification and prediction tasks. Formal verification or bug finding thus can be viewed as a non-linear optimization problem across the high dimensional input space. The problem even worsens when the learned model controls a ***plant*** governed by highly nonlinear dynamics.

5. The derivation of ***safety specifications for the learning component*** can be far from trivial [10, 27]. While for tasks such as image-based object detection, the performance specification characterizing the error rate is relatively straightforward (which commonly leads to a probabilistic threshold on error rate), for control applications, the safety and performance requirement needs to be translated into reward signals in order to be used by (reinforcement) learning methods.
6. Finally, the above challenges are further complicated by the fact that the engineering of learning-enabled components is ***iterative*** with the goal of ***continuous improvement*** [10, 13, 14, 15, 16]. It is also complicated by the dimension of avoiding malfunctioning, implying the need to design hardware or software architectures to avoid transient or permanent faults in the learning-enabled components.

Unfortunately, the state-of-the-art guidelines or standards only provide high-level principles, while concrete methods for ***safe and cost-effective*** implementation are left for interpretation. This ultimately brings the research need in the field, which we detail in subsequent sections.

3 Research Challenges

The reason why one wants to apply machine learning to a safety critical application is two-fold: (1) it is impossible to program a certain functionality of the application and (2) a machine learning model can not only perform well on existing data but also generalise well to unseen data. Nevertheless, it is required that a machine learning model has to be safe and well performed such that both safety and performance can be quantified with error bounds given. Safety will be prioritised when a balance is needed.

Remark 1. While non-trivial, it is possible that a software or hardware system can be designed and implemented with ultra-high reliability, thanks to the availability of specification and requirements. However, this is unlikely for machine learning models, due to the unavailability of specifications and the complexity of the learning process. This calls for novel design and implementation methodologies for machine learning systems to satisfy both safety and performance requirements.

For the remaining of this section, we discuss challenges a novel methodology needs to tackle. While every gap between traditional software and AI-based systems, as discussed in Section 2, leads to research challenges, we believe the most significant ones are from (1) the environmental uncertainties that an AI-based system has to face, (2) the size and complexity of the AI models themselves, and (3) the lack of novel analysis methods that are both rigorous and efficient in dealing with the new problems. These three challenges lead to our proposal of considering ***statistical guarantees*** and ***symbolic analysis*** of AI models.

3.1 Uncertainty

In machine learning, uncertainty is often decomposed into aleatoric uncertainty and epistemic uncertainty, with the former irreducible and the latter reducible in theory. To explain this, we formalise the concept of generalisability, which requires that a neural network works well on all possible inputs in the data domain $\mathtt{X}$, although it is only trained on the training dataset (X, Y).

Definition 1. *Assume that there is a ground truth function* $f : \mathtt{X} \to \mathtt{Y}$ *and a probability function* $O_p : \mathtt{X} \to [0, 1]$ *representing the operational profile. A network* $\mathcal{N}$ *trained on* (X, Y) *has a generalisation error:*

$$G^{0-1}_{\mathcal{N}} = \sum_{x \in \mathtt{X}} \mathbf{1}_{\{\mathcal{N}(x) \neq f(x)\}} \times O_p(x) \tag{1}$$

where $\mathbf{1}_{\mathtt{S}}$ *is an indicator function – it is equal to 1 when* $\mathtt{S}$ *is true and 0 otherwise.*

We use the notation $O_p(x)$ to represent the probability of an input x being selected, which aligns with the *operational profile* notion [28] in software engineering. Moreover, we use 0-1 loss function (i.e., assigns value 0 to loss for a correct classification and 1 for an incorrect classification) so that, for a given O_p, $G^{0-1}_{\mathcal{N}}$ is equivalent to the reliability measure *pfd* (the expected probability of the system failing on a random demand) defined in the safety standard IEC-61508.

We decompose the generalisation error into three:

$$G^{0-1}_{\mathcal{N}} = \underbrace{G^{0-1}_{\mathcal{N}} - \inf_{\mathcal{N} \in \mathtt{N}} G^{0-1}_{\mathcal{N}}}_{\text{Estimation error of } \mathcal{N}} + \underbrace{\inf_{\mathcal{N} \in \mathtt{N}} G^{0-1}_{\mathcal{N}} - G^{0-1,*}_{f,(X,Y)}}_{\text{Approximation error of } \mathtt{N}} + \underbrace{G^{0-1,*}_{f,(X,Y)}}_{\text{Bayes error}} \tag{2}$$

a) The *Estimation error of* $\mathcal{N}$ measures how far the learned classifier $\mathcal{N}$ is from the best classifier in $\mathtt{N}$, the set of possible neural networks with the same architecture but different weights with $\mathcal{N}$. Lifecycle activities at the **model training** stage essentially aim to reduce this error, i.e., performing optimisations of the set $\mathtt{N}$.

b) The *Approximation error of* $\mathtt{N}$ measures how far the best classifier in $\mathtt{N}$ is from the overall optimal classifier, after isolating the Bayes error. The set $\mathtt{N}$ is determined by the architecture of DNNs (e.g., numbers of layers), thus lifecycle activities at the **model construction** stage are used to minimise this error.

c) The *Bayes error* is the lowest and irreducible error rate over all possible classifiers for the given classification problem [29]. The irreducibility refers to the training process, and the Bayes errors can be reduced in data collection and preparation. It is non-zero if the true labels are not deterministic (e.g., an image being labelled as y_1 by one person but as y_2 by others), thus intuitively it captures the uncertainties in the dataset (X, Y) and true distribution f when aiming to solve a real-world problem with machine learning. We estimate this error (implicitly) at the **initiation** and **data collection** stages in activities like: necessity consideration and dataset preparation etc.

Both the Approximation and Estimation errors are reducible, and are caused by the epistemic uncertainties. The Bayes error is irreducible, and caused by the aleatoric uncertainty. The *ultimate goal* of all lifecycle activities is to reduce the three errors to 0, especially for safety-critical applications.

Aleatoric uncertainty, as discussed in [16], may include various data-related issues such as survey error, missing data, and possible shifts in the data when deployed in real-world. Definition and measurement of these uncertainties can be done more reasonably with probabilistic/statistical distributions.

3.2 Size and Complexity of the AI Models

Another significant challenge is on the AI model itself. While it is believed that larger – often overparamterised – models can perform well [30], large models cannot be analysed analytically due to its size and complexity. Formal verification methods that check the local robustness of a neural network against perturbations are either limited on the number of neurons in a network (such as [31, 32, 33]) or limited by the number of input dimensions that can be perturbed (such as [34, 35, 36]). Even the theoretical analysis in machine learning field, which is usually less rigorous than in formal methods, has to be conducted on much simpler models such as linear and random projection models (see e.g., [37]). The situation is getting worse when we have to deal with large language models, see e.g., [38] for a discussion on their safety and trustworthiness issues.

3.3 Lack of Novel Analysis Methods that are both Rigorous and Efficient

Existing analysis methods from either formal methods or software testing are mostly aimed to extend their success in traditional software and systems. For example, constraint solving and abstract interpretation are popular methods in robustness verification of neural networks, and structural testing are popular testing methods for neural networks. Actually, the traditional verification methods have already been experiencing scalability problems when dealing with traditional software (up to several thousands lines of code), and it is therefore unlikely that they are able to scale and work with modern neural networks (which typically have multi-millions or billions of neurons). For testing methods, there is a methodological barrier to cross because neurons do not have clear semantics as variables, so does the layers with respect to the statements. Such mismatches render the test coverage metrics, which are designed by adapting the known test coverage in software testing such as statement coverage, potentially uncorrelated with the properties to be tested.

Another critical difference from traditional software that is posed on the analysis methods is the perfection of neural networks. For software to be applied to safety critical applications, a “possible perfection” notion [39, 40, 41] is used. However, for machine learning, the failures are too easy to find, and it does not seem likely that a perfect, or possibly perfect, machine learning model exists for a real-world application. To this end, a novel design method is needed to ensure that an AI-based system can potentially be free from serious failures.

Moreover, multiple properties may be required for a machine learning model, e.g., robustness, generalisation, privacy, fairness, free from backdoor attacks, etc. However, these properties can be conflicting (e.g., robustness-accuracy trade-off) and many of them without formal specifications, which lead to the challenge of lacking effective methods for the analysis and improvement of them altogether for a machine learning model.

4 Methodology

The needs of safety-critical systems require that, even facing challenges that are more significant than traditional software, a legitimate methodology will still provide rigorous and provable guarantees, concerning the satisfiability of properties on the autonomous cyber-physical system under investigation. We conceptualise AI-based systems into five levels (shown in Fig. 1). For the remainder of this section, we discuss the methodology needed at each level and across levels. Specifically, at each level, we consider the following questions: For sources of uncertainty identified in earlier sections, what metrics (e.g., binary, worst-case or probabilistic) shall we use to measure them? How to efficiently evaluate those metrics? Can we provide any forms of guarantees on the evaluations? Moreover, we raise questions that span across different levels: How do metrics at higher levels break down to metrics at lower levels? If and how the guarantees (in various forms) from lower levels can propagate and compound to higher levels, ultimately aiming to make meaningful claims about the entire system.

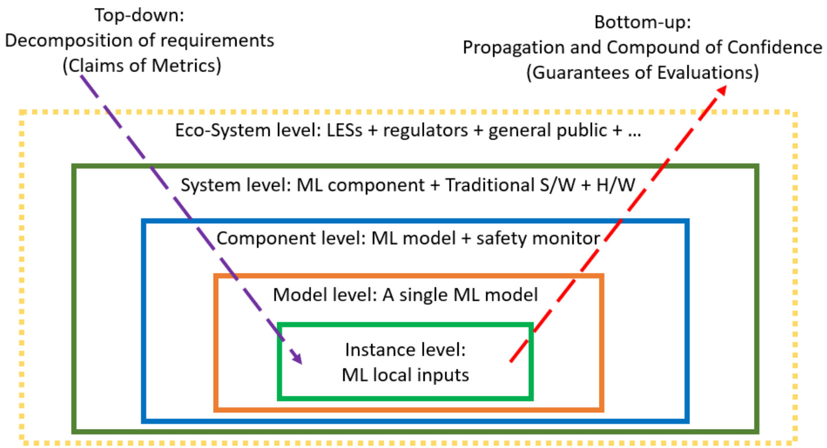
Top-down:
Decomposition of requirements
(Claims of Metrics)
Bottom-up:
Propagation and Compound of Confidence
(Guarantees of Evaluations)
Eco-System level: LESs + regulators + general public + ...
System level: ML component + Traditional S/W + H/W
Component level: ML model + safety monitor
Model level: A single ML model
Instance level:
ML local inputs

Fig. 1: Research challenges organised into five conceptual levels with top-down and bottom-up routes.

Our proposed methodology consists of the following attributes:

- a set of specification languages that describe, and connect, requirements of different levels;
- a formal method of propagating statistical guarantees about the satisfiability of the requirements across the system and the component levels;
- a rigorous method of achieving required statistical guarantees at the instance and the model levels;

This is founded on two threads of state-of-the-art research: a design and co-simulation framework (Section 4.1) and some design-time verification and validation (V&V) methods for machine learning models (Section 4.4). The co-simulation framework is to effectively simulate the real-world system and environment to a sufficient level of fidelity. The V&V methods are to detect vulnerabilities and improve the machine learning model's safety and performance. It is noted that, the V&V methods can improve the system but may not be able to provide provable safety guarantee, for reasons that we will discuss below. Once the improvement is converged (or certain termination condition is satisfied), the new methodology is applied for the ultimate achievement of provable guarantees.

4.1 State-of-the-Art 1: A Design and Co-Simulation Framework

This section summarizes the current research on the rigorous design of AI-enabled systems out of the EU Horizon 2020 project FOCETA.

Design of trustable AI models Design of trust-able AI models requires considering the complete engineering life-cycle beyond optimizing the model parameters. Figure 2 presents a flow design for the AI model development. We consider the lifecycle phases: data preparation, training, offline verification and validation, and online deployment. During the offline V&V, techniques for the falsification and explanation are applied to discover whether there are failures regarding the decision-making (i.e., falsification) or failures demonstrating the inconsistency with human's perception (i.e., explanation). In addition to their individual functionalities, falsification and explanation may benefit from mutual interactions, to make sure that a decision failure can be explained and two inconsistent explanations are tested, see e.g., [42]. A formal verification process is called only when no error can be found from both falsification and explanation.

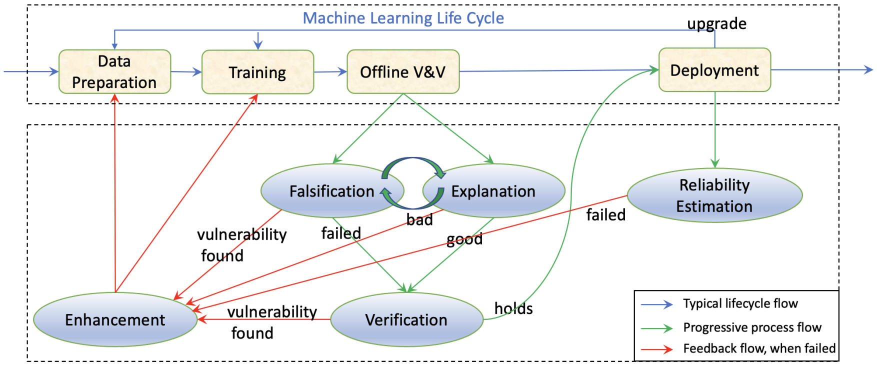

Fig. 2: A Verification and Validation Framework for Machine Learning Enhancement

In the context of a real-world learning-enabled system, the offline V&V can be insufficient, due to the scalability of the verification techniques and the environmental uncertainties that are unknown during offline development (details will be provided below). In such cases, a reliability estimation to analyse the recorded runtime data will be needed, to understand statistically whether the AI-based system can run without failures e.g., in the next hour, with high probability.

Another important module in Figure 2 is the enhancement, where the failure cases are considered for the improvement of the machine learning models, through either data synthesis or model training.

Design flow for safety-critical systems with AI components Like any other safety-critical systems, the design flow for AI-enabled systems shall also cover design and operation time activities. However, in contrast to classical critical systems where the environment is largely static and predictable, the use of AI-enabled systems reflects the need to handle an open environment.

Within FOCETA, we view the engineering of the complete system as analogous to the engineering of the AI component, where it is important to create a continuous loop of improvement between development and operation. The current state of the practice is extended toward transferring knowledge about systems and their contexts (e.g., traffic situations) from the development to operations and from the operations back to action in iterative steps of continuous improvements. The methodology enables their ongoing engineering over the complete life cycle of autonomous learning-enabled systems – from specification, design, implementation, and verification to operation in the real world with a particular focus on correctness concerning evolving requirements and the systems' safety. Moreover, the whole design flow ensures traceability between requirements and the system/component design.

A key feature is the usage of runtime monitors for the seamless integration of development and operations. In contrast to AI component monitors that largely detect situations such as out-of-distribution, system-level runtime monitors observe a system (part) via defined interfaces and evaluate predefined conditions and invariants about the system behavior based on data from these interfaces. This allows us to identify the need for AI model updates during continuous testing/verification if, for example, some data in a test scenario results in a safety property violation or if a new requirement emerges in response to a previously unknown adversarial threat.

Simulation-based Modeling and Testing at Design Time Using simulation in the design phase offers multiple advantages. It provides a cost-effective means to verify the system's performance over diverse parameter ranges and generate massive scenarios. It follows that critical methods can be already identified in the virtual environment, and only those can be replayed in the much more expensive physical setting. Simulation-based testing allows the generation of scenarios (e.g., with infrequent events) that may be impossible to realize when the system operates in its environment (e.g., with specific weather conditions, such as fog or snow). It also enables the creation of safety-critical situations (e.g., crashes) without compromising the safety of the real-life actors. The simulation framework shall allow the integration of heterogeneous components that may be designed with different tools and frameworks. In addition, there is a need to rigorously argue that the domain gap between synthetic data produced by the simulation engine and real data observed in the field is closed. In layman's words, an image being "photo-realistic" does not necessarily imply its being "real".

Deployment, Operation, and Analysis of AI critical systems at Runtime The analysis of the AI components and their integration into the AI critical system during design time, together with protective mechanisms synthesized around AI models, help the safety assurance of the overall system during real-time operation. These measures are complemented by runtime verification, which plays a central role during the AI critical operation. Runtime monitors allow us to observe the system and its interaction with the environment and gather helpful information regarding (1) violation of safety or other requirements and (2) new operational scenarios that were not captured by the training data and models, and (3) other unexpected situations and anomalies not characterized by the existing requirements set. To be effective, monitors must be present both at the component level (AI and classical) and at the system level; the information gathered by different monitors must be fused to derive useful information that can be used to i) ignore the situation (e.g., detected object misclassification) that does not impact the system-level control decision, ii) take a protective measure (e.g., switch from the advanced to a base controller) or iii) improve the design (e.g., provide a new scenario for the training data). The last point refers to an evolvable AI critical system, in which information from the system operation is collected and used to go back to the design and enhance its functionality based on new insights, thus effectively closing a loop between the design and the operation phase.

4.2 Properties and Specifications at Different Levels

On the system level, we may use temporal logic to express the required dynamic behavior. There are recent attempts to extend the temporal logic for AI-based systems. For example, [43] formalises requirements of an autonomous unmanned aircraft system based on an extension of propositional LTL, where temporal operators are augmented with timing constraints. It uses atomic propositions such as "$horizontalIntruderDistance > 250$" to express the result of the perception module, without considering the sensory input and the possible failure of getting the exact value for $horizontalIntruderDistance$. [44] introduces a specification language based on LTL, which utilises event-based abstraction to hide the details of the neural network structure and parameters. It considers the potential failure of the perception component and uses a predicate $pedestrian(\mathbf{x})$ to express if $\mathbf{x}$ is a pedestrian (referring to the ground truth). However, it does not consider the predicate's potential vulnerabilities, such as robustness, uncertainty, and backdoors. [45] proposes Timed Quality Temporal Logic (TQTL) to express monitorable [46] spatio-temporal quality properties of perception systems based on neural networks. It considers object detectors such as YOLO and uses expressions such as $D_0 : d_1 : (ID, 1), (class, car), (pr, 0.9), (bb, B1)$ to denote an object d_1 in a frame D_0 such that it has an index 1, a predictive label car, the prediction probability 0.9, and is in a bounding box B_1. Therefore, every state may include multiple such expressions, and then a TQTL formula can be written by referring to the components of the expressions in a state.

For our purpose of having a statistical guarantee for properties at the system level (see Figure 1), for any temporal logic formula φ, a statistical guarantee is needed, e.g., in the form of

$$P(err(\varphi) \leq \epsilon) > 1 - \delta \tag{3}$$

where φ is a formula such that $err(\varphi)$ denotes that estimation error on the satisfiability of φ on the system, and ϵ and δ are small positive constants. In the formula φ, we need atomic propositions that are related to the perception components. According to different assumptions, we may have different atomic propositions: instance-level atomic propositions or model-level atomic propositions. For an instance-level atomic proposition such as $pedestrian_{\epsilon,\delta}$, it expresses that the error of having a $pedestrian$ in the current input is lower than ϵ, under the confidence level no less than δ. In such cases, the statistical guarantee is established by considering the local robustness (i.e., cell unastuteness as in [47]). On the other hand, for a model-level atomic proposition such as $perception_{\epsilon,\delta}$, it expresses that the error of having a failed detection among all possible next inputs is lower than ϵ, under the confidence level no less than δ. In such cases, the statistical guarantee is established by considering the reliability as in [47].

The selection between instance-level and model-level atomic propositions depends on the assumptions. If we believe that a failure on the perception component does not have a correlation with the failures of other components, a model-level atomic proposition will be sufficient. On the other hand, if a correlation between failures is expected, and we want the verification to fully consider such correlations, an instance-level atomic proposition will be more appropriate and accurate.

Section 4.3 will discuss how to achieve the statistical guarantee (i.e., ϵ and δ). For the model-level atomic propositions, the specification language in [48] considers not only the functionality (i.e., the relation between input and output) of a trained model but also the training process (where objects such as training datasets, model parameters, and distance between posterior distributions are considered). With this, it can express the safety and security properties that describe the attacks during the lifecycle stages.

4.3 Guarantees Achieved at Component Levels

This section will discuss a potential solution that can be utilised to achieve the statistical guarantee (i.e., ϵ and δ) for an atomic proposition describing certain safety properties as summarised in [48]. As discussed in Section 4.4, this cannot be achieved by a standalone machine learning model, even if a V&V framework as in Figure 2 is applied, due to the insufficiency of machine learning models. We suggest a monitored machine learning system, i.e., a machine learning model running in parallel with a runtime monitor. As indicated in Figure 3, for non-ML safety-critical systems with clear specifications about safety, a runtime monitor often acts like an alarm to alert the unsafe behaviour. On the other hand, for ML systems without safety specifications but only with samples, a runtime monitor needs to analyze the samples and predict the safety of the current input e.g., in the manner of a traffic light system as briefly discussed below. While the predictions might not be completely correct, we expect they are conservative with a provable guarantee and as accurate as possible.

A runtime monitor checks every input of a neural network and issues warnings whenever there is a risk that the neural network might make wrong decision. As discussed earlier, given the availability of many adversarial attacks, it is unlikely that a neural network itself can achieve “possible perfection” [49] – a notion of traditional safety-critical software introduced by [39, 40, 41]. The safety of a neural network, however, can potentially be achievable with the support of a runtime monitor. Actually, in the extreme case, if a runtime monitor is so restrictive that none of the input instances can pass without warning, the neural network under the runtime monitor is safe (although the performance is bad).

The design of a runtime monitor for a given neural network is to ensure that the safety of the monitored neural network can be achieved with guarantees. While the restrictive

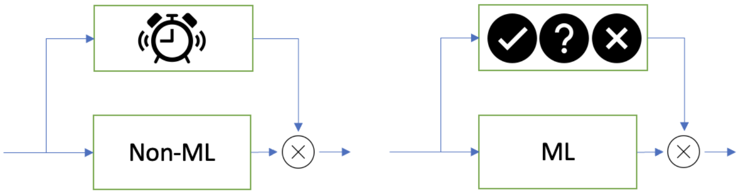

Fig. 3: Runtime Monitors with (for non-ML systems) and without (for ML systems) specifications

runtime monitor mentioned above suggests absolute safety, it is also undesirable due to its performance. In the following, we discuss a possible runtime monitor that is able to achieve statistical guarantee of the form (3). The core idea of this monitor is to represent the abstracted experiences symbolically, serving as references for future behaviors. The process involves recording observed data or their learned high-level features for each decision made by the neural network. These data points are then clustered based on their similarities, and each cluster is approximated by a box as an abstraction. Every box $\mathbf{b}$ can be described as a tuple (l, r, c, m, y, i), where l is the location of the box, r is the radius vector of the box, c is the cluster that the box belongs to, m is the number of data samples that the box contains, and y is the predictive class label of the box, and i is the correctness indicator of prediction relating to the abstracted samples. Once these abstractions are derived, they can be effectively symbolized, and operational symbols can be defined to establish the runtime monitor. When new data points and decisions arise, we compare the network's behavior for a given input with the reference abstractions. Generally, there exist two types of boxes: *positive* and *negative* ones, representing abstracted good and bad behaviors, respectively. If the behavior is similar to good behaviors (inside a positive box), the decision is accepted; if it resembles bad behaviors (inside a bad box), the decision is rejected.

Our guarantees are on two levels. The first level considers the confidence the runtime monitor classifies an input as safe or not. Assume that we have a data point that falls within a box $\mathbf{b}$ that is either positive or negative with respect to a label. Since the box will classify the data point, we can utilise the information in the box (e.g., the known points that fall within the box) to conduct a Probably Approximately Correct (PAC) analysis, or utilising Hoeffding inequality such as in [11], to determine a probability and an error. That is, for each box $\mathbf{b}$, we may have

$$P_{\mathbf{b},\mathcal{D}}(err_{\mathbf{b},\mathcal{D}}(c,h) \leq \epsilon_{\mathbf{b}}) > 1 - \delta_{\mathbf{b}} \tag{4}$$

for small positive numbers $0 < \epsilon_{\mathbf{b}}, \delta_{\mathbf{b}} < 1/2$, where $\mathcal{D}$ is the data distribution, $err_{\mathbf{b},\mathcal{D}}(c,h)$ is the probabilistic error of the hypothesis h (i.e., the probability of h does not hold) in the box $\mathbf{b}$, with respect to the concept c and the distribution $\mathcal{D}$, such that $err_{\mathbf{b},\mathcal{D}}(c,h) = P_{x\in\mathcal{D},x\in\mathbf{b}}(h(x) \neq c(x))$, i.e., the probability over x drawn from $\mathcal{D}$ and $\mathbf{b}$ that $h(x)$ and $c(x)$ differ.

The second level considers the probability of a future input that our runtime monitor can confidently classify. Assume that we have a set of n boxes in the space (e.g., the hidden space before the Softmax layer) such that they are either positive or negative (we do not consider uncertain boxes for guarantees) with respect to certain label. We can use hypothesis testing to determine the probability (the error) of a future point falling within these boxes, according to the set of known data points. Similarly, we will have

$$P_{\mathcal{D}}(err_{M^*,\mathcal{D}}(M) \leq \epsilon_{M^*}) > 1 - \delta_{M^*} \tag{5}$$

for small positive numbers $0 < \epsilon_{M^*}, \delta_{M^*} < 1/2$, where $err_{M^*,\mathcal{D}}(M) = P_{x\in\mathcal{D}}(M(x) \neq M^*(x))$ is the probability that the runtime monitor M disagrees with the ground truth M^* regarding whether an input $\mathbf{x}$ is within the confirmed boxes. Moreover, we can replace the hypothesis testing with more effective, and more scalable, probability methods such as MCMC or that we did in [49].

A "combination" (to be analytically derived as future work) of the above levels will reach a statistical way of conducting reliability estimation over runtime data. A statistical guarantee of the form

$$P_{\mathcal{D}}(err_{\mathcal{D}}(c,h) \leq \epsilon) > 1 - \delta \tag{6}$$

will be achieved, where $\mathcal{D}$ is the operational distribution of the AI component within the system, $err_{\mathcal{D}}(c,h)$ denotes the error probability of the AI model c with respect to the ground truth h, and both ϵ and δ are small positive numbers.

Remark 2. The chance constraint (as in Equation (6)) as a statistical guarantee for safety is not as strong as a deterministic guarantee, which states the absolute missing of failures, or a probabilistic guarantee, which states the missing of failures with certain probability. However, the deterministic guarantee is infeasible in practice due to the environmental uncertainties, as we have discussed for offline verification and validation. Practical methods are missing on how to achieve tight probabilistic guarantees.

4.4 State-Of-The-Art 2: Offline V&V Methods and Guarantee

This section discusses the existing verification and validation methods, and explains why they cannot provide the guarantees that are needed for AI-based systems. AI models, especially Deep Neural Networks, are known to be susceptible to the adversarial attack and backdoor attack. Given a DNN model f, which maps a high dimensional input x to a prediction class y, adversarial attack and backdoor attack add maliciously generated perturbations ϵ into the benign inputs, leading to the mis-predictions of DNNs (refer to the survey for the difference between adversarial attack and backdoor attack).

$$f(x) = y \ \& \ f(x+\epsilon) \neq y \tag{7}$$

This section will briefly review the existing V&V methods on the robustness of DNNs against the adversarial perturbation ϵ and discuss the guarantee to the safety of AI model.

Verification Verification techniques are to determine whether or not a property of a neural network holds within a given range of inputs. The existing verification techniques can be categorized according to the guarantee they provide. *Deterministic guarantees* are achieved by transforming verification of deep neural networks into a set of constraints so that they can be solved with a constraint solver, such as Satisfiability Modulo Theories (SMT) solver [50, 51], Boolean Satisfiability Problem (SAT) solver [52, 53, 54], and mixed integer linear programming (MILP) solver [55, 56]. The name "deterministic" comes from the fact that these solvers often return a deterministic answer to a query, i.e., either satisfiable or unsatisfiable. Some verification techniques can offer *one-sided guarantee*, i.e. deep neural network is robust when adversarial perturbation measured by L_p norm is bounded less than ϵ. These approaches leverage the abstract interpretation [57, 58], convex optimization [59, 60], or interval arithmetic [61, 62] to compute the approximate bound. Compared to the verification techniques with deterministic guarantee, the bounded estimation can work with larger models, up to 10000 hidden neurons, and can avoid floating point issues in existing constraint solver implementations [63]. To deal with real-world system, which contains the state of the art DNNs with at least multi-million hidden neurons, some practical verification techniques are developed to offer *converging bounds guarantee* and *statistical guarantee*. Layer-by-layer refinement [6], reduction to a two-player turn-based game [8], and global optimization-based approaches [64] are developed to compute the lower bounds of robustness by utilizing the Lipschitz constant and the bounds converge to the optimal value. The *statistical guarantee* is achieved by utilizing the statistical sampling methods, e.g. Monte Carlo based sampling, to estimate the robustness with a certain probability. CLEVER [65] estimates the robustness lower bound by sampling the norm of gradients and fitting a limit distribution using extreme value theory. [66] utilizes the multi-level splitting sampling to calculate the probability of adversarial examples in the local region as an estimation of local robustness. The local probabilistic robustness estimation can be aggregated over the train set to form the global robustness estimation [67]. [47, 68] further propose the concept of reliability, which is a combination of robustness and generalization, and estimated on the operational dataset to provide statistical guarantee on neural networks' overall performance.

Testing When working with large-scale models, often used in the industry, verification is not a good option. Verification techniques offer guarantees to the results at the expense of high computational cost. The cost goes sharply with the increase of model's complexity. Testing arises as a complement to verification. Instead of pursuing mathematics proofs, testing techniques exploit the model in a broad way to find potential faults. The first category of works is the coverage-guided testing. A large amount of coverage metrics are designed in consideration of the structure information of DNNs. Structure coverage metrics, such as neuron coverage [69], k-multisection neuron coverage [70], neuron activation pattern coverage [70], Modified Condition/Decision Coverage (MC/DC) for neuron layers [71] are proposed in the past few years. There are also a few works dedicated to designing coverage metrics for Recurrent Neural Networks (RNNs), such as modeling RNNs as abstract state transition systems and covering different states and transitions [72], and quantifying one-step hidden memory change and multi-step temporal relation [73]. They are all based on the assumption that the activation of neurons represents the functionality of DNNs. By achieving a higher coverage rate in proposed structure coverage metrics, the functionality of DNNs are more thoroughly exercised. Therefore, structure coverage metrics can guide the generation of test cases as diversified as possible, and detect different types of defects, such as adversarial examples and backdoor input [73]. However, the weak correlation between structure coverage metrics and the defects can not guarantee that increasing the coverage rate can find more faults in DNNs.

The second category of works is distribution-aware testing. There has been a growing body of research focusing on the development of distribution-aware testing techniques for DNNs. To approximate the distribution of training data, deep generative models such as Variational AutoEncoders (VAE) and Generative Adversarial Networks (GAN) are commonly used, especially for high-dimensional inputs like images. Berend et al. [74] propose the first distribution-guided coverage criterion, which integrates out-of-distribution (OOD) techniques to generate unseen test cases and provides a high level of assurance regarding the validity of identified faults in DNNs. In a study by Dola et al. [75], the validity of test cases generated by existing DNN test generation techniques is examined using VAE. By comparing the probability density estimates of a trained VAE model on data from the training distribution and OOD inputs, critical insights are obtained for validating test inputs generated by DNN test generation approaches. To generate realistic test cases that conform to requirements and reveal errors, Byun et al. [76] employ a variant of Conditional Variational Autoencoder (CVAE) to capture a manifold that represents the feature distribution of the training data. Toledo et al. [77] introduces the first method called distribution-based falsification and verification (DFV), which utilizes environmental models to concentrate the falsification and verification of DNNs on meaningful regions of the input space. This method is designed to leverage the underlying distribution of data during the process of DNN falsification and verification. Huang et al. [78] propose a hierarchical distribution-aware testing framework for DNNs. Their framework takes into account two levels of distribution: the feature level distribution, captured by generative models, and the pixel level distribution, which is represented by perceptual quality metrics. Although distribution aware testing can detect more meaningful faults for DNNs, which significantly contribute to the downstream repairing of DNNs, they still cannot provide the deterministic guarantee to the safety of DNNs.

5 Conclusion

Developing critical systems has always been challenging due to the potential harm caused by malfunctions, functional insufficiencies, or malicious attacks. The complexity is amplified when incorporating learning-enabled components, as the approaches taken by safety engineers who build the system often differ from those employed by AI/ML engineers who construct the components. Educating the general audience about AI safety concerns is essential for fostering active engagement in the ongoing discourse. However, to address the underlying engineering challenges, an interdisciplinary curriculum that bridges concepts from various fields such as AI/ML engineering and safety engineering can provide valuable insights and understanding.

We notice that there are two views on system safety in the broader community, the “binary” view and the “probabilistic” view, which present differing perspectives on how to approach safety assurance. Proponents of the binary view argue that safety is about clearly defining the system’s capabilities and limitations, establishing a definitive “safety boundary”. According to this view, we can confidently operate the system once we have a comprehensive understanding of this boundary. However, this viewpoint may hold primarily for traditional systems without AI components, where the system behavior is relatively simple and predictable.

The concept in this paper hints that we advocate the probabilistic view that safety for complex AI-enabled systems should be measured in terms of empirical probabilities[6], as modern systems are becoming increasingly complex, with inherent uncertainties that make it difficult to determine the system's safety boundary precisely. In this perspective, the boundary itself may even appear blurred due to the non-deterministic behaviors exhibited by AI algorithms. Consequently, adherents of the probabilistic view assert that safety assurance should consider the likelihood of various outcomes and incorporate risk assessment and mitigation strategies to manage uncertainties effectively.

References

1. M. Kulstad, L. Carlin, Leibniz's philosophy of mind (1997).
2. D. Gunning, M. Stefik, J. Choi, T. Miller, S. Stumpf, G.-Z. Yang, Xai—explainable artificial intelligence, Science robotics 4 (37) (2019) eaay7120.
3. S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, K.-R. Müller, Unmasking clever hans predictors and assessing what machines really learn, Nature communications 10 (1) (2019) 1096.
4. R. Confalonieri, L. Coba, B. Wagner, T. R. Besold, A historical perspective of explainable artificial intelligence, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 11 (1) (2021) e1391.
5. F. K. Došilović, M. Brčić, N. Hlupić, Explainable artificial intelligence: A survey, in: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO), IEEE, 2018, pp. 0210–0215.
6. X. Huang, M. Kwiatkowska, S. Wang, M. Wu, Safety verification of deep neural networks, in: Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30, Springer, 2017, pp. 3–29.
7. T. Dreossi, D. J. Fremont, S. Ghosh, E. Kim, H. Ravanbakhsh, M. Vazquez-Chanlatte, S. A. Seshia, Verifai: A toolkit for the formal design and analysis of artificial intelligence-based systems, in: International Conference on Computer Aided Verification, Springer, 2019, pp. 432–442.
8. M. Wu, M. Wicker, W. Ruan, X. Huang, M. Kwiatkowska, A game-based approximate verification of deep neural networks with provable guarantees, Theoretical Computer Science 807 (2020) 298–329.
9. C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, M. J. Kochenderfer, et al., Algorithms for verifying deep neural networks, Foundations and Trends® in Optimization 4 (3-4) (2021) 244–404.
10. S. A. Seshia, D. Sadigh, S. S. Sastry, Toward verified artificial intelligence, Communications of the ACM 65 (7) (2022) 46–55.
11. C. Huang, Z. Hu, X. Huang, K. Pei, Statistical certification of acceptable robustness for neural networks, in: Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 14–17, 2021, Proceedings, Part I 30, Springer, 2021, pp. 79–90.

[6] In statistics, empirical probability refers to the probability of an event based on observed data or evidence. The empirical probability is also known as experimental probability because it is derived from actual experimentation or observation.

12. T. Zhang, W. Ruan, J. E. Fieldsend, Proa: A probabilistic robustness assessment against functional perturbations, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2022, pp. 154–170.
13. S. Shafaei, S. Kugele, M. H. Osman, A. Knoll, Uncertainty in machine learning: A safety perspective on autonomous driving, in: Computer Safety, Reliability, and Security: SAFECOMP 2018 Workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Västerås, Sweden, September 18, 2018, Proceedings 37, Springer, 2018, pp. 458–464.
14. J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, et al., A survey of uncertainty in deep neural networks, arXiv preprint arXiv:2107.03342 (2021).
15. E. Hüllermeier, W. Waegeman, Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods, Machine Learning 110 (2021) 457–506.
16. C. Gruber, P. O. Schenk, M. Schierholz, F. Kreuter, G. Kauermann, Sources of uncertainty in machine learning – a statisticians' view (2023). `arXiv:2305.16703`.
17. C.-H. Cheng, G. Nührenberg, H. Yasuoka, Runtime monitoring neuron activation patterns, in: 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2019, pp. 300–303.
18. T. A. Henzinger, A. Lukina, C. Schilling, Outside the box: Abstraction-based monitoring of neural networks, in: ECAI 2020, IOS Press, 2020, pp. 2433–2440.
19. C.-H. Cheng, Provably-robust runtime monitoring of neuron activation patterns, in: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2021, pp. 1310–1313.
20. A. Lukina, C. Schilling, T. A. Henzinger, Into the unknown: Active monitoring of neural networks, in: International Conference on Runtime Verification, Springer, 2021, pp. 42–61.
21. C.-H. Cheng, C. Wu, E. Seferis, S. Bensalem, Prioritizing corners in ood detectors via symbolic string manipulation, in: International Symposium on Automated Technology for Verification and Analysis, Springer, 2022, pp. 397–413.
22. D. J. Fremont, T. Dreossi, S. Ghosh, X. Yue, A. L. Sangiovanni-Vincentelli, S. A. Seshia, Scenic: a language for scenario specification and scene generation, in: Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2019, pp. 63–78.
23. S. Zhong, K. Zhang, M. Bagheri, J. G. Burken, A. Gu, B. Li, X. Ma, B. L. Marrone, Z. J. Ren, J. Schrier, et al., Machine learning: new ideas and tools in environmental science and engineering, Environmental Science & Technology 55 (19) (2021) 12741–12754.
24. S. L. Brunton, J. N. Kutz, Data-driven science and engineering: Machine learning, dynamical systems, and control, Cambridge University Press, 2019.
25. C. V. G. Zelaya, Towards explaining the effects of data preprocessing on machine learning, in: 2019 IEEE 35th international conference on data engineering (ICDE), IEEE, 2019, pp. 2086–2090.
26. Y. Roh, G. Heo, S. E. Whang, A survey on data collection for machine learning: a big data-ai integration perspective, IEEE Transactions on Knowledge and Data Engineering 33 (4) (2019) 1328–1347.
27. S. Bensalem, C.-H. Cheng, X. Huang, P. Katsaros, A. Molin, D. Nickovic, D. Peled, Formal specification for learning-enabled autonomous systems, in: FoMLAS2022, 2022.
28. J. D. Musa, Operational profiles in software-reliability engineering, IEEE Software 10 (2) (1993) 14–32.
29. K. Fukunaga, Introduction to statistical pattern recognition, Elsevier, 2013.
30. P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, I. Sutskever, Deep double descent: Where bigger models and more data hurt, in: International Conference on Learning Representations, 2020.

31. J. Li, J. Liu, P. Yang, L. Chen, X. Huang, L. Zhang, Analyzing deep neural networks with symbolic propagation: Towards higher precision and faster verification, in: SAS2019, Springer, 2019, pp. 296–319.
32. R. Li, J. Li, C.-C. Huang, P. Yang, X. Huang, L. Zhang, B. Xue, H. Hermanns, Prodeep: A platform for robustness verification of deep neural networks, in: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2020, ACM, New York, NY, USA, 2020, pp. 1630–1634.
33. P. Yang, J. Li, J. Liu, C.-C. Huang, R. Li, L. Chen, X. Huang, L. Zhang, Enhancing robustness verification for deep neural networks via symbolic propagation, Form. Asp. Comput. 33 (3) (2021) 407–435.
34. W. Ruan, X. Huang, M. Kwiatkowska, Reachability Analysis of Deep Neural Networks with Provable Guarantees, in: Proc. of the 27th Int. Joint Conf. on Artificial Intelligence, IJCAI-18, 2018, pp. 2651–2659.
35. W. Ruan, M. Wu, Y. Sun, X. Huang, D. Kroening, M. Kwiatkowska, Global Robustness Evaluation of Deep Neural Networks with Provable Guarantees for the Hamming Distance, in: Proc. of the 28th Int. Joint Conf. on Artificial Intelligence, IJCAI-19, 2019, pp. 5944–5952.
36. P. Xu, W. Ruan, X. Huang, Quantifying safety risks of deep neural networks, Complex & Intelligent Systems (2022).
37. M. Belkin, D. Hsu, S. Ma, S. Mandal, Reconciling modern machine-learning practice and the classical bias–variance trade-off, Proceedings of the National Academy of Sciences 116 (32) (2019) 15849–15854.
38. X. Huang, W. Ruan, W. Huang, G. Jin, Y. Dong, C. Wu, S. Bensalem, R. Mu, Y. Qi, X. Zhao, K. Cai, Y. Zhang, S. Wu, P. Xu, D. Wu, A. Freitas, M. A. Mustafa, A survey of safety and trustworthiness of large language models through the lens of verification and validation (2023). `arXiv:2305.11391`.
39. B. Littlewood, J. Rushby, Reasoning about the reliability of diverse two-channel systems in which one channel is “possibly perfect”, IEEE Transactions on Software Engineering 38 (5) (2012) 1178–1194.
40. J. Rushby, Software verification and system assurance, in: 7th Int. Conf. on Software Engineering and Formal Methods, IEEE, Hanoi, Vietnam, 2009, pp. 3–10.
41. X. Zhao, B. Littlewood, A. Povyakalo, L. Strigini, D. Wright, Modeling the probability of failure on demand (pfd) of a 1-out-of-2 system in which one channel is “quasi-perfect”, Reliability Engineering & System Safety 158 (2017) 230–245.
42. W. Huang, X. Zhao, G. Jin, X. Huang, Safari: Versatile and efficient evaluations for robustness of interpretability, in: Int. Conf. on Computer Vision (ICCV’23), 2023.
43. A. Dutle, C. A. Muñoz, E. Conrad, A. Goodloe, L. Titolo, I. Perez, S. Balachandran, D. Giannakopoulou, A. Mavridou, T. Pressburger, From requirements to autonomous flight: An overview of the monitoring ICAROUS project, in: Proc. 2nd Workshop on Formal Methods for Autonomous Systems, Vol. 329 of EPTCS, 2020, pp. 23–30.
44. S. Bensalem, C.-H. Cheng, X. Huang, P. Katsaros, A. Molin, D. Nickovic, D. Peled, Formal specification for learning-enabled autonomous systems, in: FoMLAS2022, 2022.
45. A. Balakrishnan, A. G. Puranic, X. Qin, A. Dokhanchi, J. V. Deshmukh, H. Ben Amor, G. Fainekos, Specifying and evaluating quality metrics for vision-based perception systems, in: Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019, pp. 1433–1438. `doi:10.23919/DATE.2019.8715114`.
46. A. Balakrishnan, J. Deshmukh, B. Hoxha, T. Yamaguchi, G. Fainekos, Percemon: Online monitoring for perception systems, in: Runtime Verification, Springer, Cham, 2021, pp. 297–308.

47. Y. Dong, W. Huang, V. Bharti, V. Cox, A. Banks, S. Wang, X. Zhao, S. Schewe, X. Huang, Reliability assessment and safety arguments for machine learning components in system assurance, ACM Transactions on Embedded Computing Systems 22 (3) (2023) 1–48.
48. X. Huang, W. Ruan, Q. Tang, X. Zhao, Bridging formal methods and machine learning with global optimisation, in: A. Riesco, M. Zhang (Eds.), Formal Methods and Software Engineering, Springer International Publishing, Cham, 2022, pp. 1–19.
49. X. Zhao, A. Banks, J. Sharp, V. Robu, D. Flynn, M. Fisher, X. Huang, A safety framework for critical systems utilising deep neural networks, in: Computer Safety, Reliability, and Security: 39th International Conference, SAFECOMP 2020, Lisbon, Portugal, September 16–18, 2020, Proceedings 39, Springer, 2020, pp. 244–259.
50. G. Katz, C. Barrett, D. L. Dill, K. Julian, M. J. Kochenderfer, Reluplex: An efficient SMT solver for verifying deep neural networks, in: International Conference on Computer Aided Verification, Springer, 2017, pp. 97–117.
51. R. Ehlers, Formal verification of piece-wise linear feed-forward neural networks, in: Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15, Springer, 2017, pp. 269–286.
52. N. Narodytska, Formal analysis of deep binarized neural networks., in: IJCAI, 2018, pp. 5692–5696.
53. N. Narodytska, S. Kasiviswanathan, L. Ryzhyk, M. Sagiv, T. Walsh, Verifying properties of binarized deep neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 32, 2018.
54. C.-H. Cheng, G. Nührenberg, C.-H. Huang, H. Ruess, Verification of binarized neural networks via inter-neuron factoring: (short paper), in: Verified Software. Theories, Tools, and Experiments: 10th International Conference, VSTTE 2018, Oxford, UK, July 18–19, 2018, Revised Selected Papers 10, Springer, 2018, pp. 279–290.
55. C.-H. Cheng, G. Nührenberg, H. Ruess, Maximum resilience of artificial neural networks, in: Automated Technology for Verification and Analysis: 15th International Symposium, ATVA 2017, Pune, India, October 3–6, 2017, Proceedings 15, Springer, 2017, pp. 251–268.
56. A. Lomuscio, L. Maganti, An approach to reachability analysis for feed-forward relu neural networks, arXiv preprint arXiv:1706.07351 (2017).
57. T. Gehr, M. Mirman, D. Drachsler-Cohen, P. Tsankov, S. Chaudhuri, M. Vechev, Ai2: Safety and robustness certification of neural networks with abstract interpretation, in: 2018 IEEE symposium on security and privacy (SP), IEEE, 2018, pp. 3–18.
58. M. Mirman, T. Gehr, M. Vechev, Differentiable abstract interpretation for provably robust neural networks, in: International Conference on Machine Learning, 2018, pp. 3575–3583.
59. E. Wong, Z. Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope, in: International Conference on Machine Learning, 2018, pp. 5283–5292.
60. K. Dvijotham, R. Stanforth, S. Gowal, T. A. Mann, P. Kohli, A dual approach to scalable verification of deep networks., in: UAI, Vol. 1, 2018, p. 3.
61. S. Wang, K. Pei, J. Whitehouse, J. Yang, S. Jana, Formal security analysis of neural networks using symbolic intervals, in: 27th {USENIX} Security Symposium ({USENIX} Security 18), 2018, pp. 1599–1614.
62. J. Peck, J. Roels, B. Goossens, Y. Saeys, Lower bounds on the robustness to adversarial perturbations, Advances in Neural Information Processing Systems 30 (2017).
63. A. Neumaier, O. Shcherbina, Safe bounds in linear and mixed-integer linear programming, Mathematical Programming 99 (2004) 283–296.
64. W. Ruan, X. Huang, M. Kwiatkowska, Reachability analysis of deep neural networks with provable guarantees, arXiv preprint arXiv:1805.02242 (2018).
65. T.-W. Weng, H. Zhang, P.-Y. Chen, J. Yi, D. Su, Y. Gao, C.-J. Hsieh, L. Daniel, Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach, in: ICLR2018, 2018.

66. S. Webb, T. Rainforth, Y. W. Teh, M. P. Kumar, A statistical approach to assessing neural network robustness, in: International Conference on Learning Representations.
67. B. Wang, S. Webb, T. Rainforth, Statistically robust neural network classification, in: Uncertainty in Artificial Intelligence, PMLR, 2021, pp. 1735–1745.
68. X. Zhao, W. Huang, A. Banks, V. Cox, D. Flynn, S. Schewe, X. Huang, Assessing the reliability of deep learning classifiers through robustness evaluation and operational profiles, Workshop on AI Safety at IJCAI-21 (2021).
69. K. Pei, Y. Cao, J. Yang, S. Jana, Deepxplore: Automated whitebox testing of deep learning systems, in: proceedings of the 26th Symposium on Operating Systems Principles, 2017, pp. 1–18.
70. L. Ma, F. Juefei-Xu, J. Sun, C. Chen, T. Su, F. Zhang, M. Xue, B. Li, L. Li, Y. Liu, J. Zhao, Y. Wang, DeepGauge: Comprehensive and multi-granularity testing criteria for gauging the robustness of deep learning systems, in: Automated Software Engineering (ASE), 33rd IEEE/ACM International Conference on, 2018.
71. Y. Sun, M. Wu, W. Ruan, X. Huang, M. Kwiatkowska, D. Kroening, Deepconcolic: Testing and debugging deep neural networks, in: (ICSE2019), 2019.
72. X. Du, X. Xie, Y. Li, L. Ma, Y. Liu, J. Zhao, Deepstellar: Model-based quantitative analysis of stateful deep learning systems, in: Proc. of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 477–487.
73. W. Huang, Y. Sun, X. Zhao, J. Sharp, W. Ruan, J. Meng, X. Huang, Coverage-guided testing for recurrent neural networks, IEEE Transactions on Reliability 71 (3) (2021) 1191–1206.
74. D. Berend, Distribution awareness for ai system testing, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), IEEE, 2021, pp. 96–98.
75. S. Dola, M. B. Dwyer, M. L. Soffa, Distribution-aware testing of neural networks using generative models, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), IEEE, 2021, pp. 226–237.
76. T. Byun, A. Vijayakumar, S. Rayadurgam, D. Cofer, Manifold-based test generation for image classifiers, in: 2020 IEEE International Conference On Artificial Intelligence Testing (AITest), IEEE, 2020, pp. 15–22.
77. F. Toledo, D. Shriver, S. Elbaum, M. B. Dwyer, Distribution models for falsification and verification of dnns, in: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE, 2021, pp. 317–329.
78. W. Huang, X. Zhao, A. Banks, V. Cox, X. Huang, Hierarchical distribution-aware testing of deep learning, arXiv preprint arXiv:2205.08589 (2022).