Computational Intelligence

WILEY

Computational Intelligence

ORIGINAL ARTICLE OPEN ACCESS

Safety Monitoring of Machine Learning Perception Functions: A Survey

Raul Sena Ferreira[1,2] | Joris Guérin[1,2,3] | Kevin Delmas[4] | Jérémie Guiochet[1,2] | Hélène Waeselynck[1]

[1]LAAS/CNRS, Toulouse, France | [2]Université de Toulouse, Toulouse, France | [3]Espace-Dev, IRD, Université de Montpellier, Montpellier, France | [4]ONERA, Toulouse, France

Correspondence: Raul Sena Ferreira (raulsenaferreira@gmail.com) | Joris Guérin (joris.guerin@ird.fr)

Received: 13 July 2022 | **Revised:** 20 June 2024 | **Accepted:** 31 January 2025

Funding: This work was supported by Horizon 2020 (MSCA-ETN SAS, grant agreement No 812.788) and Investing for the Future - PIA3 (ANITI, grant agreement No ANR-19-PI3A-0004).

Keywords: fault tolerance | machine learning perception | runtime monitoring | safety-critical autonomous systems

ABSTRACT

Machine Learning (ML) models, such as deep neural networks, are widely applied in autonomous systems to perform complex perception tasks. New dependability challenges arise when ML predictions are used in safety-critical applications, like autonomous cars and surgical robots. Thus, the use of fault tolerance mechanisms, such as safety monitors, is essential to ensure the safe behavior of the system despite the occurrence of faults. This paper presents an extensive literature review on safety monitoring of perception functions using ML in a safety-critical context. In this review, we structure the existing literature to highlight key factors to consider when designing such monitors: threat identification, requirements elicitation, detection of failure, reaction, and evaluation. We also highlight the ongoing challenges associated with safety monitoring and suggest directions for future research.

1 | Introduction

Recent advances in Machine Learning (ML) have allowed autonomous systems to leave the safe environment of research labs to perform complex tasks, where failures can have catastrophic consequences. Examples of such safety-critical systems include self-driving cars [1], surgical robots [2], and unmanned aerial vehicles in urban environments [3]. These autonomous systems frequently use large ML models like neural networks for complex sensor signal interpretation, that is, perception [4], or decision-making, that is, control [5]. This paper focuses on safety mechanisms for critical physical systems relying on ML to process sensor signals.

In various autonomous system applications, essential perception tasks can only be solved using ML. For example, in highly uncontrolled settings, such as self-driving cars [6] or UAV emergency landing [7], deep neural networks must be used to detect pedestrians in RGB images. This information cannot be obtained from other approaches and is crucial to guarantee the system's safety. Despite the great success of modern ML-based perception, it introduces new dependability challenges: [8–10] 1. The *lack of well-defined specification*: ML models are learned from examples instead of manually coded, making their operational boundaries elusive, and preventing formal safety guarantees. 2. The *black-box nature* of the models: traceability and transparency of ML predictions is difficult. 3. The *high-dimensionality of data*: validation of the complete operational design domain is impossible. 4. The *over-confidence of neural networks*: output scores cannot be used as is to detect failures since a model can deliver wrong outputs with high confidence [11]. Hence, conventional offline safety measures, like fault prevention, removal, and

Raul Sena Ferreira and Joris Guérin equally contributed to this work.

forecasting [12] are often not sufficient to ensure safety and to certify these systems. Online fault tolerance mechanisms, such as Safety Monitors (SM), emerge as a promising alternative to improve safety in critical systems relying on ML perception. This paper focuses on SMs, which aims to keep the system in an acceptable state during operation, despite faults or adverse scenarios [13]. Safety monitors are mentioned in the literature under various terms, including safety kernels [14], safety managers [15], autonomous safety systems [16], checkers [17], guardian agents [18], safety bags [19], or emergency layers [20].

Most autonomous systems cannot be considered fault-free given the adverse and unspecified situations they encounter as they evolve in unstructured environments. Fault tolerance approaches aim to ensure that faults do not lead to catastrophic outcomes by designing both error detection and recovery mechanisms. In particular, SMs are components responsible for checking whether specific safety properties are violated and triggering corrective actions. A key characteristic of SMs is their simplicity, to ensure high reliability levels. Traditional SMs monitor system decisions to prevent hazards, but often take for granted the reliability of the state estimation provided by perception components [21] (Figure 1). This work considers critical systems relying on ML for perception tasks. As explained above, given the inherent complexities of ML-based perception, such outputs should not be trusted blindly to make safety-critical decisions. Therefore, it is essential to design specific safety monitoring approaches for these components to detect errors early and adapt the system's behavior.

Many existing studies on SM predominantly concentrate on devising techniques to detect unsafe ML predictions. Consequently, current surveys in this domain focus on categorizing error detection methods, as detailed in Section 2. However, SM encompasses more than merely identifying unsafe predictions. In this research, our objective is to comprehensively present the literature pertinent to the creation of robust SMs. As such, our paper is structured around the principal safety considerations intrinsic to SM design:

- What threats are being addressed by safety monitors? (Section 3)
- How to derive monitor requirements from safety objectives? (Section 4)
- Which detection mechanisms can be used? (Section 5)
- Which recovery actions can be used? (Section 6)
- How are safety monitors evaluated? (Section 7)

For each of these topics, the main challenges are presented along with existing approaches to tackle them (Figure 2). Although classic SM mechanisms can be useful to ensure the safety of

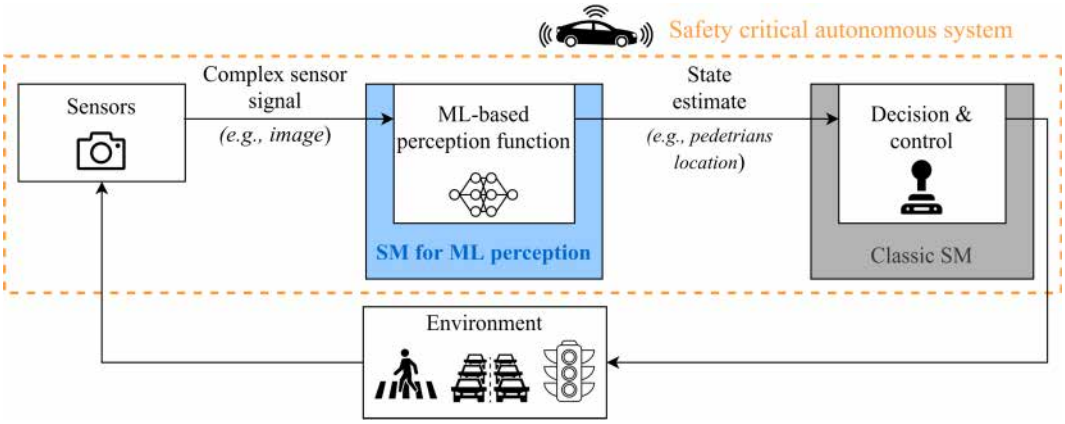

FIGURE 1 | Safety Monitors for Machine Learning-based Perception Functions. In modern autonomous systems, state estimation provided by deep learning models cannot be trusted to make safety-critical decisions. Therefore, specific fault tolerance approaches should be implemented to ensure that failures of the ML perception function will not lead to catastrophic outcomes.

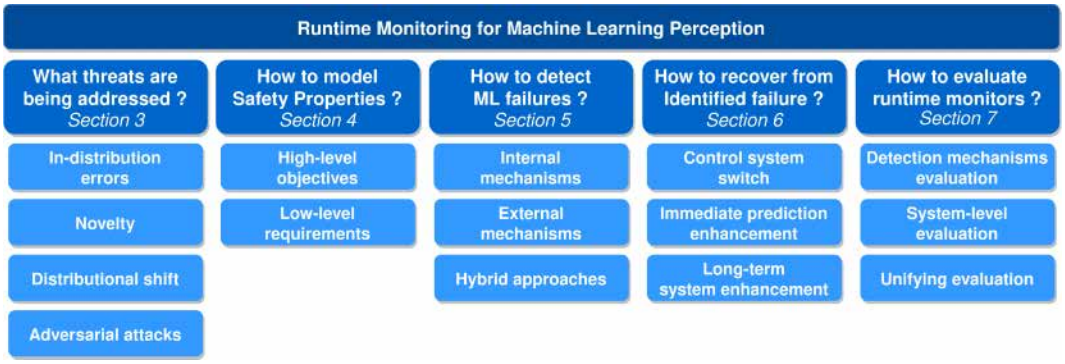

FIGURE 2 | Key questions to design, implement, and deploy reliable safety monitors for ML perception. Each section of this work discusses a specific aspect of safety monitoring of machine learning perception.

such autonomous systems, this paper focuses on the specificities related to ML perception. As shown in Figure 1, most approaches presented in this paper act close to the ML perception function, often directly considering either its inputs or outputs. However, we cast a broader net, shedding light on diverse strategies pertinent to crafting safety monitors, making this a valuable guide for safety practitioners keen on integrating ML, as well as ML experts navigating the intricacies of safety.

2 | Related Work

Recent literature about ML safety has flourished, leading to several surveys about the different facets of this domain. This section aims to delineate the contributions of these works and articulate the unique space our study occupies.

In their survey, Xu and Saleh [22] categorize the existing research on the use of ML to improve safety across various applications. Similar surveys have been proposed in domain-specific contexts like autonomous driving [23] and architecture [24]. However, these surveys primarily emphasize the benefits ML brings to safety initiatives, sidestepping the crucial aspect of ensuring the safety of the ML components themselves, which is the focus of our study.

The overview proposed by Faria [9] is closer to our objective as it presents the challenges posed by ML models in a safety-relevant context and describes potential solutions. However, it does not present the literature for safety monitoring, a critical pillar for ML safety that our survey delves deep into.

Mohseni et al. [10] also undertook an exhaustive literature review on the safety implications of ML, within the context of autonomous vehicles. Their spectrum of safety mechanisms is composed of three categories: inherently safe design, safety margins, and fail-safe mechanisms, that is, safety monitors. Regarding SM, they classify detectors by their targeted error types (uncertainty estimation, in-distribution error detection, and out-of-distribution error detection). Likewise, Pandharipande et al. [25] also discuss safety for ML-based automotive perception. For monitoring, they mostly discuss techniques to detect inputs that are outside of the expected operational design domain.

Varshney and Alemzadeh [8] consider ML safety through the lens of risk, epistemic uncertainty, and resultant harm. They discuss four categories of methods to achieve ML safety, including fail-safe strategies, echoing our emphasis on runtime monitoring. Their discussion on monitoring includes classification with rejection and uncertainty estimation techniques. They present several practical examples, but their discussion is mostly focused on the actions that are taken after ML failure has been detected, while not fully covering detection mechanisms. Recently, Mohseni et al. [26] also proposed a taxonomy of dependability techniques for ML design and deployment. Starting from engineering safety requirements, they present safety-related ML research into three categories: inherently safe design, enhancing model performance and robustness, and runtime error detection. However, regarding the specific subfield of ML monitoring, which is the focus of our work, they mostly discuss detection mechanisms (uncertainty estimation, outlier detection, and adversarial attacks detection).

Rahman et al. [27] journeyed closer to our thematic, presenting a taxonomy for runtime monitoring of ML robotic perception. They organize existing approaches to detect runtime failures into three categories: approaches using past examples of failures to predict future ones, approaches detecting inconsistencies in the perception outputs, and uncertainty estimation approaches. Similarly, Zhang et al. [28] discussed detection approaches, classifying them based on their primary objectives: failure rejection, unknown rejection, and fake rejection.

The surveys presented above are constructed in a bottom-up fashion: they first identify existing works and organize them into relevant categories. This methodology leads to discussions about ML monitoring that are incomplete, as most of the existing literature deals with the detection of unsafe predictions. Traditional safety engineering, however, recognizes error detection as just one component of runtime monitoring and other aspects are also crucial to designing and implementing safe monitors. Therefore, our study was constructed in a top-down fashion, where we first identified primary safety considerations for SM design to frame our survey's structure: threat identification, requirement elicitation, error detection, recovery mechanisms, and performance evaluation. This methodology allows us to go beyond detection mechanisms, highlight areas often overlooked in other surveys, and uncover specific areas where research is lacking from a safety perspective.

Furthermore, unlike other taxonomies that focus on targeted threats, our categorization of detection mechanisms is constructed around how the monitors are integrated within the entire system. We distinguish between internal monitors, which are inherent parts of the ML itself, and external monitors, acting independently from the monitored model. Furthermore, external monitors are classified based on the type of input they analyze – ML input, ML internal representation, or ML output. We believe that this taxonomy offers greater clarity, especially given the generic nature of many mechanisms that may be deployed against multiple types of threats [29].

3 | What Threats Are Being Addressed by Safety Monitors?

The identification of potential threats is pivotal when designing a dependable SM. Should the safety analysis reveal that a threat is likely to occur for a specific application, this should be reflected in the monitor's evaluation protocol. Different kinds of threats can affect the ML-based perception functions. Offline threats, such as poor feature engineering [30], label noise [31], inadequate ML testing [32], or bad model maintenance [33], emerge during the ML model development phase. While it is crucial to adhere to robust software engineering practices during SM development, this section mostly focuses on runtime threats, which occur during live operations.

The objective of an SM is to detect unsafe ML predictions and implement corrective actions to prevent catastrophic outcomes. Such hazardous predictions can arise from different types of

input data, which we term runtime threats. This section presents a taxonomy of runtime threats for perception functions, inspired by existing literature [34–39]. Adverse input data are classified as threats if they can be detected or mitigated using similar approaches. Throughout this section, we present runtime threats and discuss their specificities concerning detection, reaction, and evaluation.

Before diving into the different threats, we wish to emphasize that, in our recent work [29], we challenged the prevailing understanding of runtime threats in the context of monitoring. A significant portion of NN monitoring research centers on Out-Of-Distribution (OOD) detection, emphasizing the monitor's role in flagging data diverging from the training distribution. However, we highlighted the inherent ambiguity in defining "OODness": not all OOD data are inherently dangerous and not all in-distribution data are safe. We advocate for evaluating monitors based on their proficiency in detecting incorrect predictions. Nonetheless, the notion of threats delineated in this section remains crucial for SM design and evaluation. Given that a monitor's efficacy in error detection might fluctuate across diverse threats, it's paramount that every potential threat–identified through a rigorous safety analysis–is considered during the monitor's testing phase.

3.1 | In-Distribution Errors

Modern deep-learning architectures have achieved impressive results in many perception tasks. According to the latest leader board on *papers with code*[1], the top-performing model for semantic segmentation on Cityscapes [40] has a mean intersection over the union of 84.5% [41], the best model for image classification on ImageNet [42] has a top-1 accuracy close to 91%, and the leader for object detection on COCO [43] has a mean average precision around 63% [44]. These benchmarks were established on the test splits of these datasets, assumed to derive from the same distribution as the training data, denoted as In-Distribution (ID) data. While these results are excellent and allow researchers to build useful applications, even the best computer vision models are not flawless. To guarantee the system's safety, an SM should be able to handle these errors.

Beyond this fundamental model generalization issue, there is another problem: the data incompleteness. Rare conditions tend to be underrepresented since the training data only account for a small subset of all real-world possibilities [45]. Such data, presenting different characteristics than the training data, are considered Out-Of-Distribution (OOD) [46] and are discussed as different types of threats in the coming sections. There is no unified naming convention for the threats presented hereafter in the literature, but we strive to propose clear definitions to avoid ambiguity.

3.2 | Novelty Threats

A new input data encountered at runtime is considered "novel" when its category/label does not refer to any of the predefined categories known by the model [46]. For example, Blum et al. [47] studied the problem of semantic segmentation of a driving scene and trained their model on the Cityscapes dataset [40]. At runtime, when a dog crosses the road, its corresponding pixels are considered novelty as they do not belong to the predefined set of classes of Cityscapes. Hence, when a novelty input is presented to an ML perception model, it cannot return a correct answer.

The above example shows that facing novelty inputs is common for autonomous systems evolving in unstructured environments. Hence, it is crucial to equip ML perception models with defensive mechanisms against this runtime threat. A typical strategy consists of building classification models with rejection [48], with the ability to reject uncertain predictions such as objects outside the network scope. Regarding the recovery after detecting novelty threats, the actions implemented should not rely on the possibility of obtaining a better estimate of the correct prediction. Concrete approaches to detect novelty threats, recover from them, and evaluate the ability of an SM to address them are discussed in Sections 5, 6, and Section 7, respectively.

3.3 | Distributional Shift Threats

A distributional shift occurs when the marginal distribution of the runtime input data are different from the training distribution, while the label generation mechanism remains unchanged [38]. Regarding safety monitoring, we distinguish two types of shifts: covariate and semantic.

3.3.1 | Covariate Shift

A covariate shift is a condition that decreases the ML performance through time in dynamic environments [49]. In other words, covariate shift threats are new data presenting different characteristics in their composition but for which the semantic content is not different from training. For images, such threats are also called corruptions or perturbations and were presented and discussed extensively by Hendricks and Dietterich [50]. These deteriorated data can come from:

- *Failure in exteroceptive sensors*. These perturbations come from hardware defects and include various errors such as pixel traps, shifted pixels, and Gaussian noise. There are specific approaches to identify sensor faults [51], which can be addressed by tuning the sensor parameters [52] or having a backup sensor system [53].
- *Changes in external conditions*. For autonomous systems evolving in unstructured environments, the training data cannot cover all possible real-world conditions. For example, outdoor perception functions should work for different kinds of weather (e.g., snow, fog) and lighting conditions (e.g., night, sunset). As illustrated in the disengagement reports by major companies, such external factors influence the perception performance and can reduce the safety of autonomous vehicles [54].

To deal with these two covariate shift types, both traditional signal processing [55] and modern deep learning approaches [56] have been used to detect and reduce data noise. However, the techniques used against covariate shift threats depend highly on the amount of noise in the data.

3.3.2 | Semantic Shift

A semantic shift threat refers to images showcasing objects that:

- presents different attributes than known members of this category, such as a pedestrian detector trained in summer encountering individuals wearing winter clothes [57],
- displays uncommon interactions between known classes and their surroundings, like an overturned truck [58].

In contrast to novelty threats, semantic shift only includes data where the objects align with the model's predefined categories. While one might see a semantic shift as a particular case of covariate shift, its distinct challenges in safety monitoring set it apart. Notably, semantic shifts cannot be handled with denoising or backup sensors. Instead, some studies have explored the detection of object attributes that remain consistent across varying environments [59]. For instance, when confronting a pedestrian detector with shifts in clothing attributes, detecting faces rather than full bodies could be more effective. Though not designed for ML monitoring, such approaches hold promise for identifying model failures linked to semantic shift threats.

3.4 | Adversarial Threats

An adversarial input is an intentional modification of in-distribution data to make ML models fail with high confidence [60, 61]. In real-world scenarios, these malicious attacks can be made by applying modifications on targeted physical objects such as painting black lines on the road to force the ML model to interpret it as a road lane [62].

Adversarial threats can lead to serious safety issues if applied against the perception functions of critical systems. Therefore, they should be handled by specific SMs as they are likely to fool generic monitoring approaches. However, specific hardening approaches, such as gradient hiding and defensive distillation, have been developed to identify them or increase model robustness [35].

4 | How to Derive Safety Monitors From Safety Objectives?

Safety monitoring guarantees that some safety properties are not violated despite potential faults occurring in the main system. The elicitation and modeling of these properties are essential steps in designing safety monitors. For instance, Machin et al. [13] used a HAZOP-UML hazard analysis [63] to identify high-level safety objectives expressed in natural language. These high-level objectives are then converted to low-level safety requirements, expressed formally in the system's state space, and observable by the monitor. For a mobile robotic platform in a standard industrial setting, an example of a high-level safety objective is "the robot platform must not collide with a human." A low-level safety requirement can be derived by comparing the braking distance with the distance of any obstacle sensed by a laser. In this example, the low-level requirements are easy to express and implement since the sensor signal can be interpreted in terms of the high-level requirement.

The high-level safety objectives can still be identified using standard hazard analysis tools for complex systems involving machine learning perception. However, converting them into low-level monitoring requirements is not straightforward. Indeed, expressing and implementing a high-level requirement in raw sensors can result in solutions that are too conservative or even infeasible to be deployed at runtime. For example, if we consider an emergency braking system (EBS) implemented in an autonomous vehicle:

- Using *simple sensor signals* such as a laser is not enough to capture the semantic information required to distinguish between pedestrians and other moving vehicles. Such semantic information is crucial for EBS to perform two very different low-level requirements: to stop the ego vehicle when the EBS identifies an object as a pedestrian, or slightly decelerate the ego vehicle when the EBS identifies an object as a moving vehicle. Therefore, stopping the car for all sensed objects is *too conservative*, which would significantly alter the availability of the system.
- Using *complex sensor signals* such as RGB image pixels from camera sensors is not enough to guarantee that a high-level objective is not violated. That is, measuring the pixels alone is *infeasible* to perform the EBS task since such raw RGB values cannot give insightful information for the EBS to perform a high-level requirement such as avoiding a collision.

Hence, we should specifically monitor the ML function responsible for localizing pedestrians. In other words, the system-level safety objectives should be expressed as variables related to the ML model (input, activation, output).

As explained above, most current works on ML monitoring focus on detecting when a model is wrong and should not be trusted. This is a good generic formulation of the problem, agnostic of the system in which the model is embedded. However, we believe that using information from the application context to refine the low-level monitor requirements is a promising research direction. In particular, the hazard analysis of the system could be used to identify safety-critical regions of the ML model input/output space or to understand under which system configuration an ML error is hazardous. In addition, building monitors for specific subregions of the state space might allow us to come up with more effective local monitors and better allocation of resources.

Although this lead has not yet been explored for ML monitoring, some research from ML safety could serve as a first step toward building better specific monitors. In their work, Dreossi et al. [64, 65] propose to identify regions in the state space where a failure of the ML model results in a violation of a formal specification. For an autonomous vehicle use case, they show that errors of an ML-based obstacle detection model are only threats for certain state configurations (speed and distance to other vehicles). On the other hand, Salay et al. [66] introduced an approach called Classification Failure Mode Effects Analysis (CFMEA) to study the safety of an ML classifier. It serves to identify the kind of errors that can lead to a safety-critical situation. For example, CFMEA can assess the severity of different control actions based on different classification errors in an autonomous vehicle scenario.

This approach represents a promising research direction for runtime monitoring of ML perception functions. For example, knowing that an ML failure would only cause catastrophic events in some subsets of the state space could help to collect better data to design monitors in these specific regions.

5 | Which Detection Mechanisms Can be Used for Safety Monitoring?

Traditional monitors analyze both exteroceptive (e.g., distance) and proprioceptive (e.g., speed) sensors to detect safety threats. They apply simple rules on sensor data based on formal specifications [13]. However, for complex perception functions, these monitors struggle to interpret raw sensor signals like image pixels. This section discusses recent strategies to detect errors in ML model signals. Yet, even in ML-augmented autonomous systems, traditional monitoring remains indispensable to handle other sensors and ensure the system's proper functioning.

Despite its significance, the specific field of ML safety monitoring has received limited research attention. Nonetheless, various ML techniques, hailing from subfields like uncertainty estimation, anomaly detection, ensemble methods, or multimodal perception, show potential as SM detection mechanisms. This section delves into such prospective approaches, encompassing those not specifically designed for safety monitors.

This section presents a comprehensive taxonomy of detection mechanisms. Our categorization of detectors revolves around the manner in which they are assimilated into the complete system (Figure 3). Within each category, we outline the primary approaches and assess their advantages and limitations. However, it is important to mention that current research does not allow us to reach definitive conclusions regarding the efficacy of these techniques. Indeed, the literature currently presents a varied landscape with differing evaluation methods and conflicting experimental results. Notably, a study by Ferreira et al. [37] revealed unsatisfactory results for several evaluated methods despite the good results presented in the original papers.

5.1 | Internal Mechanisms

Internal detection mechanisms are approaches where the ML model itself is trained to predict its failures. In other words, the NN is designed to return both predictions and information regarding the trust in these predictions. We classify internal mechanisms into three families of approaches.

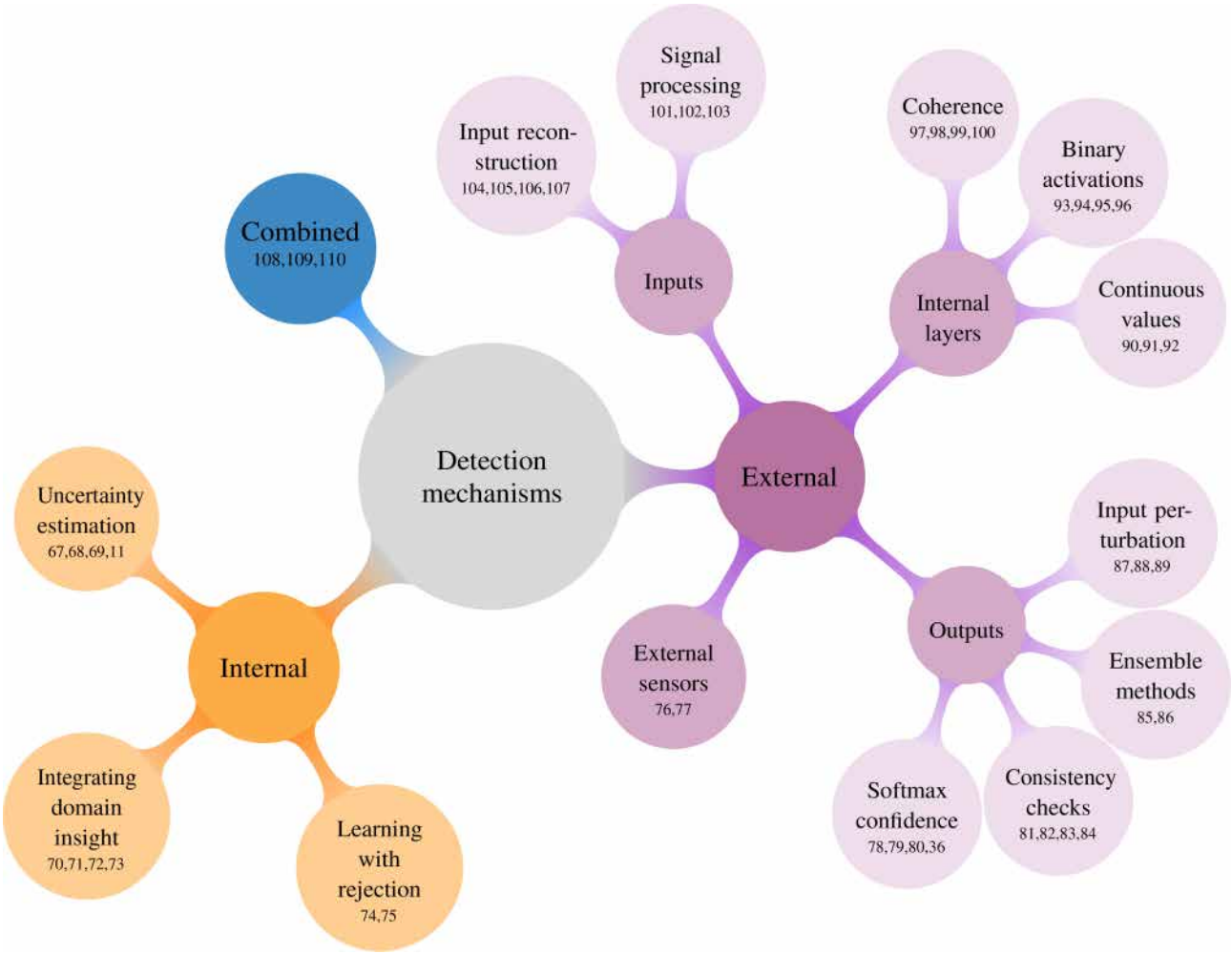

FIGURE 3 | Taxonomy of detection mechanisms. A visual representation of the different types of approaches to detect a failure of a critical ML-based perception function [67–110].

5.1.1 | Uncertainty Estimation

Uncertainty estimation in deep learning has been widely studied recently [111–113]. Most deep learning models produce a single output value per input data. Uncertainty estimation approaches replace point estimate predictions with a probability distribution over the output space. These probabilities can then be used to evaluate the risk of trusting the prediction. For example, for deep learning classifiers, the outputs of the softmax layer define a probability distribution over the possible classes. However, the community has widely questioned using raw softmax output as an uncertainty proxy. Softmax is merely a normalization technique, not designed to represent meaningful probabilities, and thus often yields overconfident predictions [78, 114].

In Bayesian Deep Learning, the weights of an NN are treated as random variables, and the objective is to learn their distributions from the training data. Then, using the Bayes rule, one can compute the distribution of the predictions. Traditional Bayesian statistics [115, 116] is a natural way to reason about uncertainty in predictive models but comes with a prohibitive computational cost to be used in practice. Recently, various approaches have been proposed to compute approximate Bayesian inference on large ML models, including Variational Inference [117–120], Laplace approximation [99], and sampling methods [121]. For a detailed review of Bayesian deep learning, we refer the reader to the following works [111, 122].

Bayesian deep learning has been applied to various perception tasks related to autonomous systems, including object detection [123], semantic segmentation [67, 124], end-to-end vehicle control [68], or visual odometry [69]. Of the techniques available, those based on Monte-Carlo Dropout [11, 125] are particularly popular due to their simplicity of implementation. This approach retains Dropout layers during inference, enabling stochastic predictions. By repeatedly running the model and analyzing prediction statistics, it evaluates uncertainty.

In theory, a clear understanding of model uncertainty is truly all that is required for constructing robust SMs for ML models. However, methods that provide provably accurate uncertainty estimates are intractable for real-time tasks in autonomous systems. To circumvent this challenge, various approximation methods have been introduced. Depending on their level of simplification, they either remain too computationally expensive or fail to offer reliable uncertainty estimates suitable for safety-critical systems. However, given the rapid advancements in this research area, the landscape could change soon.

5.1.2 | Incorporating Domain Knowledge

Domain knowledge can be leveraged to improve the training of ML models. It can be incorporated into the architecture and training process in the form of logical or numerical constraints [126]. For example, a pioneering work proposed to build an object detection model by training a hierarchy of classifiers using lexical-semantic networks to represent prior knowledge about inter-class relationships [127]. This architecture can be used to detect anomalies in the runtime predictions. Likewise, information about the relationship among different superpixels of an image is used in [70] to build a robust classification pipeline. The superpixel relationships are modeled using a graph neural network, which processes the image jointly with a convolutional neural network in the final architecture. This additional information in the model itself can be used to detect incoherence in the final predictions. Ramanathan et al. [71] build an action retrieval model by incorporating other small linguistic, visual, and logical consistency-based actions to effectively identify relationships between unobserved actions from known ones. Other such approaches include attempts to incorporate symbolic knowledge [72], as well as first-order fuzzy logic to reason about logical formulas describing general properties of the data [73].

Such domain-aware approaches are notably effective as they are specifically tailored to monitor relevant elements to a particular application. However, their limitation lies in their lack of universality. Each application or task requires a unique design, demanding considerable expert effort for every subsequent model. At times, crafting them to fit certain tasks might even be unattainable. Moreover, such niche approaches seem to have a higher rate of false positives, as indicated by Ferreira et al. [37].

5.1.3 | Learning With Rejection

In the setup of selective classification—also called classification with rejection—input data can either be classified among one of the predefined categories or be rejected, that is, the system produces no prediction. Such rejection mechanisms can actually be viewed as built-in safety monitors as they allow to reject uncertain predictions at runtime. This kind of approach has been presented as a promising way to control the confidence in the monitored model in critical autonomous driving scenarios [10]. These approaches are internal mechanisms as they consist of modifying the model and learning algorithm to account for rejection. In other words, the predictor and the rejection function are trained jointly and are part of a single unified model. Several approaches have been proposed to integrate rejection options to traditional ML models such as support vector machine [128], K-nearest neighbors [129], and boosting [74]. Recently, Geifman and El-Yaniv [75] presented Selectivenet, a neural network architecture optimized to perform classification and rejection simultaneously. Several other approaches have been introduced recently for learning with rejection, as discussed in more detail in the recent survey by Hendrickx et al. [130].

Utilizing an integrated model for prediction and rejection has its advantages. When optimized together, these elements can boost performance for specific tasks, potentially reducing inherent biases due to synergistic training. However, the drawback of such integrated techniques is that any alteration to the monitor requires a complete model retraining, further taxing computational resources. We also believe that learning with rejection does not align well with today's machine learning landscape, where leveraging open-source models as a foundation is common. Indeed, such an approach prevents seamlessly integrating a monitor without the labor-intensive need for total retraining. Finally, while such unified models might diminish bias, there is a counter-argument in favor of diversifying the design. By separating the prediction and monitoring processes, a blend of expertise—spanning data scientists to safety experts—can

collaborate, fostering a multifaceted, bias-resistant system. Designing a good rejection-enabled model requires specific expertise beyond basic machine learning knowledge, which might not always be easy to find.

5.2 | External Mechanisms

External detection mechanisms are independent components in charge of monitoring the behavior of an ML model during execution. As they are not directly tied to the ML model, they are not required to be trained jointly and can be developed later by specific safety teams. We identified different mechanisms in the literature that differ in their position in the ML perception pipeline. In particular, external detection mechanisms can monitor the ML model inputs, internal representations, outputs, or even data from other sources.

5.2.1 | Monitoring the DNN Inputs

Some approaches predict failures of an ML perception model by monitoring its inputs, for example, the raw images. These approaches are independent of the monitored ML model, as they characterize the expected operational conditions under which a neural network can be used and discard new abnormal input data before the perception function processes them. Next, we present the existing approaches.

5.2.1.1 | Traditional Approaches. Traditional signal-processing approaches can be used to identify an anomalous sensory input before it enters the ML model [101]. These approaches characterize some statistical patterns of "normal" data (i.e., from the training set) and compare them with new online inputs. In particular, for images, one can identify noise patterns of the camera and standard lighting conditions and detect abnormal images using standard image processing [102]. This kind of approach has been used to identify water droplets in images for autonomous driving scenarios [103].

These approaches are good at detecting sensor defects and variations in external conditions like lighting and weather. Yet, they fall short of identifying subtle changes in complex signals, such as novelty classes or semantic shifts.

5.2.1.2 | Input Reconstruction. Recent techniques have relied on unsupervised deep learning to identify anomalous images. They start by training an auto-encoder that learns a lower-dimensional latent representation of the images and how to reconstruct the original images from it. Then, at runtime, the auto-encoder can be used to decide if a new image is an outlier by comparing its reconstruction error to a fixed threshold defined during training [104–106, 131]. Another approach consists of training an outlier detection model on the latent representation of the auto-encoder to predict the nonconformity of new inputs [107]. In practice, this family of approaches was used to identify unexpected conditions, such as weather changes, to anticipate the misbehavior of an autonomous vehicle within a simulation environment [132]. Finally, Feng and Easwaran [133] proposed to detect unusual movements in real-time by combining optical flow operation with representation learning via a variational auto-encoder.

Compared to conventional techniques, these methods have a higher likelihood of detecting semantic shifts in data, given their reliance on deep representation learning from the training dataset. However, two neural networks trained on the same dataset may not necessarily learn identical features [134]. Relying on such an auto-encoder could introduce an additional bias if its generalization diverges from that of the model being monitored.

5.2.1.3 | Introspection. A well-explored technique in the robotics domain aims at predicting future failures at runtime. For instance, Gurau et al. [135] proposed two models that predict perception performance from observations gathered over time. Then, the monitor can switch control to a human operator if the robot's perception system is predicted to underperform. Kuhn et al. [136] proposed an introspective approach to predict future disengagements of the car by learning from previous disengagement sequences. They monitor both input images and other state data from the car.

Relative to other input monitoring methods, these techniques stand out as they strive to assimilate the underlying knowledge of the monitored ML. They learn a fresh data representation to identify unsafe input under the supervision of the model itself. However, it appears that these methods could be enhanced by accessing additional details about the monitored model, like its activations, which might aid in devising superior models for detecting unsafe input data.

5.2.1.4 | Insights Regarding Input Monitoring. Input monitoring techniques present the advantage of being independent of the monitored neural network, which facilitates the software engineering process. Yet, this very independence can be a double-edged sword, making it challenging to anticipate ML failures on specific inputs without direct model inspection. Such techniques are effective when the operational design domain (ODD) is clearly defined and the predictor is reliably error-free within its ODD. Otherwise, employing techniques tailored to the specific predictor is advantageous, as they are more likely to capture the model's true strengths and vulnerabilities.

5.2.2 | Monitoring the DNN Internal Representations

Other detection approaches monitor the values from the ML model's hidden layers. The underlying principle is that the training data alone does not fully encapsulate the model's understanding and that crucial information is embedded within the model's weights. This can be justified intuitively by the fact that different models behave differently with new inputs, even when trained on the same dataset [134, 137]. This section discusses approaches to monitor neural network activations at runtime.

5.2.2.1 | Continuous Layer Values. Several works have proposed to detect unreliable predictions by analyzing the output values of certain layers for novel input data. Rahman et al. [90] trained a binary anomaly classifier on features extracted from a hidden layer of a neural network. Another recent work proposed truncating feature activations from the penultimate layer, before the classification head, to get better uncertainty estimates [91]. Adopting a different methodology, Lukina et al. [92] employed a

centroid-based clustering technique on the internal representations of a designated layer to characterize known inputs, leveraging the distance to cluster centers as an indicator to filter out atypical data at runtime. Finally, Wang et al. [138] proposed to monitor the neurons within a faster R-CNN by representing distributions of activation patterns and calculating the Kullback–Leibler divergence between them.

The raw values of internal layers activation contain rich information about the data being processed. Initial layers capture raw data intricacies while subsequent layers focus on data interpretation. Yet, the volume of features extracted from these layers can be extensive, leading to significant computational demands, which might be unfeasible for some constrained real-time applications.

5.2.2.2 | Binary Layer Activations. To reduce the memory usage of internal representation monitors, other works have proposed looking only at the binary activations of a given layer. As ReLU is one of the most popular activation functions in deep neural networks, one can inspect whether a new input is triggering an activation of a specific neuron or not, that is, nonzero value. The advantage of considering such binary variables is that they can be stored easily using binary decision diagrams [93] or abstraction boxes [94–96]. Then, abnormal data are identified by comparing activation patterns encountered at runtime to those recorded during training.

5.2.2.3 | Coherence Between Layers. Several works have explored the potential of simultaneously examining multiple hidden layers. Wang et al. [97] introduced Dissector, a tool designed to ascertain if the outputs across different layers yield consistent decisions. Schorn and Gauerhof [98] proposed an approach called FACER, which builds a feature vector capturing activations from various layers by summing the values of each feature map. Similarly, Lee et al. [99] fitted class conditional Gaussian distributions to both low-level and upper-level features of the deep learning model and defined a confidence score based on the Mahalanobis distance. Another innovative approach consists of tracking the model's internal representations backward to build a saliency map of a given input [100]. The patterns of this map are then compared with the ones obtained for the training set within the same category. Recently, Wang et al. [139] proposed ViM (Virtual-logit Matching), combining a feature-based class-agnostic score and a logits-based class-dependent score, which obtained very good results for detecting out-of-distribution data.

Leveraging multiple hidden layers often proves advantageous, primarily because it alleviates the challenge associated with optimal monitoring layer selection. Yet, this approach compounds the issues related to computational time. As such, striking a balance between the number of layers to incorporate and the system's inherent constraints becomes pivotal.

5.2.2.4 | Insights Regarding Internal Layers Monitoring. The activations within internal layers provide an invaluable insight into a model's comprehension of the data it processes, making them pivotal for crafting effective monitors. However, the challenge lies in pinpointing the right layer for monitoring to ensure optimal outcomes—a decision far from straightforward. Marrying the ideal layer with the most suitable monitoring transformation is equally vital. Owing to these intricate selections, current results showcase significant variance. Different methodologies and layers excel for varying datasets, but the community has yet to converge on definitive guidelines on which techniques and layers best serve specific scenarios. Consequently, formulating an effective internal layer monitor remains a challenging endeavor.

5.2.3 | Monitoring the DL Outputs

Some detection approaches monitor the ML outputs. For example, for classification tasks, the output contains information about the target class and the model's confidence associated with this label.

5.2.3.1 | Manipulation of Softmax Confidence. A straightforward strategy to monitor deep neural network outputs consists of properly establishing what values of the (softmax) confidence score can be considered reliable [78]. Several enhancements over this baseline have been proposed to address the calibration issues from softmax confidence scores. Liang et al. [79] presented ODIN, which uses temperature scaling and small input perturbations to separate the softmax scores between in- and out-of-distribution data. Hsu et al. [80] modified this approach to function without requiring out-of-distribution examples and further enhanced detection capabilities using confidence score decomposition. A more recent methodology, DOCTOR, aimed to characterize the optimal discriminator, focusing solely on the softmax probability [36]. Finally, Liu et al. [140] questioned softmax's efficacy for reliable uncertainty estimation, suggesting an alternative energy score for logit transformation.

The main advantage of these techniques lies in their straightforward implementation, only requiring attention to the neural network's outputs, and ensuring minimal computational and memory burden. They have shown promise in detecting threats such as novelty, covariate shifts, and adversarial attacks. However, their performance appears to drop for the task of detecting actual model errors. It's also worth noting that these strategies primarily cater to monitoring classification models.

5.2.3.2 | Consistency Checking. Several approaches focus on verifying the spatial or temporal consistency of a sequence of predictions. One primary method involves employing expert knowledge to establish constraints on output sequences. For instance, Kang et al. [81] built a monitor for object detection by identifying flickering, that is, an object should not keep appearing and disappearing in successive frames of a video. Another strategy, showcased by Harper et al. [82], translates the official highway code's rules and clauses into logical assertions for monitoring.

Alternatively, these patterns can be learned from data. Chen et al. [83] proposed a logical framework to evaluate both temporal and spatial coherence of bounding box predictions to identify erroneous detections. Temporal coherence focuses on how bounding box labels evolve across sequential frames, while spatial coherence learns standard bounding box sizes at various locations. On another note, Guerin et al. [84] proposed a consistency monitor

tailored for objects under periodic motion, like those on production lines. They train a Gaussian process to estimate the probability for a bounding box to be at a particular location at a specific time and use it to discard erroneous detections. Finally, Rabiee and Biswas [141] detect failures of stereo vision-based perception through inconsistencies in plans generated by a vision and a supervisory sensor.

These techniques are primarily tailored for object detection tasks. One of the advantages of these methods is the potential to gather samples for subsequent ML training, either through human annotation or weak supervision.

5.2.3.3 | Ensemble Methods. Using an ensemble of ML models is a conventional approach to enhance robustness. An effective ensemble should consist of models that are "good," "independent," and "sufficiently numerous." For complex ML used in perception tasks, such ensembles can be composed of models with the same architecture but trained to identify different aspects of the data or with different architectures trained with the same data [142]. For deep neural networks, Gontijo et al. [137] conducted a study about the influence of different training parameters on what the model knows, which can be used to maximize the ensembling benefits.

While neural network ensembles are commonly used to enhance the robustness of ML systems, they can also serve as effective tools for monitoring predictions. Ensembles provide additional information, such as the level of agreement among individual components, which can be valuable for assessing an ML system's confidence in its predictions. When computational resources are not a limiting factor, building ensembles of deep neural networks presents a promising approach to monitor the coherence between individual models and mitigate the propagation of errors from a single neural network.

In practice, different voting strategies were proposed to address anomaly detection. Yahaya et al. [85] weights each model vote based on its performance and a score for what is considered normal. Roitberg et al. [86] leverage the estimated uncertainty of each prediction to measure novelty. Roy et al. [143] present a runtime monitor based on predictive processing and dual-process theory. They developed a bottom-up neural network comprising two layers: 1. a feature density model that learns the joint distribution of the original inputs, outputs, and the model's explanation for its decisions. 2. a graph Markov neural network that captures an even broader context.

5.2.3.4 | Robustness to Input Perturbation. To verify if a new input can be considered safe, it was proposed to measure its sensitivity to input perturbations. Such perturbations can be applied either to the data (e.g., image compression [87]) or to the ML model itself through random mutations [88]. Another possibility is to check the stability of a model within a radius distance calibrated during the training [89]. The underlying hypothesis for these approaches is that, for valid input, the outputs of the neural network should be robust to small perturbations.

This approach is mostly used to detect adversarial inputs, but it has also been used for practical scenarios involving critical systems. For example, Zhang et al. [144] proposed DeepRoad, a GAN-based metamorphic testing and input validation framework for autonomous driving systems.

Similarly to ensemble methods, a primary limitation of these techniques is their need for multiple neural network forward passes for each input. This can lead to extended processing times or substantial computing resource demands, which may not suit many autonomous systems, especially those with real-time and power efficiency constraints.

5.2.3.5 | Insights About Output Monitoring. Observing the neural network outputs has the inherent benefit of directly reflecting the essential information relayed to the system's decision module. In contrast, the output layer, compared to early internal representations, contains less granular details, necessitating the incorporation of additional context for optimal detection outcomes.

5.2.4 | External Sensors

To conclude this section, we present one last family of approaches to detect DNN failures. It aims at checking whether the ML predictions are consistent with signals coming from other sensors. This way, Zhou et al. [76] proposed to use an additional LIDAR sensor to monitor the runtime behavior of a semantic segmentation model. By checking the consistency of geometric properties between the predicted segmentation map and the LIDAR points, they can measure the segmentation model's accuracy at runtime. Similarly, Ramanagopal et al. [77] used a second camera to monitor the results of an object detection model. Inconsistencies in the object detector outputs between a pair of similar images are used as a hypothesis to detect false negatives, that is, missed detections. Finally, Li et al. [145] present an approach to monitor extrinsic camera calibration quality by using inertial measurement unit (IMU) data to capture mismatches of road image features.

These techniques are very powerful in detecting inconsistencies in ML predictions, bolstering the safety of autonomous systems with ML-driven perception. However, their effectiveness is primarily confined to ML tasks with a geometrical component, like segmentation or detection. For purely semantic tasks, such as classification, simple sensor signals cannot be used to provide adequate monitoring.

5.3 | Combined Detection Approaches

Various approaches have combined monitors to build more robust detection systems. Loquercio et al. [108] proposed to monitor data and model uncertainty using distinct mechanisms. Data uncertainty is assessed by propagating sensor noise characteristics through the network, while model uncertainty is assessed using Monte-Carlo Dropout [11]. The cumulative uncertainty is obtained by combining both sources using stochastic Assumed Density Filtering. Buerkle et al. [110] presented an approach using two types of detection mechanisms called sensor checks (monitoring model input with an auto-encoder) and plausibility checks (monitoring spatiotemporal coherence of model predictions). Meanwhile, Cofer et al. [109] integrated four distinct monitors for end-to-end aircraft taxiing, harmonizing data from

various sensors, standard computer vision algorithms, and input reconstruction approaches to forge a resilient neural network monitoring mechanism. Lastly, Guerin et al. [7] combined three monitors for drone emergency landings: Monte-Carlo Dropout, local resolution enhancement, and classification hierarchy.

Throughout this section, we've delved into a myriad of recent techniques for detecting unsafe predictions. These methods each bring unique qualities to the table, addressing various data types, tasks, implementation nuances, and integration methodologies. While the individual efficacy of these techniques in the realm of safe autonomous systems is still under examination, their collective use holds substantial promise in bolstering safety and paving the way for system certification. However, pinpointing the optimal amalgamation of these strategies remains a challenge, necessitating a profound understanding of prevailing methodologies, proper identification of potential threats through safety analysis, and a good understanding of system constraints and requirements. Through this survey, we aspire to assist in this complex endeavor by offering safety experts a complete view of the existing methodologies and illuminating potential pathways for the creation of more proficient combined monitors.

6 | Which Recovery Mechanisms Can be Used to Build Safety Monitors?

In Section 5, we explored a plethora of methods aimed at detecting unsafe predictions stemming from ML-based perception functions. When activated, these detection mechanisms raise a safety alert to the autonomous system, necessitating immediate and appropriate measures to avoid hazardous situations. In most ML monitoring studies, the basic alert is a basic flag, and more complex actions are usually not investigated. We call such actions *recovery mechanisms*, and this section is dedicated to their detailed examination.

6.1 | Switching the Control System

The most straightforward and widely used approach when detecting a potentially dangerous error is to switch to a simpler control algorithm, not relying on the faulty ML perception component. Phan et al. [146] proposed the neural simplex architecture, which switches from a high-performance ML-based controller to a simpler safe controller when unsafe behavior is detected. Adapting this architecture to ML-based perception functions would be highly valuable but challenging. Indeed, in many practical scenarios, it is hard to design simple controllers that do not rely on ML for state estimation. As a result, for complex autonomous systems relying on visual perception, the default recovery actions often consist of switching back control to a human driver [106] or triggering an emergency braking [109]. The former can be hazardous when a fast reaction is required. The latter might not be safe for specific scenarios, for example, autonomous cars driving on the highway. We believe that the challenging task of coming up with control procedures to bring complex autonomous systems back to safety is of significant importance and should receive more attention.

6.2 | Immediate Prediction Enhancement

For some specific threats, adapted safety measures can be used to improve the predictions of the ML-based perception function and increase confidence in the system.

6.2.1 | Input Reconstruction

A family of approaches consists of improving the quality of the ML inputs. These techniques are used when one can identify the cause of the wrong prediction and mostly consist of removing the specific type of noise. Recently, most image-denoising approaches use autoencoders, trained to reconstruct the original image without the identified noise pattern [147]. For example, image dehazing algorithms can be used to react to fog or smoke [148]. Other approaches have been proposed to react to different light exposure conditions [149, 150], saturation [151], water drops [103, 152], or even more standard Gaussian noise or impulse noise [153, 154]. Other works proposed to enhance the original image by increasing its resolution artificially [155] by using convolutional neural networks [156] and generative adversarial networks [157, 158]. Furthermore, to deal with images having chunks of pixels damaged by sensor failures, image inpainting techniques can be used [159]. Finally, to handle errors related to incomplete color information (mosaic-styled images), some works have applied demosaicing techniques [160] or gradient-based feature extraction [161].

6.2.2 | Changing Final Prediction

To respond to the detection of an adversarial example, Al-Afandi and Horvath [162] proposed to exclude the predicted classes (corresponding to the attack) and to study the resulting loss landscape to recover the original class. On the other hand, Li et al. [163] proposed an approach to reverse the effect of an adversarial attack on a classifier by studying the behavior of adversarial examples and establishing a mapping between true classes and predicted classes. Whether similar approaches can be used to respond to different kinds of threats is still an open question that is worth investigating. For example, the work by Salay et al. [66] could be extended to downgrade classification at runtime, for example, if a high classification uncertainty is detected between the classes "car" and "truck," the prediction could be changed to the sup-class "vehicle."

6.2.3 | Using Alternative Components

When an error is detected in the state estimation, a possible approach is to substitute some components of the perception pipeline and recompute its output. For example, when a sensor failure is causing the perception error, one can rely on existing backup sensors to still compute the expected prediction [164]. When high uncertainty is detected, one might use sensors with higher resolution locally (spatially or temporally) to get a better prediction [7]. However, these high-resolution sensors might not be usable in the regular operation pipeline because of processing

time or energy consumption constraints. Another idea is to use an ensemble of ML models with different coverage mechanisms to replace unsafe predictions [137].

6.3 | Impact on Long-Term System Enhancement

Finally, we also discuss how runtime threats identified by a monitoring detection mechanism can be used to improve the system's safety in the long run. A common strategy in the industry is to store a huge amount of frames while the system is running for posterior offline labeling and model retraining [165]. Post-labeling is usually done manually, using other sensors, or automatically using other ML models. Specific continual-learning approaches exist for model retraining, particularly to avoid catastrophic forgetting when incorporating novel classes [166, 167]. If one chooses to collect data detected as unsafe by a monitoring system for retraining, it will have a positive long-term impact on the safety of the system.

Although these approaches do not guarantee the immediate safety of the system, they are essential to reduce safety-critical errors in the long run. In addition, storing data detected as threats can also be useful for building relevant datasets to test future developments of safety-critical systems.

7 | How to Evaluate Safety Monitors?

A proper evaluation protocol for safety monitoring should ensure that the Safety Monitor (SM) presents the following characteristics:

1. It guarantees that the system never reaches any safety-critical state.
2. It maintains a high availability of the system.
3. It complies with the runtime constraints of the system (execution time, hardware capacity).

This section delves into the evaluation practices adopted for ML safety monitoring. We first examine the assessment methods for detection mechanisms. Being generic modules, they often permit independent evaluations detached from the complete system. Then, we explore the existing evaluation protocols used for approaches that monitor ML components at the system level. Finally, we discuss how the impact of research on SM could be increased by standardizing evaluation methodologies.

7.1 | Evaluation of Detection Mechanisms

Most detection approaches intend to identify unsafe input data, which are likely to produce erroneous output when processed by the ML model. Ensuring the efficacy and safety of these methods necessitates the crafting of pertinent test datasets, coupled with the application of appropriate evaluation metrics. This section is dedicated to elucidating these crucial aspects.

7.1.1 | Evaluation Datasets

In this section, we present how datasets are built to evaluate safety monitoring detection mechanisms. This process can be seen as fault injection, focusing on the threats presented in Section 3.

7.1.1.1 | Novelty Detection. Novelty detection aims at identifying when the label of an input does not belong to any of the predefined classes handled by the ML model. Thus, to evaluate the capacity of an SM to identify novel inputs, most approaches have used modified versions of standard image classification datasets such as Imagenet [42], MNIST [168] or CIFAR [169]. In particular, two main strategies can be used to create novelty detection datasets. The first one consists of merging two datasets with nonoverlapping class labels. One dataset is used to fit the model (training split), and its test split represents the in-distribution data for evaluation, while the other dataset serves as novelty data. This approach was used for the experiments of many papers on out-of-distribution detection [37, 78–80, 91, 98, 99, 170]. The second strategy splits a single dataset into two subsets with disjoint labels [86, 92, 94–96, 104, 105]. Recently, Wang et al. [139] attempted to build a less ambiguous novelty benchmark dataset by asking two independent human annotators to label novelty images for ImageNet.

7.1.1.2 | Distributional Shift Detection. A distributional shift occurs when the marginal distribution of the runtime input data differs from the training distribution, while the label set does not change. It can come from sensor failures, changes in external conditions, or modifications to the environment itself. Thus, most papers have relied on injecting perturbations to test images to evaluate runtime detection mechanisms for detecting such shifted data. Several papers have proposed to inject artificial corruption [50] into standard image classification datasets [37, 78–80, 90, 98] Others have injected faults into realistic autonomous systems scenarios. Some have used autonomous vehicle simulators, such as CARLA [171], to simulate different kinds of driving conditions (weather, light) [39, 107, 133]. Others have applied artificial perturbations to existing real-world datasets for autonomous driving [76, 83, 100, 144] or UAV emergency landing [7].

7.1.1.3 | Adversarial Detection. This setting represents an intentional modification of in-distribution data to make a deep learning model fail. The main approach to evaluate an SM detector at this task consists of applying an adversarial attack (see Section 3.4 for examples) to the test dataset under evaluation to constitute a binary classification dataset containing both normal and attacked images. This has been applied to standard image datasets [37, 87–89] and simulated autonomous driving scenarios [100, 107].

7.1.2 | Evaluation Metrics

In Section 3, we highlighted two competing perspectives in the domain of unsafe ML prediction detection. One view centers on out-of-distribution (OOD) detection, which seeks to recog-

nize specific runtime threats, assuming that all standard data should be embraced, while any altered data should be excluded. In contrast, the out-of-model-scope (OMS) detection focuses on pinpointing the ML model's actual prediction errors. This latter perspective has been adopted in several research works [36, 37, 78, 81, 93, 97]. In our recent work [29], we argue for broader adoption of OMS detection, postulating that the concept of "OODness" is ambiguous and that in-distribution errors should also be addressed by competent monitors. Despite these distinct paradigms, the evaluation datasets remain consistent across both views.

Detection mechanisms are binary classifiers, hence, switching between these perspectives boils down to toggling the binary monitoring labels. Then, the performance of a monitoring approach is gauged through standard binary classification metrics. In this work, we consider that a True Positive (TP) is a rejected invalid input, while a True Negative (TN) is an accepted valid input. A quick rundown of binary classification metrics includes:

- *Accuracy*: Proportion of correctly classified inputs. It can be misleading when the dataset is not balanced.
- *FP rate*: Proportion of valid inputs that were rejected.
- *FN rate*: Proportion of invalid inputs that were missed.
- *TPR@95TNR*: TP Rate when the TN Rate is 0.95. It represents the probability of finding invalid data when the rejection threshold is set so that 95% of valid data are accepted.
- *AUROC*: Area Under the Receiver Operating Characteristic (TPR against FPR). AUROC is threshold-independent and represents the probability that rejection scores of valid inputs are lower than invalid ones.
- *Precision*: Proportion of rejected inputs that were invalid.
- *Recall*: Proportion of invalid inputs that were rejected.
- *AUPR*: Area Under the Precision-Recall curve. AUPR is better than AUROC when the positive class and negative class have greatly differing base rates.
- *P@80R*: Precision when the recall is set to 0.8 [90].
- *F1-score*: Harmonic mean of the precision and recall. This score represents a unified performance evaluation when the rejection threshold has been fixed.
- *Matthews Correlation Coefficient*: It accounts for all categories of the confusion matrix (TP, FP, TN, FN).

When the monitored model addresses a different task than classification (e.g., regression), the definition of prediction failure is not as straightforward. For example, a neural network predicting the steering angle of a vehicle for the next time step will always commit some degree of error, and defining failure requires choosing a threshold for these errors. For such cases, to assess the performance of a detection mechanism, one can compare the values of task-specific metrics between accepted and rejected images. Examples of such metrics include average precision for object detection, mean squared error for regression tasks, and intersection over union for semantic segmentation.

Furthermore, it's crucial to account for the detection mechanism's computational performance in terms of execution time and memory usage to gauge the overall system overhead when integrating such safety monitors.

7.2 | System-Level Evaluation

When an ML-based perception component is embedded into the control loop of an autonomous system, not all prediction errors will lead to the same outcomes. For example, some perception errors can generate catastrophic events (e.g., missing a pedestrian crossing a road) [66]. In contrast, others might not even change the system's behavior (e.g., detecting a tree as a street lamp). In addition, Haq et al. [172] showed that offline testing (unit tests) is more optimistic than online testing (simulation) since several safety violations that were identified in simulation could not be identified offline. For this reason, it is important to evaluate how well a safety monitor is performing within the context of the system in which it is integrated.

Recent works have designed safety monitors in a real-world application context, where the impact of a prediction by the perception component can be assessed. For such cases, it is thus possible to evaluate different aspects of the performance of an SM, such as the added safety and the loss of system availability. An example was proposed by Stocco et al. [106, 132] where the monitor is implemented in a simulation environment for an end-to-end autonomous driving scenario. This way, they can play the same scenarios with and without the monitor and evaluate when critical misbehavior has been avoided (added safety) and when interventions were unnecessary (loss of availability). Another simulation-based SM evaluation was conducted by Cofer et al. [109] for an aircraft taxiing application. On their experimental dataset, they were able to avoid all the cases where the neural network led the aircraft to exit the runway thanks to their safety monitoring system. On another note, Guerin et al. [7] evaluated safety monitors for a drone emergency landing scenario. By defining a safety score for any landing zone, they can compare the emergency landing system with and without the monitors. Different monitors can be compared based on their safety benefits to the system. Guerin et al. [173] endeavored to provide a theoretical foundation for evaluating ML monitors within autonomous systems. They delineated three metrics: Safety Gain, Residual Hazard, and Availability Loss, and demonstrated their computation across various examples of ML-enabled systems.

7.3 | Evaluation Coverage

When performing evaluation, we need to consider two different scenarios: simulated, and real-world scenarios. Each presents distinct advantages and challenges.

To test perception functions, it is frequent to use a simulation environment [106, 108, 110]. This allows to reset the environment to a previous configuration and compare the responses to different perception and monitoring outputs. However, the existence of a reality/simulation gap is frequent. Indeed, there are usually significant differences between simulation environments and

real-world scenarios, where new, unexpected threats can happen [174]. On the other hand, when evaluating a perception function on real images, the collected data never represent all possible threats to which a safety-critical system might be exposed. Nevertheless, an exhaustive safety analysis of the system might help cover a higher proportion of these threats [175].

Even when evaluations are conducted in representative test scenarios, it is hard to evaluate the performance of a perception function, and safety monitor as the ground truth is often not available at runtime. This is often referred to as the oracle problem [176]. As a result, all frames and sensor values must be recorded for off-line labeling and performance and safety evaluation. Such evaluations need to be done periodically to avoid a decrease in ML performance and system safety due to dataset shifts [177].

7.4 | Evaluation for Certification

Specific evaluation and certification procedures for autonomous systems were proposed in the literature. Myers and Saigol [178] developed a framework to assess the safety of autonomous driving by applying two types of outcome-scoring rules: prescriptive and risk-based. The first contains measurable rules, which must always be verified, while the second contains undesirable outcomes that must not occur too often. De Gelder and Den Camp [179] proposed a certification scenario for self-driving vehicles, considering three stakeholders: the applicant, the assessor, and the road/vehicle authority. The applicant applies for the approval of one specific autonomous vehicle. The assessor assesses this vehicle and advises the authority, who sets the requirements and approves the vehicle for road testing. De Gelder et al. [180] proposed a risk analysis expressed as the expected number of injuries in a potential collision to compare it to road crash statistics. The authors decompose the quantified risk into the three aspects stipulated by the ISO-26262 and ISO/DIS-21448 standards: exposure, severity, and controllability. On another note, Guerin et al. [3] assessed the requirements to certify UAV operations in urban environments using a document called Specific Operations Risk Assessment (SORA) [181], which provides guidelines to develop and certify safe UAV operations.

8 | Conclusion and Open Challenges

Machine Learning solutions are being used increasingly to build perception functions for autonomous systems, but they cannot be trusted for safety-critical applications. Safety Monitors aim to ensure that the system always remains in a safe state despite the occurrence of faults. This work presents a comprehensive survey about safety monitoring of ML perception functions, addressing every step of the development process, that is, threat identification, requirements elicitation, detection of failure and reaction, and evaluation. We present existing works related to SM and highlight the current gaps in the literature to reach the level of integrity required for such safety-critical systems. After conducting this extensive study, we consider that the field's biggest limitations and open challenges are the following.

8.1 | Defining Safety Monitoring Objectives

More research should be conducted regarding how to formulate the monitoring requirements to reflect the outcomes of the safety analysis and other relevant properties of the system. To illustrate this, we can mention a result from Ferreira et al. [37] which showed that most detection mechanisms based on out-of-distribution detection (threat identification) suffer from a high number of false positives and false negatives when considering their ability to detect a failure of the ML model. This limitation results from a misalignment between SM specifications and system-level objectives. Indeed, not all threats lead to errors, and some in-distribution images lead to wrong predictions.

8.2 | Choosing Detection and Reaction Mechanisms

Different types of detection and reaction mechanisms were presented in this work. Such approaches must be properly combined with the task at hand to build a good safety monitor, which is difficult due to the vast possibilities. This choice is highly dependent on the application context, but we believe that meaningful research could be proposed to map task characteristics to detection/reaction mechanisms. For example, identifying that certain generic detection mechanisms are suitable for specific kinds of threats would be useful. Likewise, it would be valuable to study whether specific recovery actions can improve the performance of the ML model when combined with specific detection mechanisms.

8.3 | Combining Safety Monitoring Architectures

It is probably desirable to use several monitoring approaches for different aspects of the perception task. For example, specific detection mechanisms could be responsible for different regions of the input space or different types of threats. Additionally, strategies that are not purely based on data, such as plausibility checks [182], model assertion [183], and classification failure mode and effects analysis [66], can be applied to complement data-based monitors. Studying how to combine several safety monitors and verify the consistency of their outputs is an important open challenge for the field.

8.3.1 | Implementation Constraints

The safety monitors discussed in this work are expected to perform within embedded systems. Hence, it is essential to design SMs that can function under limited computing power and memory. Additionally, they must adhere to the system's energy consumption constraints. Being executed at runtime, SMs also need to meet stringent execution time criteria to ensure seamless synchronization with the primary system being monitored.

8.3.2 | Standardized Evaluation

As explained earlier, many different test datasets and evaluation metrics are used by practitioners who select evaluation procedures based on their own needs and use cases. Such evaluation scenarios might differ between domains (e.g., automotive, avionics, naval), making it difficult to compare different safety monitoring approaches. We believe that the development of a unified benchmarking framework, including different autonomous system use cases and evaluation metrics, is a promising research direction. Indeed, it will foster the development of safety monitoring approaches that will help certify safety-critical systems that rely on ML models.

8.3.3 | Certification

With the increasing use of ML approaches, solutions such as safety monitors will be an important tool to certify future critical autonomous systems. Hence, the community needs to start addressing the challenges mentioned above. As a first step, we believe that building unified evaluation benchmarks and metrics, reflecting the different aspects highlighted in this survey would greatly help to develop SM better suited to the safety-critical context. To go further and be able to certify perception components and their safety monitors, one needs to establish the impact of perception errors on the safety of the entire system, which is an unsolved problem.

Acknowledgments

This research has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 812.788 (MSCA-ETN SAS). This publication reflects only the authors' view, exempting the European Union from any liability. Project website: http://etn-sas.eu/.

This research has also benefited from the AI Interdisciplinary Institute ANITI. ANITI is funded by the French "Investing for the Future — PIA3" program under the Grant agreement No ANR-19-PI3A-0004.

Disclosure

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The authors have nothing to report.

Endnotes

[1] https://paperswithcode.com/sota.

References

1. M. G. Calvi, *Runtime Monitoring of Cyber-Physical Systems Using Data-Driven Models* (2019).

2. T. Haidegger, "Autonomy for Surgical Robots: Concepts and Paradigms," *IEEE Transactions on Medical Robotics and Bionics* 1, no. 2 (2019): 65–76.

3. J. Guérin, K. Delmas, and J. Guiochet, "Certifying Emergency Landing for Safe Urban UAV," in *7th International Workshop on Safety and Security of Intelligent Vehicles (SSIV 2021) at IEEE/IFIP Intern. Conf. On Dependable Systems and Networks (DSN)* (IEEE, 2021), 55–62.

4. C. Premebida, R. Ambrus, and Z.-C. Marton, *Intelligent Robotic Perception Systems. In: Hurtado Efren Gorrostieta, Ed* (Applications of Mobile RobotsRijeka, 2018).

5. Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking Deep Reinforcement Learning for Continuous Control," in *International Conference on Machine Learning* (PMLR, 2016), 1329–1338.

6. A. Brunetti, D. Buongiorno, G. F. Trotta, and V. Bevilacqua, "Computer Vision and Deep Learning Techniques for Pedestrian Detection and Tracking: A Survey," *Neurocomputing* 300 (2018): 17–33.

7. J. Guerin, K. Delmas, and J. Guiochet, "Evaluation of Runtime Monitoring for UAV Emergency Landing," in *2022 International Conference on Robotics and Automation (ICRA)* (IEEE, 2022), 9703–9709.

8. K. Varshney and H. Alemzadeh, "On the Safety of Machine Learning: Cyber-Physical Systems, Decision Sciences, and Data Products," *Big Data* 5, no. 3 (2017): 246–255.

9. J. M. Faria, "Machine Learning Safety: An Overview," in *Proceedings of the 26th Safety-Critical Systems Symposium, York, UK* (SCSC, 2018), 6–8.

10. S. Mohseni, M. Pitale, V. Singh, and Z. Wang, "Practical Solutions for Machine Learning Safety in Autonomous Vehicles," in *Proceedings of the Workshop on Artificial Intelligence Safety* (CEUR, 2020), 162–169.

11. Y. Gal and Z. Ghahramani, "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning," in *International Conference on Machine Learning (ICML), new York, United States* (PMLR, 2016), 1050–1059.

12. A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic Concepts and Taxonomy of Dependable and Secure Computing," *IEEE Transactions on Dependable and Secure Computing* 1, no. 1 (2004): 11–33.

13. M. Machin, J. Guiochet, H. Waeselynck, J.-P. Blanquart, M. Roy, and L. Masson, "SMOF: A Safety Monitoring Framework for Autonomous Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, no. 5 (2018): 702–715.

14. J. Rushby, "Kernels for Safety," *Safe and Secure Computing Systems* 13 (1989): 210–220.

15. C. Pace and D. Seward, "A Safety Integrated Architecture for an Autonomous Safety Excavator," in *International Symposium on Automation and Robotics in Construction* (IAARC, 2000).

16. S. Roderick, B. Roberts, E. Atkins, and D. Akin, "The Ranger Robotic Satellite Servicer and Its Autonomous Software-Based Safety System," *IEEE Intelligent Systems* 19, no. 5 (2004): 12–19.

17. F. Py and F. Ingrand, "Dependable Execution Control for Autonomous Robots," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No. 04CH37566)* (IEEE, 2004), 1136–1141.

18. J. Fox and S. Das, "Safe and Sound," *Artificial Intelligence in Hazardous Applications* 307 (2000).

19. P. Klein, "The Safety-Bag Expert System in the Electronic Railway Interlocking System Elektra," in *Operational Expert System Applications in Europe* (Elsevier, 1991), 1–15.

20. S. Haddadin, M. Suppa, S. Fuchs, T. Bodenmüller, A. Albu-Schäffer, and G. Hirzinger, "Towards the Robotic co-Worker," in *Robotics Research* (Springer, 2011), 261–282.

21. L. Masson, *Safety Monitoring for Autonomous Systems: Interactive Elicitation of Safety Rules* (PhD Thesis, 2019).

22. Z. Xu and J. H. Saleh, "Machine Learning for Reliability Engineering and Safety Applications: Review of Current Status and Future Opportunities," *Reliability Engineering and System Safety* 211 (2021): 107530.

23. K. Muhammad, A. Ullah, J. Lloret, J. Del Ser, and V. Albuquerque, "Deep Learning for Safe Autonomous Driving: Current Challenges and Future Directions," *IEEE Transactions on Intelligent Transportation Systems* 22, no. 7 (2020): 4316–4336.

24. L. Hou, H. Chen, G. Zhang, and X. Wang, "Deep Learning-Based Applications for Safety Management in the AEC Industry: A Review," *Applied Sciences* 11, no. 2 (2021): 821.

25. A. Pandharipande, C.-H. Cheng, J. Dauwels, et al., "Sensing and Machine Learning for Automotive Perception: A Review," *IEEE Sensors Journal* 23, no. 11 (2023): 11097–11115.

26. S. Mohseni, H. Wang, Y. Xiao Chaowei, W. Z. Zhiding, and J. Yadawa, "Taxonomy of Machine Learning Safety: A Survey and Primer," *ACM Computing Surveys* 55, no. 8 (2022): 1–38.

27. Q. M. Rahman, P. Corke, and F. Dayoub, "Run-Time Monitoring of Machine Learning for Robotic Perception: A Survey of Emerging Trends," *IEEE Access* 9 (2021): 20067–20075.

28. X.-Y. Zhang, G.-S. Xie, X. Li, T. Mei, and C.-L. Liu, "A Survey on Learning to Reject," *Proceedings of the IEEE* 111, no. 2 (2023): 185–215.

29. J. Guerin, K. Delmas, R. S. Ferreira, and J. Guiochet, "Out-Of-Distribution Detection Is Not all You Need," in *The 37th AAAI Conference on Artificial Intelligence (2023)* (AAAI, 2023).

30. S. Alasadi and W. S. Bhaya, "Review of Data Preprocessing Techniques in Data Mining," *Journal of Engineering and Applied Sciences* 12, no. 16 (2017): 4102–4107.

31. A. J. Bekker and J. Goldberger, "Training Deep Neural-Networks Based on Unreliable Labels," in *IEEE International Conference on Acoustics, Speech and Signal Processing, China* (IEEE, 2016), 2682–2686.

32. E. Breck, S. Cai, E. Nielsen, M. Salib, and D. Sculley, "The ml Test Score: A Rubric for ml Production Readiness and Technical Debt Reduction," in *2017 IEEE International Conference on Big Data (Big Data)* (IEEE, 2017), 1123–1132.

33. D. Sculley, G. Holt, D. Golovin, et al., "Hidden Technical Debt in Machine Learning Systems," in *Advances in Neural Information Processing Systems (NeurIPS), Montreal, Canada* (NeurIPS, 2015), 2503–2511.

34. M. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A Review of Novelty Detection," *Signal Processing* 99 (2014): 215–249.

35. A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial Attacks and Defences: A Survey," 2018 arXiv preprint arXiv:1810.00069.

36. F. Granese, M. Romanelli, D. Gorla, C. Palamidessi, and P. Piantanida, "DOCTOR: A Simple Method for Detecting Misclassification Errors," *Advances in Neural Information Processing Systems* 34 (2021): 5669–5681.

37. R. S. Ferreira, J. Arlat, J. Guiochet, and H. Waeselynck, "Benchmarking Safety Monitors for Image Classifiers With Machine Learning," in *2021 IEEE 26th Pacific Rim International Symposium on Dependable Computing (PRDC)* (IEEE, 2021), 7–16.

38. Z. Shen, J. Liu, Y. He, et al., "Towards Out-of-Distribution Generalization: A Survey," (2021), arXiv preprint arXiv:2108.13624.

39. R. S. Ferreira, J. Guérin, J. Guiochet, and H. Waeselynck, "SiMOOD: Evolutionary Testing Simulation With Out-of-Distribution Images," in *2022 IEEE 27th Pacific Rim International Symposium on Dependable Computing (PRDC)* (IEEE, 2022), 68–77.

40. M. Cordts, M. Omran, S. Ramos, et al., "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016).

41. Y. Yuan, X. Chen, X. Chen, and J. Wang, "Segmentation Transformer: Object-Contextual Representations for Semantic Segmentation," in *European Conference on Computer Vision (ECCV)* (ECVA, 2021).

42. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2009), 248–255.

43. T.-Y. Lin, M. Maire, S. Belongie, et al., "Microsoft Coco: Common Objects in Context," in *European Conference on Computer Vision* (Springer, 2014), 740–755.

44. Z. Liu, H. Hu, Y. Lin, et al., "Swin Transformer V2: Scaling Up Capacity and Resolution," 2021 arXiv Preprint arXiv:2111.09883.

45. S. Shafaei, S. Kugele, M. H. Osman, and A. Knoll, "Uncertainty in Machine Learning: A Safety Perspective on Autonomous Driving," in *International Conference on Computer Safety, Reliability, and Security (SAFECOMP)* (Springer, 2018), 458–464.

46. J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized Out-of-Distribution Detection: A Survey," 2021 Preprint arXiv:2110.11334.

47. H. Blum, P.-E. Sarlin, J. Nieto, R. Siegwart, and C. Cadena, "Fishyscapes: A Benchmark for Safe Semantic Segmentation in Autonomous Driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (IEEE/CVF, 2019).

48. F. Condessa, J. Bioucas-Dias, and J. Kovačević, "Performance Measures for Classification Systems With Rejection," *Pattern Recognition* 63 (2017): 437–450.

49. R. S. Ferreira, G. Zimbrão, and L. Alvim, "AMANDA: Semi-Supervised Density-Based Adaptive Model for Non-Stationary Data With Extreme Verification Latency," *Information Sciences* 488 (2019): 219–237.

50. D. Hendrycks and T. Dietterich, "Benchmarking Neural Network Robustness to Common Corruptions and Perturbations," in *International Conference on Learning Representations* (2019).

51. E. Khalastchi, M. Kalech, and L. Rokach, "Sensor Fault Detection and Diagnosis for Autonomous Systems," in *Proceedings of the 2013 International Conference on Autonomous Agents and Multi-Agent Systems* (Citeseer, 2013), 15–22.

52. C. Micheloni and G. L. Foresti, "Active Tuning of Intrinsic Camera Parameters," *IEEE Transactions on Automation Science and Engineering* 6, no. 4 (2009): 577–587.

53. S. Surya and R. Ravi, "Deployment of Backup Sensors in Wireless Sensor Networks for Structural Health Monitoring," in *2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI)* (IEEE, 2018), 1526–1533.

54. A. Sinha, S. Chand, V. Vu, H. Chen, and V. Dixit, "Crash and Disengagement Data of Autonomous Vehicles on Public Roads in California," *Scientific Data* 8, no. 1 (2021): 1–10.

55. M. C. Motwani, M. C. Gadiya, R. C. Motwani, and F. C. Harris, "Survey of Image Denoising Techniques," in *Proceedings of GSPX* (Proceedings of GSPX, 2004), 27–30.

56. C. Tian, L. Fei, W. Zheng, Y. Xu, W. Zuo, and C.-W. Lin, "Deep Learning on Image Denoising: An Overview," *Neural Networks* 131 (2020): 251–275.

57. A. Rasouli, I. Kotseruba, and J. K. Tsotsos, "It's Not all About Size: On the Role of Data Properties in Pedestrian Detection," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops* (ECVA, 2018).

58. R. Stumpf, "Autopilot Blamed for Tesla's Crash Into Overturned Truck," 2020.

59. J. Zhang, L. Lin, J. Zhu, et al., "Attribute-Aware Pedestrian Detection in a Crowd," *IEEE Transactions on Multimedia* 23 (2020): 3085–3097.

60. N. Akhtar and A. Mian, "Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey," *IEEE Access* 6 (2018): 14410–14430.

61. A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial Examples in the Physical World," in *Artificial Intelligence Safety and Security* (Chapman and Hall/CRC, 2018), 99–112.

62. A. Boloor, K. Garimella, X. He, C. Gill, Y. Vorobeychik, and X. Zhang, "Attacking Vision-Based Perception in End-To-End Autonomous Driving Models," *Journal of Systems Architecture* 110 (2020): 101766.

63. J. Guiochet, D. Martin-Guillerez, and D. Powell, "Experience With Model-Based User-Centered Risk Assessment for Service Robots," in *2010 IEEE 12th International Symposium on High Assurance Systems Engineering* (IEEE, 2010), 104–113.

64. T. Dreossi, A. Donzé, and S. A. Seshia, "Compositional Falsification of Cyber-Physical Systems With Machine Learning Components," *Journal of Automated Reasoning* 63, no. 4 (2019): 1031–1053.

65. T. Dreossi, D. J. Fremont, S. Ghosh, et al., "Verifai: A Toolkit for the Formal Design and Analysis of Artificial Intelligence-Based Systems," in *International Conference on Computer Aided Verification* (Springer, 2019), 432–442.

66. R. Salay, M. Angus, and K. Czarnecki, "A Safety Analysis Method for Perceptual Components in Automated Driving," in *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)* (IEEE, 2019), 24–34.

67. P.-Y. Huang, W.-T. Hsu, C.-Y. Chiu, T.-F. Wu, and M. Sun, "Efficient Uncertainty Estimation for Semantic Segmentation in Videos," in *Proceedings of the European Conference on Computer Vision (ECCV)* (ECVA, 2018), 520–535.

68. C. Hubschneider, R. Hutmacher, and J. M. Zöllner, "Calibrating Uncertainty Models for Steering Angle Estimation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)* (IEEE, 2019), 1511–1518.

69. G. Costante and M. Mancini, "Uncertainty Estimation for Data-Driven Visual Odometry," *IEEE Transactions on Robotics* 36, no. 6 (2020): 1738–1757.

70. G. Chhablani, A. Sharma, H. Pandey, and T. Dash, "Superpixel-Based Domain-Knowledge Infusion in Computer Vision," 2021 arXiv Preprint arXiv:2105.09448.

71. V. Ramanathan, C. Li, J. Deng, et al., "Learning Semantic Relationships for Better Action Retrieval in Images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2015), 1100–1109.

72. J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Broeck, "A Semantic Loss Function for Deep Learning With Symbolic Knowledge," in *International Conference on Machine Learning* (PMLR, 2018), 5502–5511.

73. I. Donadello, L. Serafini, and A. S. Garcez, "Logic Tensor Networks for Semantic Image Interpretation," in *IJCAI International Joint Conference on Artificial Intelligence* (IJCAI, 2017), 1596–1602.

74. C. Cortes, G. DeSalvo, and M. Mohri, "Boosting With Abstention," *Advances in Neural Information Processing Systems* 29 (2016): 1660–1668.

75. Y. Geifman and R. El-Yaniv, "Selectivenet: A Deep Neural Network With an Integrated Reject Option," in *International Conference on Machine Learning* (PMLR, 2019), 2151–2159.

76. W. Zhou, J. S. Berrio, S. Worrall, and E. Nebot, "Automated Evaluation of Semantic Segmentation Robustness for Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems* 21, no. 5 (2019): 1951–1963.

77. M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson, "Failing to Learn: Autonomously Identifying Perception Failures for Self-Driving Cars," *IEEE Robotics and Automation Letters* 3, no. 4 (2018): 3860–3867.

78. D. Hendrycks and K. Gimpel, "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks," 2016 arXiv Preprint arXiv:1610.02136.

79. S. Liang, Y. Li, and R. Srikant, "Enhancing the Reliability of Out-Of-Distribution Image Detection in Neural Networks," in *International Conference on Learning Representations* (2018).

80. Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, "Generalized Odin: Detecting out-Of-Distribution Image Without Learning From Out-of-Distribution Data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE/CVF, 2020), 10951–10960.

81. D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, "Model Assertions for Debugging Machine Learning," in *NeurIPS MLSys Workshop*, vol. 10 (NeurIPS, 2018).

82. C. Harper, G. Chance, A. Ghobrial, S. Alam, T. Pipe, and K. Eder, "Safety Validation of Autonomous Vehicles Using Assertion-Based Oracles," 2021 arXiv Preprint arXiv:2111.04611.

83. Y. Chen, C.-H. Cheng, J. Yan, and R. Yan, "Monitoring Object Detection Abnormalities via Data-Label and Post-Algorithm Abstractions," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2021), 6688–6693.

84. J. Guérin, A. M. de Paula Canuto, and L. M. G. Goncalves, "Robust Detection of Objects Under Periodic Motion With Gaussian Process Filtering," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)* (IEEE, 2020), 685–692.

85. S. W. Yahaya, A. Lotfi, and M. Mahmud, "A Consensus Novelty Detection Ensemble Approach for Anomaly Detection in Activities of Daily Living," *Applied Soft Computing* 83 (2019): 105613.

86. A. Roitberg, Z. Al-Halah, and R. Stiefelhagen, "Informed Democracy: Voting-Based Novelty Detection for Action Recognition," in *British Machine Vision Conference* (BMVA, 2018).

87. Y. Kantaros, T. Carpenter, S. Park, et al., "VisionGuard: Runtime Detection of Adversarial Inputs to Perception Systems," 2020 arXiv Preprint arXiv:2002.09792.

88. J. Wang, G. Dong, J. Sun, X. Wang, and P. Zhang, "Adversarial Sample Detection for Deep Neural Network Through Model Mutation Testing," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (IEEE, 2019), 1245–1256.

89. J. Liu, L. Chen, A. Mine, and J. Wang, "Input Validation for Neural Networks via Runtime Local Robustness Verification," 2020 arXiv Preprint arXiv:2002.03339.

90. Q. M. Rahman, N. Sünderhauf, and F. Dayoub, "Did You Miss the Sign? A False Negative Alarm System for Traffic Sign Detectors," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), 3748–3753.

91. Y. Sun, C. Guo, and Y. Li, "ReAct: Out-Of-Distribution Detection With Rectified Activations," in *Advances in Neural Information Processing Systems*, ed. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. S. Liang, and V. J. Wortman (Curran Associates, Inc., 2021), 144–157.

92. A. Lukina, C. Schilling, and T. A. Henzinger, "Into the Unknown: Active Monitoring of Neural Networks," in *International Conference on Runtime Verification* (Springer, 2021), 42–61.

93. C.-H. Cheng, G. Nührenberg, and H. Yasuoka, "Runtime Monitoring Neuron Activation Patterns," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy* (IEEE, 2019), 300–303.

94. T. A. Henzinger, A. Lukina, and C. Schilling, "Outside the Box: Abstraction-Based Monitoring of Neural Networks," in *24th European Conference on Artificial Intelligence-ECAI 2020* (IOS press, 2020), 2433–2440.

95. R. S. Ferreira, J. Guerin, J. Guiochet, and H. Waeselynck, "SENA: Similarity-Based Error-Checking of Neural Activations," in *26th European Conference on Artificial Intelligence-ECAI 2023* (IOS press, 2023).

96. C. Wu, Y. Falcone, and S. Bensalem, "Customizable Reference Runtime Monitoring of Neural Networks Using Resolution Boxes," 2021 arXiv Preprint arXiv:2104.14435.

97. H. Wang, J. Xu, C. Xu, X. Ma, and J. Lu, "Dissector: Input Validation for Deep Learning Applications by Crossing-Layer Dissection," in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)* (IEEE, 2020), 727–738.

98. C. Schorn and L. Gauerhof, "FACER: A Universal Framework for Detecting Anomalous Operation of Deep Neural Networks," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)* (IEEE, 2020), 1–6.

99. K. Lee, K. Lee, H. Lee, and J. Shin, "A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks," *Advances in Neural Information Processing Systems* 31 (2018): 11060–11066.

100. V. Chen, M.-K. Yoon, and Z. Shao, "Task-Aware Novelty Detection for Visual-Based Deep Learning in Autonomous Systems," in *2020 IEEE International Conference on Robotics and Automation (ICRA)* (IEEE, 2020), 11060–11066.

101. J. Ndong and K. Salamatian, "Signal Processing-Based Anomaly Detection Techniques: A Comparative Analysis," in *Proc. 2011 3rd International Conference on Evolving Internet* (IARIA, 2011), 32–39.

102. S. S. Kim and A. L. N. Reddy, "Image-Based Anomaly Detection Technique: Algorithm, Implementation and Effectiveness," *IEEE Journal on Selected Areas in Communications* 24, no. 10 (2006): 1942–1954.

103. H. Liao, D. Wang, C. Yang, and J. Shine, "Video-Based Water Drop Detection and Removal Method for a Moving Vehicle," *Information Technology Journal* 12, no. 4 (2013): 569–583.

104. M. Sabokrou, M. Khalooei, M. Fathy, and E. Adeli, "Adversarially Learned One-Class Classifier for Novelty Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), 3379–3388.

105. T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar, "Improving Reconstruction Autoencoder Out-of-Distribution Detection With Mahalanobis Distance," (2018), arXiv Preprint arXiv:1812.02765.

106. A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour Prediction for Autonomous Driving Systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering* (ACM/IEEE, 2020), 359–371.

107. F. Cai and X. Koutsoukos, "Real-Time out-Of-Distribution Detection in Learning-Enabled Cyber-Physical Systems," in *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)* (IEEE, 2020), 174–183.

108. A. Loquercio, M. Segu, and D. Scaramuzza, "A General Framework for Uncertainty Estimation in Deep Learning," *IEEE Robotics and Automation Letters* 5, no. 2 (2020): 3153–3160.

109. D. Cofer, I. Amundson, R. Sattigeri, et al., "Run-Time Assurance for Learning-Based Aircraft Taxiing," in *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)* (IEEE, 2020), 1–9.

110. C. Buerkle, F. Geissler, M. Paulitsch, and K.-U. Scholl, "Fault-Tolerant Perception for Automated Driving A Lightweight Monitoring Approach," 2021 arXiv Preprint arXiv:2111.12360.

111. J. Mena, O. Pujol, and J. Vitrià, "A Survey on Uncertainty Estimation in Deep Learning Classification Systems From a Bayesian Perspective," *ACM Computing Surveys* 54, no. 9 (2021): 1–35.

112. J. Gawlikowski, C. R. N. Tassi, M. Ali, et al., "A Survey of Uncertainty in Deep Neural Networks," 2021 arXiv Preprint arXiv:2107.03342.

113. D. Ulmer, "A Survey on Evidential Deep Learning for Single-Pass Uncertainty Estimation," 2021 arXiv Preprint arXiv:2110.03051.

114. M. Sensoy, L. Kaplan, and M. Kandemir, "Evidential Deep Learning to Quantify Classification Uncertainty," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (NeurIPS, 2018), 3183–3193.

115. M. Seeger, *Bayesian Modelling in Machine Learning: A Tutorial Review* (EPFL, 2006).

116. G. E. P. Box and G. C. Tiao, *Bayesian Inference in Statistical Analysis* (John Wiley & Sons, 2011).

117. W. Chen, Y. Shen, H. Jin, and W. Wang, "A Variational Dirichlet Framework for Out-of-Distribution Detection," 2018 arXiv Preprint arXiv:1811.07308.

118. D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone, "Dirichlet-Based Gaussian Processes for Large-Scale Calibrated Classification," *Advances in Neural Information Processing Systems* 31 (2018).

119. A. Malinin and M. Gales, "Predictive Uncertainty Estimation via Prior Networks," *Advances in Neural Information Processing Systems* 31 (2018).

120. S. Rossi, P. Michiardi, and M. Filippone, "Good Initializations of Variational Bayes for Deep Models," in *International Conference on Machine Learning* (PMLR, 2019), 5487–5497.

121. M. Welling and Y. W. Teh, "Bayesian Learning via Stochastic Gradient Langevin Dynamics," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (Citeseer, 2011), 681–688.

122. E. Goan and C. Fookes, "Bayesian Neural Networks: An Introduction and Survey," in *Case Studies in Applied Bayesian Data Science* (Springer, 2020), 45–87.

123. D. Feng, A. Harakeh, S. Waslander, and K. Dietmayer, "A Review and Comparative Study on Probabilistic Object Detection in Autonomous Driving," *IEEE Transactions on Intelligent Transportation Systems* 23, no. 8 (2021): 9961–9980.

124. J. Mukhoti and Y. Gal, "Evaluating Bayesian Deep Learning Methods for Semantic Segmentation," 2018 arXiv Preprint arXiv:1811.12709.

125. Y. Gal, J. Hron, and A. Kendall, "Concrete Dropout," *Advances in Neural Information Processing Systems* 30 (2017).

126. T. Dash, S. Chitlangia, A. Ahuja, and A. Srinivasan, "Incorporating Domain Knowledge Into Deep Neural Networks," 2021 arXiv Preprint arXiv:2103.00180.

127. M. Marszalek and C. Schmid, "Semantic Hierarchies for Visual Object Recognition," in *2007 IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2007), 1–7.

128. G. Fumera and F. Roli, "Support Vector Machines With Embedded Reject Option," in *International Workshop on Support Vector Machines* (Springer, 2002), 68–82.

129. M. E. Hellman, "The Nearest Neighbor Classification Rule With a Reject Option," *IEEE Transactions on Systems Science and Cybernetics* 6, no. 3 (1970): 179–185.

130. K. Hendrickx, L. Perini, D. Plas, W. Meert, and J. Davis, "Machine Learning With a Reject Option: A Survey," 2021 arXiv Preprint arXiv:2107.11277.

131. F. Cai, J. Li, and X. Koutsoukos, "Detecting Adversarial Examples in Learning-Enabled Cyber-Physical Systems Using Variational Autoencoder for Regression," in *2020 IEEE Security and Privacy Workshops (SPW)* (IEEE, 2020), 208–214.

132. A. Stocco and P. Tonella, "Towards Anomaly Detectors That Learn Continuously," in *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)* (IEEE, 2020), 201–208.

133. Y. Feng and A. Easwaran, "Robust Out-of-Distribution Motion Detection and Localization in Autonomous CPS: Wip Abstract," in *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems* (ACM/IEEE, 2021), 225–226.

134. J. Guérin, S. Thiery, E. Nyiri, O. Gibaru, and B. Boots, "Combining Pretrained CNN Feature Extractors to Enhance Clustering of Complex Natural Images," *Neurocomputing* 423 (2021): 551–571.

135. C. Gurău, D. Rao, C. H. Tong, and I. Posner, "Learn From Experience: Probabilistic Prediction of Perception Performance to Avoid Failure," *International Journal of Robotics Research* 37, no. 9 (2018): 981–995.

136. C. B. Kuhn, M. Hofbauer, G. Petrovic, and E. Steinbach, "Introspective Black Box Failure Prediction for Autonomous Driving," in *2020 IEEE Intelligent Vehicles Symposium (IV)* (IEEE, 2020), 1907–1913.

137. R. Gontijo-Lopes, Y. Dauphin, and E. D. Cubuk, "No One Representation to Rule Them All: Overlapping Features of Training Methods," 2021 arXiv Preprint arXiv:2110.12899.

138. X. Wang, L. Xie, C. Dong, and Y. Shan, "Real-Esrgan: Training Real-World Blind Super-Resolution With Pure Synthetic Data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE/CVF, 2021), 1905–1914.

139. H. Wang, Z. Li, L. Feng, and W. Zhang, "ViM: Out-Of-Distribution With Virtual-Logit Matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE/CVF, 2022), 4921–4930.

140. W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-Based Out-of-Distribution Detection," *Advances in Neural Information Processing Systems* 33 (2020): 21464–21475.

141. S. Rabiee and J. Biswas, "IVOA: Introspective Vision for Obstacle Avoidance," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (IEEE, 2019), 1230–1235.

142. S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective* (Academic press, 2015).

143. A. Roy, A. Cobb, D. Bastian Nathaniel, B. Jalaian, and S. Jha, "Runtime Monitoring of Deep Neural Networks Using Top-Down Context Models Inspired by Predictive Processing and Dual Process Theory," in *AAAI 2022 Workshop on Designing Artificial Intelligence for Open Worlds* (AAAI, 2022).

144. M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)* (IEEE, 2018), 132–142.

145. B. Li, X. Xiao, Y. Zhang, H. Li, and H. Wang, "Camera-IMU Extrinsic Calibration Quality Monitoring for Autonomous Ground Vehicles," *IEEE Robotics and Automation Letters* 7, no. 2 (2022): 4614–4621.

146. D. Phan, N. Paoletti, R. Grosu, N. Jansen, S. A. Smolka, and S. D. Stoller, "Neural Simplex Architecture," in *NASA Formal Methods: 12th International Symposium* (Springer, 2020), 97–114.

147. L. Gondara, "Medical Image Denoising Using Convolutional Denoising Autoencoders," in *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)* (IEEE, 2016), 241–246.

148. K. H. Abdulkareem, N. Arbaiy, A. A. Zaidan, et al., "A New Standardisation and Selection Framework for Real-Time Image Dehazing Algorithms From Multi-Foggy Scenes Based on Fuzzy Delphi and Hybrid Multi-Criteria Decision Analysis Methods," *Neural Computing and Applications* 33 (2021): 1029–1054.

149. Q. Yan, D. Gong, J. Q. Shi, et al., "High Dynamic Range Imaging via Gradient-Aware Context Aggregation Network," *Pattern Recognition* 122 (2022): 108342.

150. Q. Yan, D. Gong, J. Q. Shi, et al., "Dual-Attention-Guided Network for Ghost-Free High Dynamic Range Imaging," *International Journal of Computer Vision* (2021): 1–19.

151. Z. Liu, W. Lin, X. Li, et al., "ADNet: Attention-Guided Deformable Convolutional Network for High Dynamic Range Imaging," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE/CVF, 2021), 463–470.

152. R. Qian, R. T. Tan, W. Yang, J. Su, and J. Liu, "Attentive Generative Adversarial Network for Raindrop Removal From a Single Image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2018), 2482–2491.

153. Y. Zhou, J. Jiao, H. Huang, et al., "When Awgn-Based Denoiser Meets Real Noises," in *Proceedings of the AAAI Conference on Artificial Intelligence* (AAAI, 2020), 13074–13081.

154. C. Zhang and P. Gao, "Countering Adversarial Examples: Combining Input Transformation and Noisy Training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision* (IEEE/CVF, 2021), 102–111.

155. Z. Wang, J. Chen, and S. Hoi, "Deep Learning for Image Super-Resolution: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, no. 10 (2020): 3365–3387.

156. W. Shi, J. Caballero, F. Huszár, et al., "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), 1874–1883.

157. C. Ledig, L. Theis, F. Huszár, et al., "Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2017), 4681–4690.

158. R. Marinescu, D. Moyer, and P. Golland, "Bayesian Image Reconstruction Using Deep Generative Models," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications* (NeurIPS, 2021).

159. C. Saharia, W. Chan, H. Chang, et al., "Palette: Image-To-Image Diffusion Models," in *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications* (NeurIPS, 2021).

160. Z. Ni, K.-K. Ma, H. Zeng, and B. Zhong, "Color Image Demosaicing Using Progressive Collaborative Representation," *IEEE Transactions on Image Processing* 29 (2020): 4952–4964.

161. W. Zhou, L. Zhang, S. Gao, and X. Lou, "Gradient-Based Feature Extraction From Raw Bayer Pattern Images," *IEEE Transactions on Image Processing* 30 (2021): 5122–5137.

162. J. Al-Afandi and A. Horváth, "Class Retrieval of Adversarial Attacks," in *Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV)* (IEEE/CVF, 2021).

163. C. Y. Li, R. Sánchez-Matilla, A. S. Shamsabadi, R. Mazzon, and A. Cavallaro, "On the Reversibility of Adversarial Attacks," in *2021 IEEE International Conference on Image Processing (ICIP)* (IEEE, 2021), 3073–3077.

164. G. Kakamanshadi, S. Gupta, and S. Singh, "A Survey on Fault Tolerance Techniques in Wireless Sensor Networks," in *2015 International Conference on Green Computing and Internet of Things (ICGCIoT)* (IEEE, 2015), 168–173.

165. K. Andrej, "Tesla Autonomous Driving Talk at CVPR 2021," 2021.

166. G. Ven and A. S. Tolias, "Three Scenarios for Continual Learning," 2019 arXiv Preprint arXiv:1904.07734.

167. Y. Liu, B. Schiele, and Q. Sun, "Adaptive Aggregation Networks for Class-Incremental Learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE/CVF, 2021), 2544–2553.

168. L. Deng, "The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]," *IEEE Signal Processing Magazine* 29, no. 6 (2012): 141–142.

169. A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features From Tiny Images* (University of Toronto, 2009).

170. A. Shafaei, M. Schmidt, and J. J. Little, "A Less Biased Evaluation of Out-of-Distribution Sample Detectors," in *30th British Machine Vision Conference* (BMVA, 2019).

171. A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proceedings of the 1st Annual Conference on Robot Learning* (PMLR, 2017), 1–16.

172. F. U. Haq, D. Shin, S. Nejati, and L. C. Briand, "Comparing Offline and Online Testing of Deep Neural Networks: An Autonomous Car Case Study," in *2020 IEEE 13th International Conference on Software Testing, Validation and Verification (ICST)* (IEEE, 2020), 85–95.

173. J. Guerin, R. S. Ferreira, K. Delmas, and J. Guiochet, "Unifying Evaluation of Machine Learning Safety Monitors," in *2022 IEEE 33rd International Symposium on Software Reliability Engineering (ISSRE)* (IEEE, 2022), 414–422.

174. L. Zandbergen, *Evaluating the Simulation Gap for Training off-Road Self-Driving* (University of Amsterdam, Amsterdam, 2021).

175. M. Borg, C. Englund, K. Wnuk, et al., "Safely Entering the Deep: A Review of Verification and Validation for Machine Learning and a Challenge Elicitation in the Automotive Industry," *Journal of Automotive Software Engineering* 1, no. 1 (2019): 1–19.

176. G. Jahangirova, "Oracle Problem in Software Testing," in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis* (ACM, 2017), 444–447.

177. S. Rabanser, S. Günnemann, and Z. Lipton, "Failing Loudly: An Empirical Study of Methods for Detecting Dataset Shift," *Advances in Neural Information Processing Systems* 32 (2019).

178. R. Myers and Z. Saigol, "Pass-Fail Criteria for Scenario-Based Testing of Automated Driving Systems," 2020 arXiv Preprint arXiv:2005.09417.

179. E. De Gelder and C. O. O. Den, "Procedure for the Safety Assessment of an Autonomous Vehicle Using Real-World Scenarios," 2020 arXiv Preprint arXiv:2012.00643.

180. E. De Gelder, H. Elrofai, A. K. Saberi, J.-P. Paardekooper, C. O. O. Den, and S. B. De, "Risk Quantification for Automated Driving Systems in Real-World Driving Scenarios," *IEEE Access* 9 (2021): 168953–168970.

181. Joint Authorities for Rulemaking of Unmanned Systems (JARUS), "JARUS Guidelines on Specific Operations Risk Assessment (SORA) v2.0.," (2019), Guidelines: EASA.

182. J. Kontos, Á. Vathy-Fogarassy, and B. Kránicz, "Phase Plane-Based Approaches for Event Detection and Plausibility Check of Vehicle Dynamics," in *2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES)* (IEEE, 2021), 31–36.

183. D. Kang, D. Raghavan, P. Bailis, and M. Zaharia, "Model Assertions for Monitoring and Improving ML Models," *Proceedings of Machine Learning and Systems* 2 (2020): 481–496.