THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par : *l'Institut National Polytechnique de Toulouse (INP Toulouse)*

Présentée et soutenue le *09/05/2023* **par :**

Raul SENA FERREIRA

Runtime safety monitoring of ML-based perception functions in autonomous systems

---

JURY

Andrea Bondavalli	Full Professor	Rapporteur
Chokri Mraidha	Directeur de Recherche	Rapporteur
Simon Burton	Full professor	Examinateur
Frédéric Lerasle	Professeur des universités	Examinateur
Joris Guerin	Chargé de recherche	Invité
Jérémie GUIOCHET	Professeur des universités	Directeur de thèse
Hélène WAESELYNCK	Professeur des universités	Co-directeur de thèse

---

École doctorale et spécialité :
EDSYS : Informatique 4200018

Unité de Recherche :
Laboratoire d'Analyse et d'Architecture des Systèmes - Centre National de la Recherche Scientifique

Directeur(s) de Thèse :
Jérémie GUIOCHET et *Hélène WAESELYNCK*

Rapporteurs :
Andrea Bondavalli et *Chokri Mraidha*

Aknowledgements

First I would like to thank my parents *Joaquim Jose (Kim)* and *Lucimary (Meire)*, and my grandparents *Maria do Ceu (Mimi)*, and *Jose de Moura (Zeca)*. They always supported me, and they always were a source of inspiration and motivation, each one for a specific reason, even if we were separated by the Atlantic Ocean during all these years. I would like to thank my beloved *Nicoleta*, who always incentivized me and stood with me, especially during some tough moments in the last few years. Moreover, she gave birth to the new and most important thing in my world, my baby girl *Leya*.

I would like to thank my supervisors *Jeremie Guiochet and Helene Waeselynck* who provided me with valuable research training and directions through this Ph.D., and also the great discussions and moments that we shared together in our department. I would like to thank *Joris Guerin* for his great insights, for our nice discussions, and for his thesis revision which was very important to complete this Ph.D. I also would like to thank professors *Chokri Mraidha and Andrea Bondavalli* for their efforts in reviewing this thesis and providing me with great feedback. Thanks to Professor *Simon Burton* for the thesis examination, and for the deep and insightful discussions; and thanks to Professor *Frederic Lerasle* for the thesis examination and for leading the jury.

A big thank you to all my colleagues and professors from the TSF team from the Laboratory for Analysis and Architecture of Systems, of the French National Center for Scientific Research (LAAS-CNRS), in which we spent a lot of a great and memorable time together. I would also like to thank the institute Fraunhofer for our brief but very valuable collaboration. Thanks to all my colleagues and professors from the Safer and Autonomous Systems (SAS) project for the great exchange and amusing moments that we spent together.

Finally, worth to mention that, 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/.

[1]Mais amor, menos confiança.
[2]Não há bem que sempre dure,
nem mal que não tenha fim.
[1]José de Moura Ferreira,
[2]Maria do Céu Cardoso dos Santos.
My beloved grandparents

Contents

Abstract

Highly-accurate machine learning (ML) image classifiers cannot guarantee that they will not fail at operation. Thus, their deployment in safety-critical applications such as autonomous vehicles is still an open issue. The use of fault tolerance mechanisms such as safety monitors (SM) is a promising direction to keep the system in a safe state despite errors of the ML classifier. As the prediction from the ML is the core information directly impacting safety, many works are focusing on monitoring the ML model itself. This thesis focuses on such approaches by covering all necessary aspects to build, test, and evaluate SM in perception functions built with ML, which can be summarized into the following: 1) An extensive literature review in which we introduce a new taxonomy of the existing literature regarding threat identification, requirements elicitation, detection of failure and reaction, and evaluation. 2) A new baseline framework for benchmarking such SM, covering the entire pipeline, from data generation to evaluation. 3) An evolutionary simulation testing of safety-critical perception systems, which is capable of decreasing, at least 10 times, the amount of time to find a set of hazards in safety-critical scenarios such as autonomous emergency braking system simulation. 4) A new SM approach for safety monitoring of ML classifiers.

Acronyms

ALOOC adversarial learned one-class classification. 31, 33–37

CFMEA Classification Failure Mode Effects Analysis. 10

CNN convolutional neural networks. 45

DNN deep neural networks. 4, 16, 25, 27, 33, 38

EBS emergency breaking system. 9

FGSM fast gradient sign method. 33

GA genetic algorithm. 40, 42

GAN generative adversarial network. 16, 40

GPS global positioning system. 17

GTSRB German traffic signal recognition. 32

ID In-Distribution. 6

IMU inertial measurement unit. 17

ML Machine Learning. ix, 1, 2, 4, 5, 9–17

OOB outside-of-the-box. 31, 33–37

OOD Out-Of-Distribution. 6, 15, 24, 27–30

PCA principal component analysis. 32

ReLU rectified linear unit. 14

SM Safety Monitor. 1, 2, 4–8, 10, 16

SUT system under test. xi, 26–28, 30, 34–36, 38

UAV unmanned aerial vehicle. 20, 22

List of Figures

List of Tables

Chapter 1

Introduction

A robotic system is autonomous when it can operate in a real-world environment for an extended period of time without being controlled by humans [Bekey, 2005]. Thanks to recent advances in Machine Learning (ML), autonomous systems have started to leave the safe environment of research laboratories to perform complex tasks, where failures can have catastrophic consequences. Examples of such safety-critical autonomous systems include self-driving cars [Calvi, 2019], surgical robots [Haidegger, 2019], and unmanned aerial vehicles in urban environments [Guérin et al., 2021a], among others. These autonomous systems frequently use large ML models, such as deep neural networks, to either interpret complex sensor signals (perception [Premebida et al., 2018]) or to make decisions based on these signals (control [Duan et al., 2016]). This work focuses on the former and discusses mechanisms to ensure the safety of autonomous systems that use ML-based perception components.

For many autonomous systems applications, specific essential perception tasks can only be solved by using ML. For example, in highly uncontrolled settings such as self-driving cars, a common way to accurately detect and locate pedestrians is to process complex RGB images using deep neural networks [Brunetti et al., 2018]. Today, this information cannot be obtained from other sources and is crucial to guarantee the safe behavior of the vehicle. However, despite the great success of such perception functions, the use of ML presents new dependability challenges that have been discussed extensively in recent research [Varshney and Alemzadeh, 2017, Faria, 2018, Mohseni et al., 2020]. Some of the recurring safety issues highlighted in these works include:

1. The *lack of well-defined specification*: ML models are learned from examples instead of coded manually. Hence, the boundaries of the operating range are unknown, and one cannot prove formally that specific safety constraints are always verified.

2. The *black-box nature* of the models: traceability and transparency of ML predictions are 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 of neural networks cannot be used as is to detect failures since it is possible for a model to deliver a wrong output with high confidence [Gal and Ghahramani, 2016].

For all these reasons, most offline safety activities (fault prevention, fault removal, and fault forecasting [Avizienis et al., 2004]) are not sufficient to ensure safety and certify such autonomous systems. However, online fault tolerance mechanisms deliver services despite the presence of faults and are a promising alternative to improve safety-critical systems using such perception functions based on ML. This thesis focuses on such approaches, particularly to Safety Monitor (SM), which goal is to keep the system in an acceptable state during operation despite faults or

adverse scenarios. As highlighted by Machin et al. [2018], SM are mentioned in the literature under many different terms, such as safety kernel [Rushby, 1989], safety manager [Pace and Seward, 2000], autonomous safety system [Roderick et al., 2004], checker [Py and Ingrand, 2004], guardian agent [Fox and Das, 2000], safety bag [Klein, 1991], or emergency layer [Haddadin et al., 2011]. All these approaches have been extensively applied in cyber-physical systems, but not to ML-based functions.

One of the main problems to perform such safety monitoring of ML-based perception functions is that classical SM approaches applied during the design and operation cannot guarantee that a DNN decision is safe. The reason is that, for some corner cases, ML outputs wrong decisions that cannot be verified by inspecting the code logic or the sensor values and lead to hazards Dreossi et al. [2019a]. We can deploy rule-based monitors when redundant observation sources are applied and safety properties are correctly expressed. However, monitoring an ML component for vision is particularly complex to find a redundant source of observation. Moreover, it is not apparent to express a safety property at the level of an ML prediction. For these reasons, this thesis focuses on the challenging task of deploying and evaluating SM for critical systems using ML for perception functions, such as autonomous vehicles equipped with deep learning models for critical functions such as road sign classification, pedestrian detection, or road lane segmentation.

Thus, in Chapter 2, we present an extensive literature review on the safety monitoring of perception functions using ML in a safety-critical context. We introduce a new taxonomy of the existing literature to represent the main considerations when designing such monitors: threat identification, requirements elicitation, detection of failure and reaction, and evaluation. We highlight the ongoing challenges linked to these SMs and ideas for future work as well.

An important challenge is to define how to properly assess the efficiency of such SMs in the context of safety-critical applications. Thus, in Chapter 3, we propose a new baseline framework for benchmarking monitors applied to ML image classifiers. Furthermore, the proposed framework covers the entire pipeline, from data generation to evaluation. Our approach measures monitor performance with a broader set of metrics than usually proposed in the literature. Moreover, we benchmark three different monitor approaches in 79 benchmark datasets containing five categories of out-of-distribution (OOD) data for vision-based tasks defined in Chapter 2: class novelty, noise, anomalies, distributional shifts, and adversarial attacks. Our results indicate that these monitors are no more accurate than a random monitor.

To deal with OOD threats, literature usually applies data augmentation techniques or SM such as OOD detectors to increase robustness. Evaluating such solutions using traditional metrics can be misleading since not all OOD data lead to failures in the perception system. Hence, testing a perception system should be more reliable if using images captured by the system at runtime instead of just measuring ML performances on a dataset. However, the amount of time spent to generate diverse test cases during a simulation of perception components can grow quickly since it is a combinatorial optimization problem. Therefore, in Chapter 4, aiming to provide a solution for this challenging task, we present SiMOOD, an evolutionary simulation testing of safety-critical perception systems, which comes integrated into the CARLA simulator. Unlike related works that simulate scenarios that raise failures for control or specific perception problems such as adversarial and novelty, we provide an approach that finds relevant OOD perturbations that can lead to hazards in safety-critical perception systems. Moreover, our approach can decrease, at least 10 times, the amount of time to find a set of hazards in safety-critical scenarios such as autonomous emergency braking system simulation.

Recent works confirmed, theoretically and empirically, that such OOD detectors tend to

perform spurious detections resulting in high rates of false positives/negatives regarding the detection. Recent findings indicate that errors provoked by OOD data are mostly impossible to be detected without prior knowledge [Ye et al., 2021], and any OOD detection algorithm without a model selection module tends to be incomplete [Gulrajani and Lopez-Paz, 2020]. Therefore, in Chapter 5, we introduce SENA, a similarity-based error-checking of neural activations. SENA is an SM for ML-based classifiers and it is divided into two steps: 1) it verifies the similarity between activation function values from an incoming image and a set of activation function values extracted from true positives and false negatives from the ML predictions during the training; 2) it uses a statistical core extraction process [Ferreira et al., 2019] to select a minimal subset of the most representative true positive activation function values to be compared at runtime. The first experiments without any further optimization indicate that this method has competitive performance with state-of-the-art data-based monitors. Finally, in Chapter 6, we conclude with a summary of the contributions with perspectives and thoughts for future work. Here is a list of accepted/under-review articles listed in reverse chronological order:

- RS Ferreira, J Guérin, J Guiochet, H Waeselynck, "SENA: Similarity-based Error-checking of Neural Activations", (under review at ECAI 2023), submitted on 05/2023.
- RS Ferreira, J Guérin, K Delmas, J Guiochet, H Waeselynck, "Safety Monitoring of Machine Learning Perception Functions: a Survey", (under review at the Journal of Computational Intelligence), submitted in 07/2022.
- J Guérin, K Delmas, RS Ferreira, J Guiochet, "OOD detection is not all you need", 37th AAAI Conference on Artificial Intelligence (AAAI 2023).
- RS Ferreira, J Guérin, J Guiochet, H Waeselynck, "SiMOOD: evolutionary testing SiMulation with Out-Of-Distribution images", 27th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2022).
- J Guérin, RS Ferreira, K Delmas, J Guiochet, "Unifying Evaluation of Machine Learning Safety Monitors", The 33rd IEEE International Symposium on Software Reliability Engineering (ISSRE 2022).
- RS Ferreira, J Arlat, J Guiochet, H Waeselynck, "Benchmarking Safety Monitors for Image Classifiers with Machine Learning", 26th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC 2021).
- Raul Sena Ferreira, "Towards safety monitoring of ML-based perception tasks of autonomous systems", The 31st IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW 2020).

Chapter 2

Safety Monitoring of Machine Learning Perception Functions

As shown in Figure 1, an SM for ML perception function, often inspect either the inputs or outputs of an ML model. However, monitoring ML in a safety context is still an open issue. Thus, in this chapter, we propose a survey of all major issues and current research on the safety monitoring of ML-based perception functions.

Relatively little research has been conducted specifically on the safety monitoring of ML-based perception functions. Recently, two surveys focused on the safety of deep learning perception components in autonomous systems. Muhammad et al. [2020] analyzed recent deep learning perception works related to autonomous driving safety (e.g., lane detection, pedestrian detection, collision avoidance). They highlight the current issues and propose several recommendations. They also briefly discuss deep learning research areas related to safety, such as SM. Although the objectives and application context are similar, our survey is about safety monitoring, i.e., additional components to ensure the safety of the system, whereas Muhammad et al. [2020] discusses approaches to enhance safety by improving the robustness of deep neural networks (DNN) in a realistic vehicular environment. On the other hand, Rahman et al. [2021] presented a survey about the safety monitoring of perception functions. They organize existing approaches to detect runtime failures in three categories:

1. Approaches that use past examples of failures to predict future ones.
2. Approaches that detect inconsistencies in the perception outputs.
3. Approaches that are based on uncertainty estimation.

Our work differs from Rahman et al. [2021] by organizing the existing safety monitoring literature in a top-down fashion, i.e., starting from the requirements, then the design, and finally the assessments of SMs. This methodology allows highlighting the specific areas where research is lacking to reach the expected levels of integrity and to discuss some fields of the literature (e.g., recovery and evaluation) that were not addressed in other surveys.

Many recent works from different fields of ML have the potential to be used to build successful SMs. This chapter aims to present such approaches and discuss them in the context of the implementation of SMs. Thus, this chapter is organized based on the main safety considerations when designing SMs:

- What threats are being addressed by safety monitors? (Section 2.1)
- How to derive monitor requirements from safety objectives? (Section 2.2)
- Which detection mechanisms can be used? (Section 2.3)
- Which recovery actions can be used? (Section 2.4)

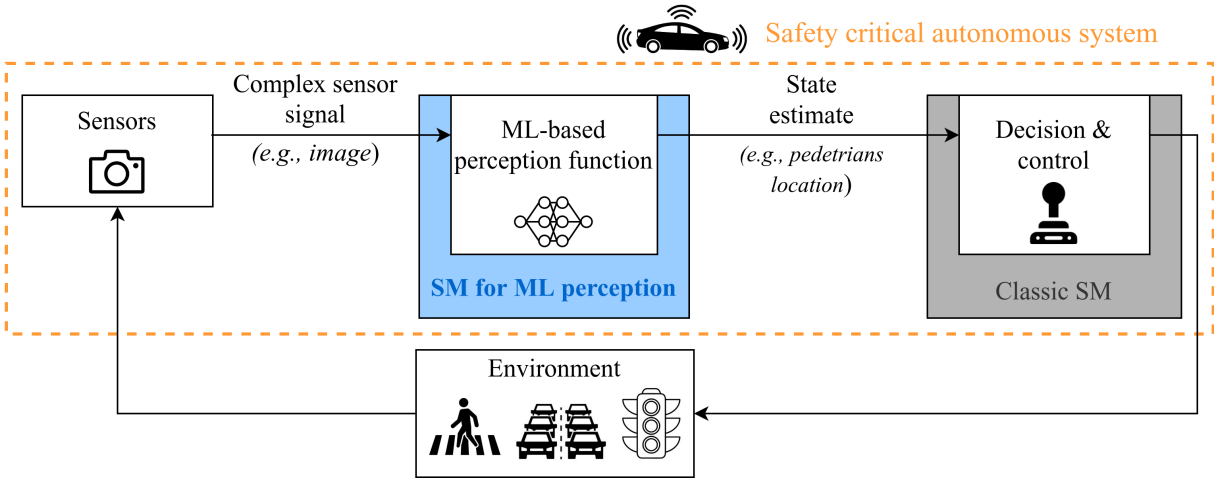


Figure 1: ***Safety Monitors for Machine Learning-based Perception Functions.*** *In modern autonomous systems, state estimation provided by deep learning models cannot be trusted to make safety-critical decisions. Therefore, specific fault tolerance approaches should be implemented to ensure that failures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

- How are safety monitors evaluated? (Section 2.5)

For each of these themes, 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 such autonomous systems, this chapter focuses on the specificities related to ML perception.

2.1 What threats are being addressed by SM?

Understanding the potential threats for a given problem is crucial to design an SM and providing a relevant evaluation protocol. For example, if the safety analysis of a system reveals that a threat is likely to occur, it should be reflected in the evaluations conducted to test the monitor. For example, different kinds of threats can affect the performance of ML-based perception functions. In particular, *offline threats* are introduced during the development phase of the ML model, and *runtime threats* occur during live operations.

Offline threats usually come from data inconsistency and problems in the software engineering process. Some examples of offline threats include inadequate ML pre-processing and/or feature engineering [Alasadi and Bhaya, 2017], label noise [Bekker and Goldberger, 2016], inadequate ML testing [Breck et al., 2017], and bad model maintenance [Sculley et al., 2015]. Although this chapter focuses specifically on approaches to prevent runtime threats, it is crucial to ensure that proper software engineering practices are observed during the development of safety monitors. Therefore, testing and maintenance are crucial steps in developing efficient monitors.

The objective of a SM is to detect when an ML perception model is wrong and to implement corrective actions to ensure that erroneous predictions will not lead to catastrophic events. For example, different kinds of input data can lead to errors from an ML model, and we refer to the event of receiving such input as a runtime threat. This section presents a taxonomy of the runtime threats for perception functions, inspired by the existing literature [Pimentel et al., 2014, Chakraborty et al., 2018, Granese et al., 2021, Shen et al., 2021]. Different kinds of faulty

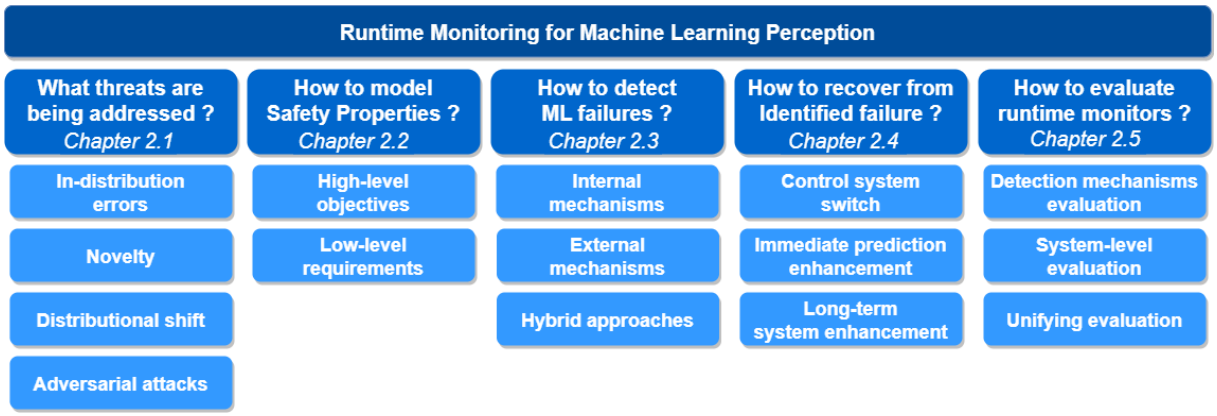


Figure 2: ***Taxonomy of the current research on Safety Monitoring of Machine Learning Perception Functions.*** *Each section of this chapter discusses a specific aspect of safety monitoring of machine learning perception.*

or adverse input data are grouped as threats if they can be detected with similar approaches, or if they can be addressed with similar corrective actions. Therefore, throughout this section, we present such runtime threats and discuss their specificities with respect to detection, reaction, and evaluation.

2.1.1 In-distribution errors

Modern Deep Learning architectures have obtained excellent results in many perception tasks used in autonomous systems. For example, according to the current leader board listed on *papers with code*[1], the best performing model for semantic segmentation on Cityscapes Cordts et al. [2016] has a mean intersection over the union of 84.5% [Yuan et al., 2021], the best model for image classification on ImageNet [Deng et al., 2009] has a top-1 accuracy close to 91%, and the leader for object detection on COCO [Lin et al., 2014a] has a mean average precision around 63% [Liu et al., 2021b]. These results were obtained on the test splits of these datasets, which are assumed to come from the same distribution as the training data, i.e., In-Distribution (ID) data. Although these results are excellent and allowed researchers to build useful applications, the best-performing computer vision models are still not free of errors.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: data incompleteness. Rare conditions tend to be underrepresented since the training data only account for a subset of all real-world possibilities [Shafaei et al., 2018]. New data presenting different characteristics than the training data are usually referred to as Out-Of-Distribution (OOD) data. Yang et al. [2021] presented a taxonomy of the different settings under which modern OOD research is conducted. In this work, we present a different classification better suited to the context of safety monitoring. In particular, we discuss how these threats can be monitored at runtime. We note that there is no unified naming convention for the threats presented hereafter and that they might be found under different names in the literature. We strive to propose clear definitions of the threats discussed in this work to avoid ambiguity.

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

2.1.2 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 [Yang et al., 2021]. For example, Blum et al. [2019] studied the problem of semantic segmentation of a driving scene and trained their model on the Cityscapes dataset [Cordts et al., 2016]. 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 in building classification models with rejection [Condessa et al., 2017], 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 2.3, 2.4, and Section 2.5, respectively.

2.1.3 Distributional shift threats

A distributional shift occurs when the marginal distribution of the runtime input data is different from the training distribution, while the label generation mechanism keeps unchanged [Shen et al., 2021]. Regarding safety monitoring, we distinguish two distinct forms of distributional shifts: covariate shifts and semantic shifts.

2.1.3.1 Covariate shift

A covariate shift is a condition that decreases the ML performance through time in dynamic environments [Ferreira et al., 2019]. 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 Hendrycks and Dietterich [2019]. The deteriorated data can result from either failure in exteroceptive sensors or environmental conditions that were not seen during training.

- Sensor failures Sensor failures represent data perturbations coming from hardware defects. They include various errors such as pixel traps, shifted pixels, Gaussian noise, and Poisson noise. There are specific approaches to identify sensor faults [Khalastchi et al., 2013], which can be addressed by tuning the sensor parameters [Micheloni and Foresti, 2009] or having a backup sensor system [Surya and Ravi, 2018].

- Changes in external conditions For autonomous systems evolving in unstructured environments, the training datasets used for ML perception models cannot cover all possible external conditions encountered in the real world. For example, the visual perception functions of an autonomous vehicle 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 performance of perception components and can reduce the safety of the vehicle [Sinha et al., 2021].

To deal with these two covariate shift types, both traditional signal processing [Motwani et al., 2004] and modern deep learning approaches [Tian et al., 2020] 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.

2.1.3.2 Semantic shift

Semantic shift threats are input data representing semantic objects that either:

- presents different attribute than known members of this category, e.g., pedestrian detector trained during the summer which encounters people wearing winter clothes [Rasouli et al., 2018],
- present a rare interaction between known classes and the environment, e.g., a standard truck that is overturned on the road [Stumpf, 2020].

A semantic shift only includes cases where the categories of the objects present in the data are within the predefined set of categories handled by the model (in contrast to novelty threats). Although it would be valid to consider the semantic shift as a particular case of covariate shift, we believe it presents different attributes concerning safety monitoring. In particular, such distributional shifts cannot be handled by denoising or backup sensors. Instead, some works have focused on detecting specific attributes of the objects which are less likely to change across different environments [Zhang et al., 2020]. Although it was not implemented for ML monitoring, we believe that it has the potential to detect model failures related to semantic shift threats. For example, to monitor a pedestrian detector facing a shift in clothes attributes, one could imagine detecting faces instead of bodies.

2.1.4 Adversarial threats

An adversarial input is an intentional modification of in-distribution data to make ML models fail with high confidence [Akhtar and Mian, 2018, Kurakin et al., 2018]. 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 [Boloor et al., 2020].

Adversarial threats can lead to serious safety issues if applied against the perception functions of critical systems. Therefore, they should be handled by specific SM 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 [Chakraborty et al., 2018].

2.2 How to derive SM 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. [2018] used a HAZOP-UML analysis [Guiochet et al., 2010] 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 breaking system (EBS) implemented in an autonomous vehicle:

- Using *simple sensor signals* such as a laser is not enough to capture the semantic information required for an EBS to distinguish between pedestrians and other moving vehicles. Such semantic information is crucial to EBS performing 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 useful information to 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 sub-regions 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 towards building better specific monitors. In their work, Dreossi et al. [2019a,b] 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. [2019a] 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.

2.3 Which detection mechanisms can be used to build SM?

Before we start discussing modern approaches to handle ML perception functions, we briefly discuss traditional detection mechanisms that have been used to monitor regular autonomous systems. These techniques are generally intended to detect violations of specific safety properties based on a model of the system and its environment. Information from both exteroceptive sensors (e.g., distance sensors) and proprioceptive sensors (e.g., speed) are processed to detect safety threats such as the ability of a vehicle to stop before reaching an obstacle [Ozguner et al., 2007] or to avoid a collision [Al-Khoury, 2017]. In addition, traditional approaches usually try to detect abnormal temporal behavior or assumption violation. They rely on the fact that sensor data can be interpreted in terms of formal specifications and that it can be trusted [Machin et al., 2018]. However, for complex perception functions, traditional SM cannot be expressed directly in terms of raw sensor signals (e.g., image pixels). Hence, new approaches are required to detect errors in the signals provided by the ML models, which is the focus of this section. Nevertheless, traditional SM approaches should still be implemented in autonomous systems that use ML perception to deal with other sensors and ensure that the rest of the system appropriately handles the correct ML predictions.

Despite the importance of safety monitoring of ML-based perception functions, few research papers have explicitly focused on this topic. However, different approaches from the ML community have the potential to be used as detection mechanisms in safety monitors. They come from various ML sub-fields such as uncertainty estimation, anomaly detection, ensemble methods, or multi-modal perception. This section discusses all approaches that could be used to detect a failure of a critical ML perception function, even those not yet applied specifically to design safety monitors. They are classified into two main categories: *internal mechanisms* and *external mechanisms*. A visual representation of the proposed taxonomy of the existing detection mechanisms is proposed in Figure 3. We highlight that more research is needed to understand which of the techniques presented hereafter are viable alternatives for safety monitoring.

2.3.1 Internal mechanisms

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

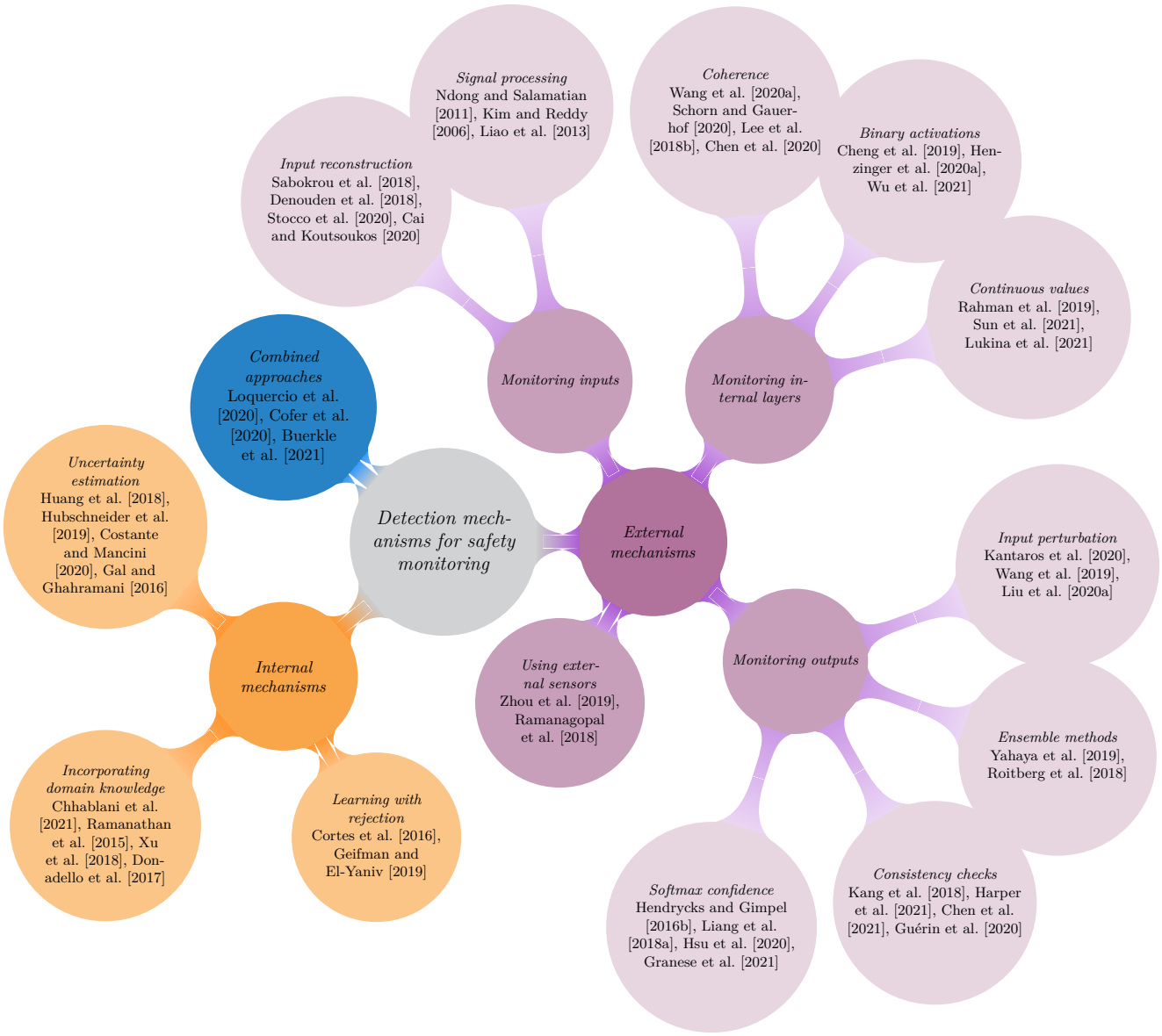


Figure 3: ***Taxonomy of detection mechanisms****. A visual representation of the different types of approaches to detect the failure of a critical ML-based perception function.*

2.3.1.1 Uncertainty estimation

Uncertainty estimation in deep learning has been widely studied recently (three surveys published in 2021 [Mena et al., 2021, Gawlikowski et al., 2021, Ulmer, 2021]). Most deep learning models produce point estimate predictions, i.e., 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, using the raw softmax output as a proxy for uncertainty has been widely questioned by the community as they often lead to overconfident predictions [Sensoy et al., 2018, Hendrycks and Gimpel, 2016b]. To tackle this issue, external calibration approaches have been proposed (see Section 2.3.2.3), and internal approaches as well.

The field that studies probabilistic neural networks is called Bayesian Deep Learning. In this setting, the weights of a neural network 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 for a given input data. These concepts from traditional Bayesian statistics [Seeger, 2006, Box and Tiao, 2011] are a natural way to reason about uncertainty in predictive models, but it comes with a prohibitive computational cost to be used in practice. Various approaches have been proposed recently to compute approximate Bayesian inference on large ML models to tackle this problem. Popular approaches include Variational Inference [Chen et al., 2018, Milios et al., 2018, Malinin and Gales, 2018, Rossi et al., 2019], Laplace approximation [Lee et al., 2018b] and sampling methods Welling and Teh [2011]. For a detailed review of recent Bayesian deep learning approaches, we refer the reader to one of the following works [Goan and Fookes, 2020, Mena et al., 2021].

Bayesian deep learning has already been applied to various tasks related to autonomous driving [Feng et al., 2021], such as semantic segmentation [Mukhoti and Gal, 2018, Huang et al., 2018], end-to-end vehicle control [Hubschneider et al., 2019] or visual odometry [Costante and Mancini, 2020]. Among all the available techniques, the ones based on Monte-Carlo Droupout [Gal and Ghahramani, 2016, Gal et al., 2017] appear to be the most popular for practical scenarios due to their simplicity of implementation. It uses Dropout layers during training and keeps them active at inference time to make stochastic predictions. Then, by running the model several times for a given input, one can compute statistics on the predictions to evaluate model uncertainty.

2.3.1.2 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 [Dash et al., 2021]. 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 [Marszalek and Schmid, 2007]. 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 Chhablani et al. [2021] 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. [2015] builds 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 [Xu et al., 2018], as well as first-order fuzzy logic to reason about logical formulas describing general properties of the data [Donadello et al., 2017].

2.3.1.3 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, i.e., the system produces no prediction. This kind of approach has been presented as a promising way to control the confidence in the monitored model in critical autonomous driving scenarios [Mohseni et al., 2020]. These approaches are internal mechanisms as they consist in 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 into traditional ML models such as support vector machine [Fumera and Roli, 2002], K-nearest neighbors [Hellman, 1970], and boosting [Cortes et al., 2016]. Recently, Geifman and El-Yaniv [2019] presented Selectivenet, a neural network architecture optimized to perform classification and rejection simultaneously.

2.3.2 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 which differ in their position in the ML perception pipeline. In particular, external detection mechanisms can monitor either the ML model inputs, internal representations, outputs, or even data from other sources.

2.3.2.1 Monitoring the DNN inputs

Some approaches predict failures of an ML perception model by monitoring its inputs, e.g., the raw images. These approaches 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. It is independent of the ML function, which facilitates the software engineering process. However, this independence is also a drawback as it is hard to predict if a neural network will fail on a given input without looking into the model itself. Next, we present the existing approaches.

- Traditional approaches Traditional signal processing approaches can be used to identify an anomalous sensory input before it enters the ML model [Ndong and Salamatian, 2011]. These approaches characterize some statistical patterns of "normal" data (i.e., from the training set) and compare them to 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 [Kim and Reddy, 2006]. This kind of approach was used in Liao et al. [2013] to identify water droplets in images for autonomous driving scenarios.

- Input reconstruction Recently, a popular technique has relied on unsupervised deep learning to identify anomalous images. It starts 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 [Sabokrou et al., 2018, Denouden et al., 2018, Stocco et al., 2020, Cai et al., 2020]. Another approach consists in training an outlier detection model on the latent representation of the auto-encoder to predict the nonconformity of new inputs [Cai and Koutsoukos, 2020]. 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 [Stocco and Tonella, 2020]. Finally, Feng and Easwaran [2021] proposed to detect unusual movements in real-time by combining optical flow operation with representation learning via a variational auto-encoder.

- Introspection A well-explored technique in the robotics domain aims at predicting future failures at runtime. For instance, Gurău et al. [2018] 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. [2020] 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.

2.3.2.2 Monitoring the DNN internal representations

Other ML failure detection approaches consider monitoring values taken by hidden layers of the model. These techniques are based on the idea that the training data alone is not sufficient to characterize what the model knows and that crucial information is contained in 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 [Gontijo-Lopes et al., 2021, Guérin et al., 2021b]. This section discusses approaches to monitor neural network activations at runtime.

- Continuous layer values Several works have proposed to detect failures of a neural network by looking at the values taken by the output of a given layer for new input data. For example, Rahman et al. [2019] proposed to detect abnormal data by training a binary classifier on features extracted from a given layer of the monitored ML model. Another recent work proposed to extract a representation from the penultimate layer of the model and to build a confidence estimator by truncating the activation values of the classification head [Sun et al., 2021]. A threshold is then applied to this confidence to decide whether an input is valid. Lukina et al. [2021] applied a centroid-based clustering approach to the internal representation of a given layer to characterize known inputs. The distance to the cluster centers is then used as an indicator to discard abnormal input data encountered at runtime. Finally, Wang et al. [2021] monitor the neurons within a faster R-CNN by representing distributions of neuron activation patterns and by calculating corresponding distances between them with the Kullback-Leibler divergence.

- 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 rectified linear unit (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, i.e., non-zero value. The advantage of considering such binary variables is that

they can be stored easily using binary decision diagrams [Cheng et al., 2019] or abstraction boxes [Henzinger et al., 2020a, Wu et al., 2021]. Then, abnormal data are identified by comparing activation patterns encountered at runtime to those recorded during training.

- Coherence between layers Other works have proposed to look into several hidden layers simultaneously. Wang et al. [2020a] proposed a tool called Dissector, which verifies if using the outputs of different layers leads to coherent decisions. Their goal is to identify inputs that represent unexpected conditions, which are considered outside of the model capabilities. Schorn and Gauerhof [2020] proposed an approach called FACER to build a feature vector representing the activations of different layers by summing the values of each feature map. Then a binary classifier is built by constructing a supervised dataset containing OOD data. Similarly, Lee et al. [2018b] fitted class conditional Gaussian distributions to both low-level and upper-level features of the deep learning model under Gaussian discriminant analysis and built a confidence score based on the Mahalanobis distance. Another approach was proposed in Chen et al. [2020], where the model's internal representations are tracked backward to build a saliency map of a given input. The patterns of the saliency map are then compared with the ones obtained for the training set in the same category.

2.3.2.3 Monitoring the DNN 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. The advantage of observing the neural network outputs is that it directly represents what needs to be correct, i.e., the information passed to the system's decision module. In contrast, the output layer contains less low-level information than earlier internal representations.

- Manipulation of softmax confidence A first intuitive technique to monitor the outputs of a deep neural network consists in properly establishing what values of the softmax confidence score can be considered reliable [Hendrycks and Gimpel, 2016b]. Several enhancements over this baseline method have been proposed to address the fact that softmax confidence scores are often poorly calibrated. Liang et al. [2018a] presented an approach called ODIN, which uses temperature scaling and small input perturbations to separate the softmax score distributions between in- and out-of-distribution data. A modified input pre-processing approach was introduced by Hsu et al. [2020] to allow ODIN to be trained without OOD examples. To increase detection performance, they also proposed confidence score decomposition to better represent regions with a high density of points. Finally, an approach to characterize the optimal discriminator was proposed in Granese et al. [2021]. A feasible implementation was presented based only on the softmax probability.

- Consistency checking Other approaches focus on verifying the spatial or temporal consistency of a sequence of predictions. The first technique to define constraints on output sequences is to use expert knowledge. This way, Kang et al. [2018] built a monitor for object detection by identifying flickering, i.e., an object should not keep appearing and disappearing in successive frames of a video. Another direction is explored in Harper et al. [2021], where the authors transform the rules and clauses of the official highway code into logical expressions (assertions), which can be applied as monitoring checks. Another possibility is to learn these patterns from data. Recently, Chen et al. [2021] proposed a logical framework to evaluate the temporal and spatial coherence of bounding box predictions and

thus identify erroneous detection results. Temporal coherence is assessed by observing the evolution of bounding box labels in consecutive frames. Spatial coherence is assessed by learning the standard size of bounding boxes at different locations. On another note, Guérin et al. [2020] proposed to build a consistency monitor for objects under periodic motion (e.g., production lines). They train a Gaussian process to estimate the probability for a bounding box to be at a specific location at a given time and use it to discard erroneous detections. Finally, Rabiee and Biswas [2019] proposed to detect failures of stereo vision-based perception from divergence in plans generated by a vision and a supervisory sensor. An advantage of these approaches is the possibility of collecting examples for further ML training with human labeling or weak supervision. These approaches have been developed for object detection problems.

- Ensemble methods A traditional way of improving the robustness of an ML model is to use an ensemble of other ML models to support the decision. A good ensemble should be composed of models that are "good", "independent" and "sufficiently numerous". In particular, such ensembles for SM can be composed of other models with the same architecture but trained to identify different aspects of the data or with different architectures trained with the same data [Theodoridis, 2015]. For DNN, Gontijo-Lopes et al. [2021] conducted a study about the influence of different training parameters on what the model knows, which can be used to maximize the ensembling benefits. In recent works, different voting strategies were proposed to address the problem of anomaly detection. For example, Yahaya et al. [2019] consider weighting each model vote based on its performance and a score for what is considered normal, while Roitberg et al. [2018] leverage the estimated uncertainty of each model in their predictions to measure the novelty of new input. When computational power is not an issue, building such ensembles of deep neural networks appear as a promising idea to monitor coherence between individual models and avoid propagating errors of a single neural network. Finally, Roy et al. [2022] 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.

- Robustness to input perturbation To verify if a new input can be considered "normal", it was also proposed to measure its sensitivity to input perturbations. Such perturbations can be applied either to the data (e.g., image compression [Kantaros et al., 2020]) or to the ML model itself through random mutations [Wang et al., 2019]. Another possibility is to check the stability of a model within a radius distance calibrated during the training [Liu et al., 2020a]. 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. [2018] proposed DeepRoad, a generative adversarial network (GAN) based metamorphic testing and input validation framework for autonomous driving systems.

2.3.2.4 External sensors

To conclude this section, we present one last family of approaches to detect failures of deep neural networks. It aims at checking whether the ML predictions are consistent with signals coming from other sensors. This way, Zhou et al. [2019] 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 accuracy at runtime. Similarly, Ramanagopal et al. [2018] 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, i.e., missed detections. Finally, Li et al. [2022] presents an approach to monitor extrinsic camera calibration quality by using inertial measurement unit (IMU) data to capture mismatches of road image features.

2.3.3 Combined detection approaches

Several recent approaches have proposed combining different techniques presented above to build more robust detection systems. Loquercio et al. [2020] proposed to monitor data uncertainty and model uncertainty separately, using two distinct mechanisms. First, the noise characteristics of the sensor are propagated through the network to account for data uncertainty in future predictions. Then, model uncertainty is assessed using Monte-Carlo Dropout [Gal and Ghahramani, 2016], and the total uncertainty is obtained by combining both sources using stochastic using Assumed Density Filtering. Buerkle et al. [2021] 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). Cofer et al. [2020] use four different monitors for a task of end-to-end aircraft taxiing. In particular, they combine information from different sensors (GPS, Inertial Reference System), standard computer vision algorithms, and input reconstructions approach to build a robust neural network monitoring system. Finally, Guerin et al. [2022a] combined three monitors (Monte-Carlo Dropout, local resolution increase, and classification hierarchy) for the task of drone emergency landing.

2.4 Which recovery mechanisms can be used to build SM?

In Section 2.3, many approaches were presented to detect uncertain and potentially hazardous predictions of an ML-based perception function. When activated, these detection mechanisms raise a safety alert to the autonomous system, which must take appropriate actions to avoid hazardous situations. In most ML monitoring studies, the basic alert is a flag, and more complex actions are usually not investigated. We call such actions *recovery mechanisms*, which we discuss them in this section.

2.4.1 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. [2019] proposed the neural simplex architecture, allowing it to switch 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. It is hard to design simple controllers in many practical scenarios, not relying

on ML for state estimation. As a result, for complex autonomous systems relying on visual perception, the default recovery actions often consist in switching back control to a human driver [Stocco et al., 2020] or triggering an emergency breaking [Cofer et al., 2020]. The former can be hazardous when a fast reaction is required. The latter might not be safe for specific scenarios, e.g., 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.

2.4.2 Immediate prediction enhancement

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

- Input reconstruction The first family of approaches consists in improving the quality of the input of the ML model. These techniques are used when one can identify the cause of the wrong prediction from the input image, and they mostly consist of removing the specific type of noise that was found. Recently, most image-denoising approaches use autoencoders, trained to reconstruct the original image without the identified noise pattern [Gondara, 2016]. For example, image dehazing algorithms can be used to react to fog or smoke [Abdulkareem et al., 2021], approaches have been proposed to react to different light exposure conditions [Yan et al., 2022, 2021], saturation [Liu et al., 2021c], water drops [Liao et al., 2013, Qian et al., 2018], or even more standard Gaussian noise or impulse noise [Zhou et al., 2020, Zhang and Gao, 2021]. Other works proposed to enhance the original image by increasing its resolution artificially [Wang et al., 2020b] by using convolutional neural networks [Shi et al., 2016] and generative adversarial networks [Ledig et al., 2017, Marinescu et al., 2021]. Furthermore, to deal with images having chunks of pixels damaged by sensor failures, image inpainting techniques can be used [Saharia et al., 2021]. Finally, to handle errors related to incomplete color information (mosaicked images), some works have applied demosaicing techniques [Ni et al., 2020] or gradient-based feature extraction [Zhou et al., 2021].

- Changing final prediction To respond to the detection of an adversarial example, Al-Afandi and Horváth [2021] 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. [2021] 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. [2019b] could be extended to downgrade classification at runtime, e.g., if a high classification uncertainty is detected between the classes “car” and “truck”, the prediction could be changed to the sup-class “vehicle”.

- 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 [Kakamanshadi et al., 2015]. When high uncertainty is detected, one might use sensors with higher resolution locally (spatially or temporally) to get a better prediction [Guerin et al., 2022a]. 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 [Gontijo-Lopes et al., 2021].

2.4.3 Long-term system enhancement

Finally, we also discuss how runtime threats identified by a monitoring detection mechanism can be used to improve the safety of the monitored system 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 [Andrej Karpathy, 2021]. 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 [Van de Ven and Tolias, 2019, Liu et al., 2021a]. 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 to build relevant datasets to test future developments of safety-critical systems.

2.5 How to evaluate SM?

A proper evaluation protocol for monitoring should verify whether the SM presents the following characteristics:

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

This section discusses how the existing approaches for safety monitoring of ML-based perception functions are evaluated. We first discuss how the detection mechanisms presented in Section 2.3 are evaluated. As they usually are generic components, they often can be evaluated independently of a given system. Then, we show how the few existing approaches to monitor ML perception components at the system level have been evaluated. Finally, we discuss how the impact of research on SM could be increased by creating new, unified evaluation methodologies.

2.5.1 Evaluation of detection mechanisms

Most detection approaches presented in Section 2.3 intend to identify bad input data, which would produce erroneous output when processed by the ML model. However, instead of predicting model failures, these approaches are often evaluated on the surrogate problem of identifying specific runtime threats. As explained in Section 2.1, different kinds of changes in the input data encountered at runtime, which we call runtime threats, can hinder the performance of a neural network. This section presents the datasets and metrics from the literature to evaluate monitors for runtime threats.

2.5.1.1 Evaluation datasets

In this subsection, we present how datasets are built for the evaluation of the SM detection mechanism. It can be seen as a fault injection focused on the threats presented in Section 2.1.

- Novelty detection: This setting aims at identifying when the class label of an input image does not belong to any of the predefined outputs of the ML model. Thus, to evaluate the capacity of an SM to identify novel inputs, most approaches have modified standard image classification datasets such as Imagenet [Deng et al., 2009], MNIST [Deng, 2012] or CIFAR [Krizhevsky, 2009], among others. In particular, two main strategies can be used to create novelty detection datasets. The first one consists in merging two datasets with non-overlapping 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 [Sun et al., 2021, Schorn and Gauerhof, 2020, Lee et al., 2018b, Hendrycks and Gimpel, 2016b, Liang et al., 2018a, Hsu et al., 2020, Shafaei et al., 2019]. The second strategy uses a single dataset but splits the images into two subsets with disjoint labels [Denouden et al., 2018, Sabokrou et al., 2018, Lukina et al., 2021, Henzinger et al., 2020a, Wu et al., 2021, Roitberg et al., 2018].

- Distributional shift detection: This setting 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. Then, the goal is to identify corrupted images. Several papers have proposed to inject artificial corruption [Hendrycks and Dietterich, 2019] into standard image classification datasets [Rahman et al., 2019, Schorn and Gauerhof, 2020, Hendrycks and Gimpel, 2016b, Liang et al., 2018a, Hsu et al., 2020]. Others have injected faults into realistic autonomous systems scenarios. Some have used autonomous vehicle simulators, such as CARLA [Dosovitskiy et al., 2017], to simulate different kinds of driving conditions (weather, light) [Feng and Easwaran, 2021, Cai and Koutsoukos, 2020]. Others have applied artificial perturbations to existing real-world datasets for autonomous driving [Chen et al., 2020, 2021, Zhang et al., 2018, Zhou et al., 2019] or UAV emergency landing [Guerin et al., 2022a].

- 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 in applying an adversarial attack (see Section 2.1.4 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 [Kantaros et al., 2020, Wang et al., 2019, Liu et al., 2020a] and simulated autonomous driving scenarios [Cai and Koutsoukos, 2020, Chen et al., 2020].

- Predicting model errors: Finally, instead of identifying specific threats, several detection mechanisms have been evaluated directly to identify inputs for which the ML model fails, i.e., activation of the SM is correct if the prediction of the ML model is wrong. Several papers have proposed experiments to evaluate the ability of an out-of-distribution detector to identify failures of the monitored neural network on the test split of the training

dataset [Cheng et al., 2019, Wang et al., 2020a, Hendrycks and Gimpel, 2016b]. Others further applied some of the perturbations presented above before conducting such an evaluation [Kang et al., 2018, Granese et al., 2021].

2.5.1.2 Evaluation metrics

In the previous section, we discussed ways to create evaluation datasets containing both "good" and "bad" data, thus defining binary classification problems. Hence, most of these papers assess the strength of their approaches using standard binary classification metrics. Here, we consider that a True Positive (TP) is a rejected invalid input, a True Negative (TN) is an accepted valid input, a False Positive (FP) is a rejected valid input, and a False Negative (FN) is an accepted invalid input. These metrics are defined as follows:

- 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 to find 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 Rahman et al. [2019].
- 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 might not be 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.

Finally, it is also important to consider the detection mechanism's execution time and memory to estimate the computational overhead by using such a safety monitor.

2.5.2 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, as explained in Salay et al. [2019b], some perception errors can generate catastrophic events (e.g., missing a pedestrian crossing a road). 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. [2020] 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. [2020], Stocco and Tonella [2020], 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. [2020] for an aircraft taxiing application. On their experimental dataset, they were able to avoid all the cases where the neural network leads the aircraft to exit the runway thanks to their safety monitoring system. Finally, Guerin et al. [2022a] 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 to the system.

2.5.3 Open-challenges for SM evaluation

Some open challenges are discussed in this Subsection.

Evaluation for certification: specific evaluation and certification procedures for autonomous systems were proposed in the literature. Myers and Saigol [2020] 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 which must not occur too often. De Gelder and Den Camp [2020] 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. [2021] 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, Guérin et al. [2021a] assessed the requirements to certify UAV operations in urban environments using a document called Specific Operations Risk Assessment (SORA) Joint Authorities for Rulemaking of Unmanned Systems (JARUS) [2019], which provides guidelines to develop and certify safe UAV operations.

As mentioned above, recent works have discussed evaluation frameworks for autonomous vehicles. However, 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.

Evaluation coverage: when performing an evaluation, we need to consider two different scenarios: simulated, and real-world scenarios.

- Simulated scenarios: to test perception functions, it is frequent to use a simulation environment [Stocco et al., 2020, Loquercio et al., 2020, Buerkle et al., 2021]. This allows resetting the environment to a previous configuration and comparing the responses to different perceptions 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 [Zandbergen, 2021]. 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 [Borg et al., 2019].

- Real-world scenarios: 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 [Jahangirova, 2017]. 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 [Rabanser et al., 2019].

2.6 Conclusion

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 chapter presents a comprehensive survey about the safety monitoring of ML perception functions, addressing every step of the development process, i.e., 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 followings:

- Defining SM 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 show in Chapter 3, 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.

- 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.

- Combining SM 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 [Kontos et al., 2021], model assertion [Kang et al., 2020], and classification failure mode and effects analysis [Salay et al., 2019b], can be applied to complement data-based monitors. Studying how to best combine several safety monitors and verify the consistency of their outputs is an important open challenge for the field.

- Implementation constraints: the SM discussed in this work is expected to perform within embedded systems. Hence, it is essential to design SM that can function under limited computing power and memory. Moreover, that complies with the system's constraints regarding energy consumption. In addition, an SM is executed at runtime, i.e., it must follow strict requirements regarding execution time to ensure synchronization with the main monitored system.

- 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.

As a first step to address the mentioned challenges, 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. For example, in the next chapter, we propose two dimensions of evaluation that need to be investigated: the SM's overall impact on the system under test; and the detection performance of the SMs. Hence, we select one state-of-the-art SM from each of the three main types of data-based SM presented in this chapter, and we develop a benchmark framework that applies OOD perturbations on images.

Chapter 3

Testing SM for ML perception functions at the unit level

In this chapter, we focus on unit testing of the monitor, which is isolated from the rest of the system. Many recent works focus on monitors dedicated to ML model surveillance. They broadly fall into three types of monitors: observation of the inputs of the ML model [Liu et al., 2020a], its outputs [Hendrycks and Gimpel, 2016a] or from intermediate layers in case of DNN [Cheng et al., 2019, Henzinger et al., 2020b]. However, they are all based on the exploitation of the training data. Therefore, we refer to them as data-based monitors compared to safety monitors based on rules (or safety properties). Similar to uncertainties inherent to the use of ML, confidence in such SM is an open issue. Thus, it is essential to estimate how efficient such monitors are and if it is possible to include them in a safety context. This estimation should consider all potential situations leading to an error of the ML and should be based on metrics dedicated to measuring the monitor efficiency. Therefore, we focus on a framework for benchmarking such monitors, augmented with an additional primary mechanism to inhibit the decision in error detection.

As presented in Figure 4, the complete system under test is composed of an SM that inspects the ML model. Our benchmark adapts and extends current metrics used in the ML community to estimate the SM detection performance at runtime, its impact on the system, and the overhead induced by the use of the monitor. Our main contributions are:

- *A new baseline framework for benchmarking SM for ML classifiers.* To the best of our knowledge, this is the first work that proposes an initial framework that benchmarks SM for ML components from different perspectives.
- *A comprehensive benchmark experiment containing different* ***data-based*** *SM methods, datasets, and results.* Our experiments reveal the advantages and drawbacks of the main modern data-based SM approaches for image classifiers built with ML. We perform experiments on five challenging types of out-of-distribution data for image classifiers.

This work is organized as follows: in Section 3.1, we present an overview of our framework, along with its main objectives. In Section 3.2 we present our experiments, datasets, and results from our framework to three ML monitors. Finally, in Section 3.3, we present our conclusions.

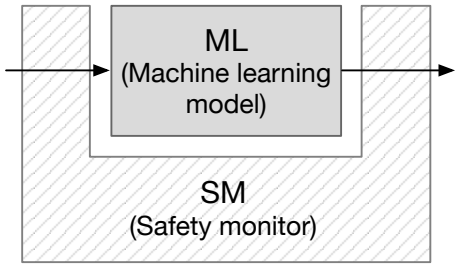


Figure 4: ***System under test.*** *It is composed of ML+SM.*

3.1 Benchmarking SM for ML classifiers

The proposed framework is an adaptation of the FARM [Arlat et al., 1990] methodology. FARM is a methodology for fault injection for dependability validation. It uses the input domain of a target system as a set of faults F and a set of activations A that specifies the domain to test the target system. The output domain refers to a set of readouts R used for posterior evaluation with a set of metrics M. Our framework is divided into three modules, as illustrated in Figure 5.

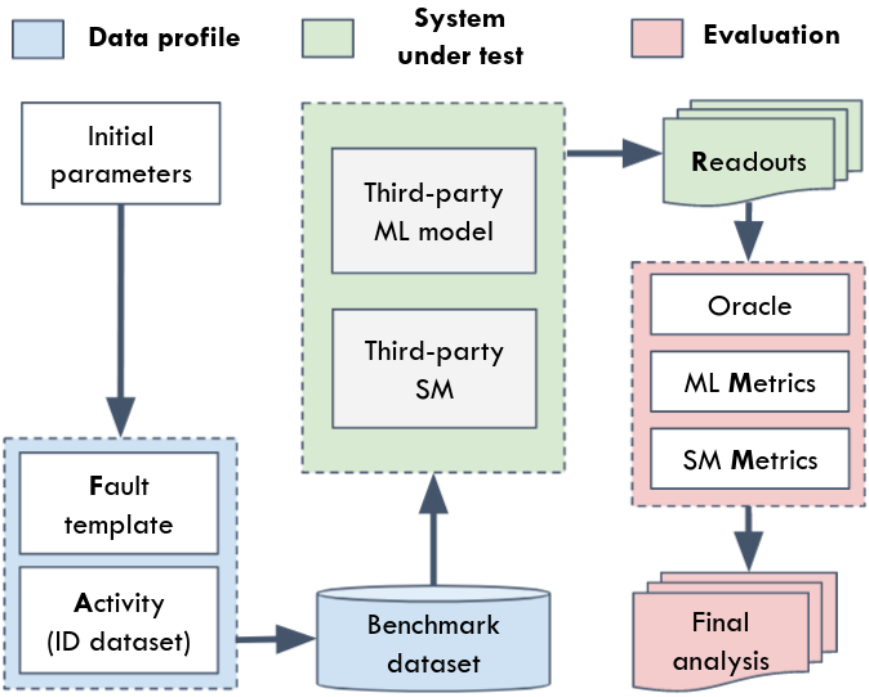


Figure 5: ***A high-level overview of the framework.*** *It is divided into three modules.*

After setting some initial parameters such as type of data generation, the amount of data to be tested, and which ML models and SM to include, start with the first module, called the data profile. This module generates benchmark datasets in the next module, called SUT.

The SUT is responsible for testing the ML and the SM performance, generating the results (readouts) at the end of the process. Readouts are inputs for the evaluation module.

The evaluation module is responsible for analyzing the readouts through several metrics for different components of the system. It provides the final analysis for each type of readout. Next, we explain each module and its components.

3.1.1 Data profile

This is the first module and it is composed of two items:

- **Activity**: it contains in-distribution (ID) data, that is, instances from a distribution

known by the ML.

- **Fault template**: it contains rules for generating OOD data.

The benchmark dataset receives the ID and OOD data to test how the ML and the SM behave when exposed to expected and unexpected data. Even though all types of OOD data can be part of a benchmark dataset, we generated datasets for each category of OOD data.

An essential premise is that the generated benchmark dataset is not applied to build or validate the SM or train the ML model. This premise guarantees unbiased experiment results and is more realistic since, in real scenarios, there are no guarantees about which type of data will arrive at runtime.

3.1.2 System under test

The system under test (SUT) module receives the benchmark dataset as input and outputs readouts for each component. To validate our framework, we chose ML models that classify images and one SM for each of the three categories: based on input, internal mechanisms, and outputs. Therefore, this module is composed of two main items: a third-party **ML model** already trained; and a third-party **safety monitor** containing a detection mechanism. We call the ML and the SM third-party components since they can be built aside from the framework.

This module simulates a stream of randomly ordered images. We apply datasets with random images since the image datasets available in the literature use such a setting.

The simulation works as a stream of images coming from the benchmark dataset set D that contains images X and labels y, arriving at one of each at a time. The ML model receives this image and makes a $\hat{y}$ classification.

Next, the system triggers the SM, which checks a set of predefined properties. These properties can be the ML's classification $\hat{y}$ with the associated confidence level, intermediate layers, the input X, or even a combination of two or more of these properties. In the case of this pseudo-code example, the SM inspects the ML model's internal properties (DNN's hidden layer) during the classification, as recently suggested in the literature [Cheng et al., 2019, Henzinger et al., 2020b].

After the inspection, the SM detector mechanism raises a detection alarm or not ($\hat{m}$). For this work, we consider a simple reaction strategy for SM. If an alarm is raised, it invalidates the ML classification result; otherwise, it accepts the ML classification. Next, we observe whether such intervention produced a desirable outcome for the system or not. We call it *overall detection* $\hat{s}$. A desirable outcome means that canceling an ML output/agreeing with it was beneficial to the SUT. This result is independent of whether the SM-specific detection was correct or not in the OOD detection task. The criteria for considering these two dimensions of the detection (specific and overall) correct or not is discussed in the following subsection. This process continues until the end of the stream. Thus, the readouts are the ground truth y, the ML classification $\hat{y}$, the SM specific detection $\hat{m}$, and its overall detection $\hat{s}$. They are divided into three categories:

1. **ML readouts**: it contains classification (e.g., class number), confidence value from the last layer, and intermediate values from hidden layers.

2. **SM readouts**: it contains two types of detection: a) specific (e.g., it outputs 1 for OOD detection; 0 otherwise); b) overall (e.g., after OOD detection, it cancels the ML decision (1); 0 otherwise).

3. **General readouts**: it contains general outputs from all components above, such as processing time and memory.

As can be noted, the SM readouts contain results in two dimensions: specific and overall. Considering two dimensions for detection is more realistic and complete than just analyzing the detection rate for the OOD data. The reason is that while the SM tries to detect OOD data, it also can avoid the ML from giving an answer that is different from the ground truth. The opposite is also true. As a result, the SM can incorrectly detect ID data as OOD data, hindering the correct decisions of ML. All possible situations are given in the next subsection.

3.1.3 Evaluation

This module receives the readouts as inputs containing false positives/negatives and true positives/negatives regarding OOD data detection's specific task and the overall task of avoiding an unsafe outcome. This module evaluates two main aspects:

1. The SMs performance: the objective is to assess the SM results regarding the specific task of detecting OOD data. This evaluation's inputs are SM OOD detection (e.g., raises/does not raise the alarm).

2. The overall impact of the SM: the objective is to determine if the SM improves or worsens the overall SUT accuracy. We analyze the SUT using the ML alone (baseline) and ML with the SM. This evaluation's inputs are the overall decisions made by the SUT with ML alone and ML with SM, processing time, and memory.

3. Time and memory overhead induced by SMs: the objective is to investigate the memory and processing time efficiency of these methods during runtime.

This module contains two major items: the **oracle**, and the **metrics for ML and SM**. The oracle determines positive and negative data and whether a test has passed or failed. Thus, the oracle considers a correct specific detection when the SM correctly detects OOD data. Besides, the oracle considers a correct overall detection for the SUT when the SM correctly avoids a wrong ML classification for ID or OOD data. The oracle takes into consideration the following scenarios to determine whether a readout is correct or incorrect:

- **ID data arrives in the stream**
  - **If the SM detects OOD data:** it means a *false positive for the specific task* of OOD detection.
    - **If the ML classification is equal to the ground truth:** it means a *false positive for the overall task* of avoiding a failure since the SM intervened without necessity.
    - **If the ML classification is different than the ground truth:** it means a *true positive for the overall task* of avoiding a failure since it canceled the ML misclassification for ID data.
  - **If the SM does not detect OOD data:** it means a *true negative for the specific task* of OOD detection.

    - **If the ML classification is equal to the ground truth:** it means a *true negative for the overall task* of avoiding a failure since the ML gave a correct classification even though the SM did not detect OOD data.
    - **If the ML classification is different than the ground truth:** it means a *false negative for the overall task* of avoiding a failure since the ML misclassified the OOD data and the SM did not detect it, which could cancel a wrong ML output.

- **OOD data arrives in the stream**

  - **If the SM detects OOD data:** it means a *true positive for the specific task* of OOD detection.
    - **If the ML classification is equal to the ground truth:** it means a *false positive for the overall task* of avoiding a failure since the ML gave a correct classification but the SM wrongly intervened.
    - **If the ML classification is different than the ground truth:** it means a *true positive for the overall task* of avoiding a failure since the ML gave an incorrect classification and the SM correctly canceled the ML decision.
  - **If the SM does not detect OOD data:** it means a *false negative for the specific task* of OOD detection.
    - **If the ML classification is equal to the ground truth:** it means a *true negative for the overall task* of avoiding a failure since the SM detection for OOD data also avoided an ML misclassification.
    - **If the ML classification is different from the ground truth:** it means a *false negative for the overall task* of avoiding a failure since the SM detection for OOD data also avoided an ML misclassification.

The only exception to the above rules is novelty class detection. When OOD data arrives in the stream, if the SM correctly detects OOD data, it will always be interpreted as true positive since the SM always correctly cancel the ML classification, independently of the ground truth. Conversely, if the SM does not detect OOD data, it will always be a false negative. If the SM lets the ML deal with a class that it was not trained in before, it can be regarded as unsafe. In order to evaluate these situations, we have selected in the literature a set of metrics that are pertinent to our study. Hence, we apply seven metrics:

- *Matthews correlation coefficient (MCC)*: it ranges from -1 if (ML or SM always wrong) passing through 0 (ML or SM is accurate as random) to 1 (ML or SM always right). This metric is more reliable than traditional metrics such as accuracy. It yields a high score only if the ML or SM can be assertive in all of the four confusion matrix categories [Chicco and Jurman, 2020]. In contrast, accuracy only considers the portion of the right answers. In a scenario in which the number of a class is 80%, the accuracy could yield a score of 80% for an SM that did not detect other classes.

- *False positive rate (FPR)*: also known as type-I error. It indicates how many false alerts the SM raises for the task to detect OOD data. High values mean the SM indicates a problem wherein there are not in most cases.

- *False negative rate (FNR)*: also known as type-II error. It indicates how often the SM misses detecting OOD data. If this value is high, it indicates that the SM does not recognize the difference between the ID and OOD data.
- *Precision and recall (Pr and Re)*: the fraction of correct detection and the fraction of available OOD data. These metrics help to indicate how well the SM detected OOD data through the benchmark dataset.
- *Micro-F1*: harmonic mean (global) for the prediction x recall. It helps to assess the quality of multi-label binary problems, which makes them suitable to be applied in the SUT evaluation.
- *Critical difference diagram*: it is applied for all MCC results through all datasets. It shows how statistically different are the results between the SMs.

Next, we detail each component's choice, its parameters, and the benchmark results.

3.2 Experiments and results

3.2.1 Data profile

We use three datasets as the *activity* of our framework:

- GTSRB [Sermanet and LeCun, 2011]: German traffic signs with 43 classes, with 39200 instances for training and 12600 for testing.
- BTSC [Jain et al., 2019]: Belgium traffic signs with 62 classes, divided into 7000 images for training and testing.
- CIFAR-10 [Krizhevsky and Hinton, 2010]: ten general classes (e.g., dog, car ...), with 50000 instances for training and 10000 for testing.

Next, we apply a *fault template* for the five classes of OOD presented in Chapter 2.1: novelty, anomaly, distributional shift, noise, and adversarial inputs. For adversarial inputs generation, we apply the fast gradient signed method (FGSM) [Dong et al., 2017]. For the noise and distributional shift, we generate 19 different transformations with two types of intensity varying from 1 to 5 (e.g., snow (1) = image with a bit of snow; snow (5) = heavy snow) [Hendrycks and Dietterich, 2019]. All transformations were applied over CIFAR-10 and GTSRB datasets. The benchmark datasets are composed of a transformed version of the ID datasets that we previously applied for training the ML algorithm and building the SM. Thus, we use 20% of the ID data for the benchmark dataset without transformation. The entire original dataset is transformed into a specific variation. This 20% of ID data was not applied to the ML training.

For novelty class detection, the fault template applies one dataset as an ID dataset and another dataset with new classes as OOD data, resulting in three benchmark datasets:

- GTSRB as ID data, and BTSC as OOD data: this combination tests whether the SM can distinguish new classes that have similar characteristics to the known ones.
- GTSRB as ID data, and CIFAR-10 as OOD data: this combination tests whether the SM can distinguish new classes that are very different from the known ones.

- CIFAR-10 as ID data, and GTSRB as OOD data: this combination tests the same as the aforementioned combination. However, since the ID data is different, the ML and the SM are built with different data. Hence, this permutation produces different outcomes.

In total, we produced 79 benchmark datasets. More details can be found in the results section of our repository [Anonymous, 2021].

3.2.2 System under test

We simulate a scenario in which the SM has to detect OOD data by checking one RGB-colored image at a time in a randomly ordered stream of images. Once the SM makes a detection, it cancels the ML classification since this classification is potentially wrong. If nothing is detected, the ML classification is accepted.

For the ML model, we use a LeNet [LeCun et al., 2015] since it is a simple and traditional convolutional neural network (CNN) algorithm. It contains around 100,000 parameters and 128 neurons in the last hidden layer. For the monitors (noted as SM), we use one method of each of the three different strategies of data-based SM for monitoring a DNN model:

- *DNN inputs*: adversarially learned one-class classifier for novelty detection (ALOOC) [Sabokrou et al., 2018]. This method learns how to reconstruct each class during the training phase. It receives an image during operation, tries to reconstruct this image to the known class, and analyzes the loss error resulting from this reconstruction. If this loss error surpasses a safe threshold, ALOOC flags it as OOD.

- *DNN intermediate values*: outside-of-the-box abstraction (OOB) [Henzinger et al., 2020b]. This method projects a 2D-box region from the activation function values (RELU) from the hidden layers of the DNN during the training phase. At runtime, it receives an image and projects a point from it built from the outputs of the activation function from the same DNN layer used during the training. If this point falls outside the 2D box, OOB will flag this image as OOD.

- *DNN outputs*: detector of out-of-distribution images in neural networks (ODIN) [Liang et al., 2018a]. This method learns how to balance the confidence values outputted along with the DNN decision. It uses these values from the last layer of DNN during training and applies a method known as temperature scaling, which will scale the confidence values to a new threshold for considering an image as OOD. During the operation, ODIN verifies if the DNN confidence value over an incoming image respects the threshold. If not, ODIN will flag this image as OOD.

3.2.3 Parameter's evaluation and choice

Next, we present the chosen parameters for the SMs. To avoid biased results, all the initial parameters were tuned using only the ID dataset and are summarized in Table 1.

We explored the original parameters for each SM, and we also found/tested new values/methods not mentioned in their original papers (marked with *). For ALOOC, the authors did not investigate the influence of the optimizers in the original paper. Thus, we tested two different optimizers: ADAM [Zhang, 2018] and RMSProp [Wichrowska et al., 2017]. We also analyzed the best model for each class through different epochs. In general, all the models achieved a better

Table 1: ***SM parameters tested in the experiments.*** *We tested parameters beyond the ones tested in the related works papers.*

SM	Parameter name	Parameter values
ALOOC	Optimizer	ADAM, RMSProp
	Epoch	200
	Loss threshold	average per class*
OOB	γ	0, 0.1, 0.35
	# of clusters	0, 3, 5*, 17*
	Dimensionality reduction	simple, PCA*, ISOMAP*
ODIN	Temperature	1000
	Magnitude	0.0014, 0.0025
	Confidence threshold	0.0237*, 0.1007*

convergence around 200 epochs. Since the threshold value for considering whether a class is considered OOD does not indicate in the original paper, we considered the average reconstruction loss for each class during the training.

For the OOB method, we test the parameter responsible for enlarging the size of boxes (τ) with the same range of values proposed by the original paper. We applied the same number of clusters as suggested by the original paper (no clusters or three clusters). However, we also tested with a possible optimal number of clusters. The possible optimal number of clusters K for the K-means algorithm contained in the OOB method is chosen through the Elbow analysis [Kodinariya and Makwana, 2013]. Thus, the best K value was 5 and 17 for CIFAR-10 and GTSRB, respectively. Finally, we also tested three different approaches for the 2D-dimensionality reduction parameter: simple projection (proposed in the original paper), PCA [Yang et al., 2004], and ISOMAP [Balasubramanian et al., 2002]. ISOMAP is a popular nonlinear dimensionality reduction method, and PCA is also a popular method, but linear. With these two methods, we can analyze if the choice of the dimensionality reduction methods influences the outcomes.

Regarding ODIN, we set the temperature (1000) and the magnitude parameters (0.0025 and 0.0014, for GTSRB and CIFAR-10, respectively) as suggested by the authors. However, we chose the confidence thresholds for determining OOD data by selecting the lower confidence value outputted from the method when exposed to the training data. In this case, 0.0237 and 0.1007, for GTSRB and CIFAR-10, respectively. We had to assume a confidence value threshold since it would not be possible to use ODIN as an SM without it.

3.2.4 Results

3.2.4.1 SMs performance

we use the positive/negative data and the evaluation metrics as mentioned in Section 3.1. Since we measure the detection, there is no necessity to include the measurements for the ML. Best results are written in **bold**.

Table 2 shows these results for *novelty class*: GTSRB as ID dataset and BTSC as OOD dataset; CIFAR-10 as ID and GTSRB as OOD; and GTSRB as ID and CIFAR-10 as OOD. The first benchmark dataset has the challenge of having ID and OOD data with a similar domain, while the other two have very different classes from each other.

The MCC results indicate that the SM is sometimes slightly better than a random classifier

Table 2: ***Comparing data-based monitors for*** **novelty class.** *SMs yielded a high amount of false positives.*

Variation	SM	MCC	FPR	FNR	Pr	Re	F1
GTSRB-BTSC	ALOOC	0.01	**0.50**	0.49	0.17	0.51	**0.57**
	OOB	**0.23**	0.61	0.11	**0.24**	0.90	0.52
	ODIN	0.03	0.99	**0.0**	0.16	**1.0**	0.06
CIFAR10-GTSRB	ALOOC	0.02	0.63	0.34	0.18	0.66	0.47
	OOB	0.11	0.72	0.16	0.20	0.84	0.41
	ODIN	**0.23**	**0.61**	**0.10**	**0.24**	**0.9**	**0.52**
GTSRB-CIFAR10	ALOOC	0.05	**0.56**	0.37	0.81	0.63	0.63
	OOB	**0.15**	0.79	0.09	**0.82**	0.91	**0.74**
	ODIN	0.07	1.0	**0.02**	0.17	**0.98**	0.06

Table 3: ***Comparing data-based monitors for CIFAR-10 and GTSRB datasets with a*** **adversarial attack.** *Methods based on inputs and internal values yield a high amount of false positives while monitor based on outputs suffers from a high amount of false negatives.*

CIFAR-10							
Variation	Method	MCC	FPR	FNR	Pr	Re	F1
FGSM	ALOOC	-0.23	0.89	0.29	0.28	0.71	0.25
	OOB	-0.13	0.92	**0.16**	0.31	**0.84**	0.24
	ODIN	**0.06**	**0.14**	0.81	**0.37**	0.19	**0.62**
GTSRB							
Variation	Method	MCC	FPR	FNR	Pr	Re	F1
FGSM	ALOOC	**0.19**	**0.22**	0.59	0.44	0.41	**0.66**
	OOB	-0.01	1.0	**0.0**	**0.31**	**1.0**	0.14
	ODIN	0.11	0.92	0.02	0.34	0.98	0.26

(MCC ≈ 0). We observe that these methods' weakness is in the high amount of false positives, yielding a borderline MCC performance. According to the results, just ALOOC obtained a false-positive rate of around 50%. However, it had many false negatives, which can be considered worse depending on the scenario. In general, all SM had a poor performance due to the unreliable nature of DNN confidence values, and the high nonlinear nature of activation functions in ODIN and OOB, respectively.

Next, in Table 3, we show the same analysis but using CIFAR-10 and GTSRB after being modified through an *adversarial attack* known as FGSM.

According to the results, for CIFAR-10, the methods achieved negative values for MCC, which indicates performance worse than a random classifier. For GTSRB, the results are slightly better, but the MCC values can be considered as flawed as a random classifier. Despite the good values of micro-F1 for ODIN and ALOOC, in the CIFAR-10 and GTSRB datasets, respectively, the rate of false negatives was high. It means that the ML classifier did not give the correct prediction, and the SM did not detect the attack.

Next, in Table 4, we show the results for CIFAR-10 and GTSRB datasets with different

types of *distributional shift*. Here, we see a surprising but negative outcome: OOB suffers from a lot of false positives. In contrast, ODIN suffers from many false negatives. It means that they always tend to miss a detection or to say that everything is OOD data. ALOOC got a fair MCC value for CIFAR-10 with intense fog and a good MCC value for GTSRB with heavy snow.

Table 4: *Comparing data-based monitors for CIFAR-10 and GTSRB with* distributional shift. *A high amount of false negatives from all methods most of the time.*

CIFAR-10							
Variation	Method	MCC	FPR	FNR	Pr	Re	F1
rotated	ALOOC	0.0	**0.0**	0.29	**1.0**	0.71	**0.83**
	OOB	**0.02**	1.0	**0.0**	0.14	**1.0**	0.04
	ODIN	-0.1	0.32	0.81	0.09	0.19	0.66
snow (5)	ALOOC	-0.01	0.42	0.59	0.14	0.41	0.62
	OOB	0.0	1.0	**0.0**	0.14	**1.0**	0.04
	ODIN	**0.14**	**0.08**	0.81	**0.29**	0.19	**0.8**
fog (5)	ALOOC	**0.47**	**0.03**	0.59	**0.68**	0.41	**0.88**
	OOB	0.0	1.0	**0.0**	0.14	**1.0**	0.04
	ODIN	-0.01	0.21	0.81	0.13	0.19	0.73
GTSRB							
Variation	Method	MCC	FPR	FNR	Pr	Re	F1
rotated	ALOOC	0.0	**0.0**	0.55	**1.0**	0.45	**0.62**
	OOB	**0.09**	0.75	0.16	0.21	0.84	0.38
	ODIN	-0.12	1.0	**0.02**	0.19	**0.98**	0.06
snow (5)	ALOOC	**0.81**	**0.0**	0.29	**1.0**	0.71	**0.94**
	OOB	0.01	0.83	0.16	0.2	0.84	0.29
	ODIN	0.16	0.85	**0.02**	0.22	**0.98**	0.28
fog (5)	ALOOC	-0.28	0.93	0.29	0.16	0.71	0.15
	OOB	**0.0**	**0.84**	0.16	**0.2**	0.84	**0.28**
	ODIN	**0.0**	0.98	**0.02**	**0.2**	**0.98**	0.1

Another interesting result for ALOOC is that despite the low MCC performance on the other benchmark datasets, it got relatively good micro-F1 results. It highlights that this method tends to have a good amount of false negatives but rarely gives a wrong output when detection is signaled. Again, methods based on inspecting inputs seem to be more effective in detecting changes in the pixel value distribution.

Once we evaluate the methods in all datasets, we investigate how their results statistically differ from each other. The reason is to observe if there is an SM that is better than the others. We applied a Friedman test with Nemenyi posthoc with 95% the confidence level for all methods ranks through every dataset regarding the MCC metric. From that, we build a critical difference diagram illustrated in Figure 6.

The result shows that ODIN achieved the best results in more benchmark datasets than the other two SM. However, there is no statistical difference between the SMs. It means that regarding the MCC results, there is no best SM method. Next, we evaluate how much the SM impacts the SUT when it works along with the ML classifier.

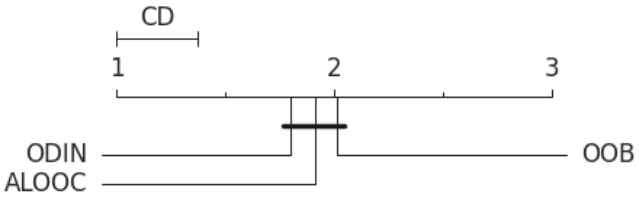


Figure 6: ***Critical difference diagram:*** *no statistical difference between the methods.*

3.2.4.2 Overall impact of the SM

As previously explained, the objective is to evaluate how much the SM impacts the SUT when it works along with the ML classifier. Table 5 shows the overall SM's impact in the SUT when using GTSRB or CIFAR-10 regarding ID data. The results are expressed as MCC values. To evaluate the SM, we also use a percentage of relative change showing how much better/worse it performed compared to the baseline (ML alone).

Table 5: ***MCC values for SUT with/without SM for GTSRB or CIFAR-10 as ID dataset.*** *Results indicate a huge negative impact of SM in the system.*

Method	GTSRB	CIFAR-10
ML alone	**0.96**	**0.74**
ML + ALOOC	0.51	0.53
ML + OOB	0.64	0.61
ML + ODIN	0.50	0.64

In the GTSRB dataset, the best values for the ALOOC method were obtained when using the ADAM optimizer. In contrast, the OOB method obtained the best results when using 3 clusters, ISOMAP, and an enlargement factor of 0.1. As mentioned earlier, for ODIN, the threshold was set to 0.0237.

For the CIFAR-10 dataset, the best optimizer for ALOOC was ADAM. For OOB, the best parameters were: 5 clusters, ISOMAP, and an enlargement factor of 0.35. For ODIN, we set the threshold to 0.1007.

As can be noted, all SM methods perform worse than ML alone when exposed to ID data. This result is expected due to the generalization power of ML models. However, a reliable SM is also one that can perform well when exposed to ID data, avoiding an ML misclassification or simply not raising a false alarm hindering a correct ML classification.

These results show the best SM's performance was obtained by the outside-of-the-box and ODIN, on GTSRB and CIFAR-10, respectively. Results indicate that the SM performed consistently better than a hypothetical random classifier ($MCC = 0$). However, they also show a significant gap between the performance of the ML alone and the SM for ID data, especially in the GTSRB which has four times more classes than CIFAR-10. This result is interesting since the tested monitors use information from the ML model and build their hypothesis over ID data.

3.2.4.3 Time and memory overhead induced by SMs

We show in Table 6 an example of the performance for ALOOC, OOB, and ODIN, respectively, in the three benchmark datasets for novelty class detection. The values are expressed in seconds

and represent the average time spent on the tasks of classification and monitoring on a single image. We used a computer with a Processor Intel(R) Core(TM) i7-9850H CPU @ 2.60GHz, and 32GB of memory.

Worth mentioning that we can have slight differences between the prediction time in the three methods since they use different deep learning libraries. ALOOC uses Keras, OOB uses Tensorflow, and ODIN uses PyTorch. Besides, the SUT time also contains the remaining time from other small processes involved in the experiment, so the sum between the prediction and monitoring time is lower than the SUT time.

Table 6: ***Time impact of SM per instance in seconds for novelty class detection.*** *Monitors are responsible for more than 96% of the total processing time.*

Method	ML	SM	SUT
ALOOC	0.0070 (3.0%)	0.2217 (96.8%)	0.2288
OOB	0.0021 (3.9%)	0.0529 (96.0%)	0.0551
ODIN	0.0007 (3.0%)	0.0246 (96.9%)	0.0254

According to the results, ODIN is the fastest method. Except when using OOB with ISOMAP, all three methods needed no more than 0.07 seconds to spot OOD data. Such process time can be considered fast enough to be applied at runtime. Besides, they can be optimized, potentially reducing this time.

However, the monitoring task is always slower than the prediction and can be responsible for 96% of the necessary time for a SUT to do the task. Since memory is also an important constraint, especially on embedded systems, we show in Table 7 the memory efficiency of the methods when performing a novelty class detection over the GTSRB dataset.

Table 7: ***Memory size of ML, and SM in MB.*** *Monitors based on autoencoders may introduce a huge memory overhead to the SUT.*

Method	ML	SM
ALOOC	3.8	98.9
OOB	4.3	6.4
ODIN	2	1.5

As can be seen, ALOOC needs a considerable amount of memory or disk space due to the necessity of developing one monitor for each class. For instance, even though ALOOC needs just 2.3MB per class, it is necessary almost 100MB to monitor all 43 classes contained in the GTSRB dataset. For OOB, it is also necessary to build one monitor for each class. However, the amount of memory needed is not huge since it uses just some stored arrays to make the boxes. For ODIN, it is not necessary to build one SM for each class. Moreover, since the algorithm needs to be applied during the training phase to collect the thresholds for the confidence values, it can use just a tiny amount of memory to do the inspection.

3.2.5 Threats to validity

In order to analyze and mitigate threats to the validity of the results, we present below a summary of arguments for external and internal validation.

- *External validity*: All tested SMs were already compared to other methods in their original papers. They obtained the best results during the comparison. Besides, we also chose three different approaches to monitoring (inputs, intermediate values, outputs) that may cover an acceptable range of SM approaches based on data. However, it is worth mentioning that this study is a first benchmark work. Thus, other approaches may be published with better performances in the future.

- *Internal validity*: What could have led us to the wrong conclusions in our study?

  - *Bad parameters choices for the SM*: aiming at improving the SM performance, we explored additional parameter values than those explored in their original papers. For instance, we applied different dimensionality reduction methods for the outside-the-box. However, even though different dimensionality reduction methods can bring better results, they also introduce a high cost to time and memory performance. Regarding ALOOC and ODIN, we had to choose criteria for the threshold value for considering whether a class is considered out-of-distribution or not. This way, if one applies different criteria for these methods, they will result in different outcomes. However, such criteria had to be deducted since they are not contained in their original papers. Besides, ODIN used OOD data to calibrate their monitor, which can be considered unrealistic. Hence, in this work, all the parameters were chosen using just ID data. It makes the scenario more realistic and harder, which drastically decreases the SMs performance.

  - *Choice of datasets*: even though we chose datasets widely applied in the computer vision literature, the choice of the amount of ID data accessible for building the ML classifier and the SM can influence both. Furthermore, the amount of ID and OOD data in the benchmark datasets also can influence the results. However, we followed traditional ways to divide data (e.g., 80/20 for training and testing). Besides, the authors of OOB [Henzinger et al., 2020b], and ALOOC [Sabokrou et al., 2018] test their methods considering the same dataset for ID and OOD data, which they test novelty detection by training the ML with 9 classes and omitting one, or training with 8 classes and omitting two, and so on. However, we in this work we test novelty detection using an entirely new dataset as OOD data along with a part of ID data (ex: CIFAR-10 + GTSRB).

    Domain of validity: The setting CIFAR10-GTSRB constitutes two different datasets that do not have the same validity domain [Riccio and Tonella, 2020] (i.e., GTSRB is not only out-of-distribution but also out of the validity domain). Thus, the task of detecting OOD data should be less complicated in this scenario. However, the SM methods continued to be inefficient.

3.3 Conclusion

In this work, we proposed a framework for benchmarking safety monitors for ML classifiers. We argue that there is a need for a better framework for benchmark SMs based on data by applying more metrics beyond accuracy and ROC curves. Thus, we proposed a minimal set of measurements that should be considered when benchmarking such solutions.

Besides, we also showed that measuring the performance of these SMs is less straightforward than it seems. We presented two dimensions that need to be investigated: the SM's overall

impact on the SUT; and the detection performance of the SMs. Our approach allowed insights into three categories (e.g., for inputs, intermediate values, and outputs) of data-based monitors.

Our results indicate at least four general takeaways for the current solutions that are solely based on DNN's data:

- The overall accuracy in the detection tends to be as bad as a random classifier due to the high amount of false positives or false negatives, depending on the scenario.
- Such methods tend to negatively affect the system under test since they yield too many false positives interfering with the correct ML decision.
- They need a considerable amount of memory or disk space since a monitor can consume up to 100MB.
- They are not so fast to perform the detection and have a considerable overhead compared to the ML software.

The tested SM was exposed just to novelty detection in their original papers. However, testing these SM with the other four types of OOD data was important to show that:

- *SM based on DNN inputs*: it has the advantage that it does not need to inspect the internals of the DNN. Thus, it can make the monitoring independent of the ML classifier. However, it has the drawback of being biased in the data used during the training.
- *SM based on DNN intermediate values*: it has the advantage of inspecting the ML model as a white box. That is, it allows us to look at the internal values that led the DNN to decide. However, the drawback is that the activation functions are insufficient to provide all information helpfully learned from the DNN.
- *SM based on DNN outputs*: it has the advantage that it decreases and equilibrates the confidence values from the DNN. However, similarly to approaches based on DNN intermediate values, the drawback is that it relies heavily on the performance of the DNN.

The main conclusion is that current data-based monitors do not provide sufficient confidence against all possible threats. However, we still need to assess them as part of a safety-critical function in order to evaluate the safety aspect of the system, we propose, in the next chapter, an evolutionary simulation testing of safety-critical perception systems, which comes integrated into the CARLA simulator. Unlike related works that simulate scenarios that raise failures for specific perception problems such as novelty, we provide an approach that finds the most relevant OOD perturbations that can lead to hazards in safety-critical perception systems.

Chapter 4

Testing SM for ML perception functions at the system level

As explained before, in order to evaluate the safety aspect of the system we need to simulate relevant OOD perturbations that can lead to hazards in the system. Hence, we move from the task of unit testing to the system-level test. As a case study, in this chapter, we move from an image classification problem to a safety-critical system dependent on an object detector model built with a modern deep learning algorithm. Such an experiment is relevant to the academy and industry since object detection models tend to have problems such as ghost detections [Bogdoll et al., 2022], misclassifications [Sun et al., 2021], or even being blind to the presence of new objects [Sabokrou et al., 2018].

To increase ML robustness, practitioners usually apply data augmentation techniques during training or OOD detectors during operation [Sabokrou et al., 2018, Sun et al., 2021, Liang et al., 2018b]. The performance of such methods is usually demonstrated by inspecting the number of false positives and false negatives when exposed to several datasets containing common image perturbations and new objects [Hendrycks and Dietterich, 2019, Secci and Ceccarelli, 2020]. However, simulation is an essential part of testing since not all ML errors lead to hazards, and most of the time, the physics of such autonomous systems need to be taken into account when searching for hazards. Therefore, in this chapter, we present SiMOOD, an approach dedicated to test perception functions built with ML. SiMOOD is integrated within CARLA simulator [Dosovitskiy et al., 2017].

We apply OOD perturbations during simulation instead of using it over static datasets as it is usually done in the literature [Hendrycks and Gimpel, 2016b, Hendrycks and Dietterich, 2019, Shafaei et al., 2019]. However, since each OOD perturbation has several parameters, the number of possible simulations to find a hazard grows exponentially. Moreover, the time to simulate even a tiny part of these simulations grows quickly. To mitigate this problem, SiMOOD uses a two-step approach. As illustrated in Figure 7, first, it performs a unit testing of the ML

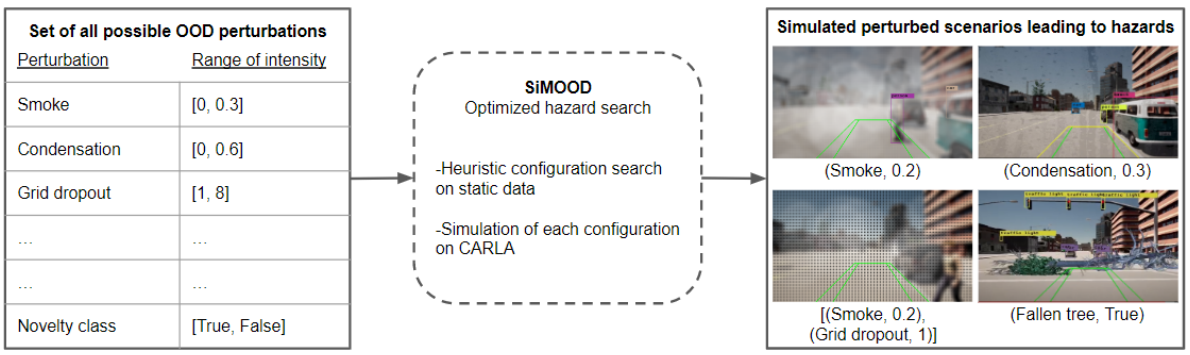


Figure 7: ***SiMOOD simulation flow.*** *After an initial configuration of all possible perturbations, the algorithm searches possible candidates that will lead to hazards during simulation.*

model using a genetic algorithm (GA) on the same dataset applied to train the ML model. It determines which combination of OOD perturbations with their respective intensity levels increases the number of incorrect predictions. Second, the selected perturbations are applied to the simulation to verify if they lead to hazards in the system. That is, SiMOOD takes a set of existing scenarios and injects image perturbations selected by a GA-on-data approach and posteriorly tests them on simulations, proving itself capable of turning safe scenarios into unsafe ones, which can be analyzed in order to improve the safety of the system.

We consider four categories of OOD perturbations: novel classes, distributional shifts, noise, and anomalies. Our main contributions are:

- *Open-source testing approach integrated with CARLA simulator*: SiMOOD apply several relevant OOD perturbations at simulation time. Such perturbations also vary in intensity. Performing such tests yields more realistic results than performing them over static datasets. Besides, SiMOOD is open-source [Ferreira, 2022].
- *Optimized search for hazards*: SiMOOD is an off-the-shelf testing approach for perception functions in a simulated environment. It uses an evolutionary approach to find the minimal OOD perturbations that may lead to hazards during simulation. Due to its GA-on-data approach, SiMOOD decreases 10 times the amount of time required to find a set of hazards in safety-critical tests while considering a huge search space.

This chapter is organized as follows: in Section 4.1, we discuss specific literature about the concepts used in our approach, the simulated OOD data, and existing related works. In Section 4.2, we detail the architecture of our approach and introduce its main functionalities. In Section 4.3, we show the experiments and the insights provided by the results. Finally, in Section 6, we present our final considerations and limitations.

4.1 Background

We focus on simulation-based testing of perception systems composed of ML algorithms, specifically object detectors exposed to out-of-distribution images. Therefore, works that focus on finding safety violations on control [Li et al., 2020], system specifications [Zapridou et al., 2020], error space exploration [Singh et al., 2021], path planning [Arcaini et al., 2021], and new driving scenes generation [O'Kelly et al., 2018] are outside of the scope of this work.

As the first example of related work, Pei et al. [2017] proposes DeepXplore, an automated white-box testing of deep learning systems. The authors argue that DeepXplore can measure code coverage by measuring the neuron coverage of a deep learning algorithm when perturbing the images that are fed into these ML models. Following a similar idea of neuron coverage, Tian et al. [2018] propose DeepTest, an automated testing approach of deep-neural-network-driven autonomous cars. The authors can find several ML failures in a test set in which the ML model should perform a steering angle action correctly. Zhang et al. [2018] push forward the idea of perturbing images for testing by proposing Deeproad: a gan-based metamorphic testing and input validation framework for autonomous driving systems. DeepRoad generates driving scenes with different weather conditions by using GAN along with the corresponding real-world weather scenes. DeepXplore, DeepTest, and DeepRoad test safety-critical perception functions against generated OOD images. However, these works perform tests on datasets, and such offline testing alone cannot guarantee that safety violations will not happen during online testing [Haq et al., 2021]. Therefore, our work also differs from the previous related works.

Regarding related works that also propose simulation tests, Dreossi et al. [2019a] propose to test the safety-critical system when exposed to novel classes. The authors apply a falsification method over images with different light conditions and distances to find regions of uncertainty. After that, they test it against a simulator. Rossolini et al. [2022] test the robustness of semantic segmentation models by applying adversarial colored patches on the simulation and real images. The authors present extensive experiments to validate the proposed attack and defense approaches in real-world scenarios. Boloor et al. [2020] also test and simulate physical adversarial examples that can affect the detection performance for object detection tasks. Boloor et al. [2020] apply black lines on the streets to foul the object detector and force it to go in a different direction. Fahmy et al. [2022] propose a tool to debug DNNs for safety analysis. The tool finds clusters of images with common characteristics that lead to errors by using the information propagated by DNN neurons during the prediction. Finally, Haq et al. [2021] showed that for continuous values, DNN prediction errors not identified by testing DNNs with static datasets can yield many safety violations during online testing. However, the authors also emphasize that such a premise cannot be applied when testing perception-based tasks. Our work goes one step further on this problem by focusing on minimal image perturbations that lead to hazards during simulation.

Our approach differs from Rossolini et al. [2022] since our focus is on the object detection task. Our approach also differs from Boloor et al. [2020], Dreossi et al. [2019a], Fahmy et al. [2022] since these related works are focused on an in-depth analysis of one type of OOD data (e.g., adversarial, novelty, and distributional shift respectively). In contrast, we consider four types of OOD data that are a source of threats for perception systems with ML-based object detectors. We also provide a list of fine-grained intensity perturbations selected over a higher search space. Next, we present an overview of SiMOOD.

4.2 SiMOOD: evolutionary testing Simulation with Out-Of-Distribution images

SiMOOD generates hazardous perturbations in a specific scenario in four types of OOD data: novelty, anomaly, distributional shift, and noise. For novelty class and anomaly perturbations SiMOOD works with binary intensity levels (true/false) and continuous values for distributional shift and noise. For example, for novelty class experiments, SiMOOD can inject *a new object* from the operation domain design in a random point of the scenario, and for anomaly experiments, SiMOOD can inject *a known object* in a way that is not expected to be found in the original dataset. As noted, there is no sense in applying intensity levels for novelty and anomaly perturbations but rather in applying random locations for the novel/anomalous objects in the scene.

For perturbations that can vary with a range of continuous values, SiMOOD needs to generate values that can lead to hazards in the simulation. Since the search space for such values is large, SiMOOD applies an evolutionary search to find the values that have more chance of provoking ML failures that lead to hazards in the simulation.

As illustrated in Figure 8, SiMOOD is divided into three parts: generation, simulation, and evaluation. Thus, during the generation, SiMOOD applies an evolutionary search to find the n-most relevant OOD combinations. Each combination represents a vector containing m tuples in the format (perturbation, intensity). Each perturbation has a range of intensity values. For instance, a single combination of size m=2 can be formed by [(Gaussian noise, 0.2), (Blur, 3)].

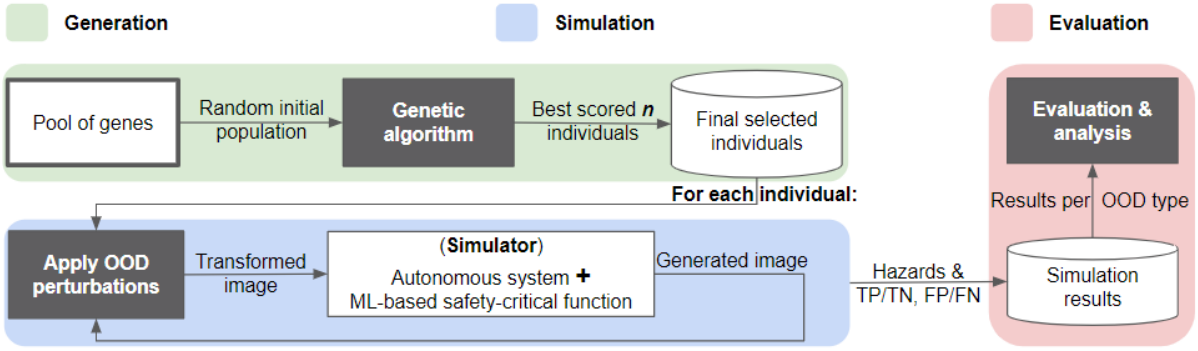


Figure 8: ***SiMOOD overview:*** *generation, simulation, and evaluation phases.*

Thus, we introduce the following definitions:

- **Genes**: A gene is represented as an OOD perturbation with its respective intensity value (e.g., Gaussian noise, 0.2). Each OOD perturbation varies in intensity from *0* (no effect) to L (severe perturbation but still interpretable).
- **Individual**: an individual is composed of a number of genes. For instance, an individual of size m=*2* is represented by a vector of m tuples: [(Gaussian noise, 0.2), (Blur, 1)].
- **Population**: it is a set of individuals. An initial population of size n means a generation of n-individuals.
- **Genetic algorithm iterations**: since the number of combinations and their parameters is large, the space search is similarly large. Therefore, in the first iteration, the GA randomly selects an initial population of size n.

After this generation step, the n-most relevant individuals selected by the GA are iteratively simulated in a safety-critical system. Each combination is applied to each frame. The transformed image is exposed to the ML-based safety-critical function, and the results are stored for future analysis. Next, we detail each part of SiMOOD.

4.2.1 Generation

During this part, SiMOOD performs the task of finding suitable configurations (e.g., combinations of OOD perturbations) using a genetic algorithm (GA) approach as illustrated in Figure 9. A GA comprises an initial population randomly selected from the original population. A population represents a group of individuals. Each individual comprises a group of genes (or a group of *parameters*). The GA *selects* the best individuals based on a *fitness score* outputted from a *fitness function*. After it, the k best-selected individuals generate k new individuals for the next generation. For each pair of selected individuals, a *crossover* point is chosen randomly from within the genes. Finally, a *mutation* can occur to maintain diversity within the population.

These perturbations are combinations of *distributional-shift* and *noise* perturbations. For example, Gaussian noise with a blur effect, or Spackle noise in an environment with heavy smoke. Since the ML always gives wrong predictions to new classes, the novelty class category is not applied in the GA but rather as an option when performing a simulation with our approach. Next, we translate the GA terms to our task as illustrated in Figure 9:

- **Initial population** it is randomly generated and represents a subset of all possible combinations of OOD transformations.

- **Apply perturbations** it sequentially applies a combination of perturbations over an image following an order of occurrence of these perturbations inside of an individual. That is, an individual containing [(Blur, 0.1), (Smoke, 0.3)] leads to a blur transformation followed by a smoke transformation.

- **Fitness evaluation** each set of perturbations is applied to the same dataset used to train the ML model. Hence, we have n transformed datasets that are exposed to a fitness function. The fitness function uses the same ML model applied later in the simulation as part of the final fitness score. Thus, the fitness function calculates all true/false positives/negatives yielded by the ML model when exposed to these transformed datasets. Any classical ML metric value α between 0 and 1 can be used (e.g., mAP, false positive/negative rates (FPR) regarding the detection task, precision and recall).

  The **fitness score** f is a sum of α and the normalized vector of perturbation intensities η multiplied by a smoothness term ω, that is, $f = \alpha + (\eta * \omega),\ \omega \in \mathbb{R}[0, 1]$. These terms are added to α to force the GA to reward perturbations that lead to hazards but with a minimum amount of intensity. Worth mentioning that if α is based on measuring the ML model errors (ex: mean average error) instead of the model correct predictions, this regularization term is decreased by α instead (e.g., $|(\eta * \omega) - \alpha|$).

  To calculate η we first normalize the vector of perturbation intensities $v = v_1, ..., v_m$ through the formula $(val - min\_val)/(max\_val - min\_val)$, which val means the intensity value of a particular OOD perturbation, min_val and max_val represents using the minimum (e.g., 0, or no effect), and the maximum value of intensity for a particular OOD perturbation. Hence, after applying the above formula for each gene of an individual of size m, we obtain the final value by $\eta = \frac{1}{m} \sum_{i=1}^{m} v_i$.

  Regarding ω, it is intended to penalize a high intensity η over the ML metric α. The value of ω can be increased if one wants to give more importance to η, or decreased otherwise. Besides, since our focus is to find different perturbations rather than find the highest intensity levels, we do not allow $\omega > 1$.

- **Selection** once all fitness scores are collected, the algorithm selects k best OOD perturbations that had the most relevant fitness scores. In this work, we choose to work with an odd number of individuals (e.g., k=2) since it is a simple premise of the GA, which performs a crossover between the best pairs of the generation. This criteria is also generic and depends on the objective of the test. In our case, we want to find the minimal OOD perturbations that have a high probability of leading to hazards. Hence, the selection mechanism will choose the one leading to the worst ML performances. For instance, if the fitness score is based on accuracy, SiMOOD selects k individuals that led to the worst results using minimal perturbation intensities.

- **Crossover** the selected individuals k generates k new individuals by exchanging their genes.

- **Mutation** it might occur in the resulting pair of individuals by changing a specific gene at an arbitrary point. Unlike the crossover, the mutation does not always happen and has a low probability of occurrence. The mutation randomly chooses a gene to replace and

assigns a new random value to it. In our approach, to avoid too many repeated individuals, a mutation also occurs when there is a crossover between equal individuals.

- **Population update** the new individuals are added to the current population, their fitness scores are evaluated and added to the fitness scores list of the current population. After that, the algorithm selects the best k individuals for the next generation and excludes the worst k ones. The process repeats until the desired number of generations is reached.

Finally, at the end of the process, SiMOOD outputs the resulting population containing the final selected individuals. That is the combinations that are candidates of being a source of hazards during simulation. However, these possible hazards cannot be confirmed until we perform the simulation tests. Next, we explain how we perform these simulations.

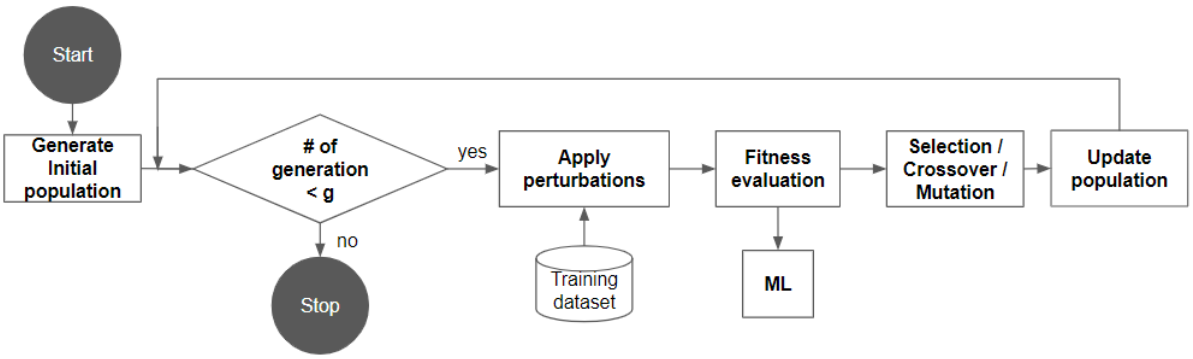


Figure 9: ***Genetic algorithm approach:*** *searching for relevant OOD perturbations.*

4.2.2 Simulation

Following the same term given in the previous subsection, in this part, we take the relevant individuals selected from the previous step and apply them to each frame of the simulation.

During simulation, we collect the ground truth from the original image provided by CARLA without perturbation, and we dynamically generate the object's bounding boxes from the simulator data, adapting the method developed in [Adib, 2022]. This method generates 2D bounding boxes from the projected 3D bounding box of the *visible* objects in the camera image. It uses a distance filter, an angle filter, an occlusion filter, and instance segmentation of the LiDAR sensor provided by CARLA. Besides, to be as lightweight as possible, we adapted this module to generate the ground truth of the objects of interest in the safety scenario. For instance, in an advanced emergency braking scenario, one needs just the ground truth of the objects that interact with the ego-vehicle, such as cars and pedestrians, instead of checking for all other objects in the scene. All the ML model and safety results are stored for future analysis at the end of the simulation.

4.2.3 Evaluation

Besides simulation metrics such as processing time, and memory, we evaluate the safety-critical function (hazards) and the ML model (ML metrics). We use the number of hazards during simulation for the safety metrics. Here, we consider two types of hazards: a) accident (e.g.,

car hits pedestrian); and b) dangerous stop (e.g., the emergency brake is activated without necessity). For the ML metrics, we use the *mean average precision (mAP)* of the bounding boxes generated by the ML model [Cartucho et al., 2018]. It is a standard metric applied to evaluate object detectors. Next, we detail the experiments with SiMOOD.

4.3 Experiments and results

We evaluate SiMOOD capacity to find a suitable choice of minimal perturbations that lead to hazards in CARLA simulations. We choose a safety-critical scenario in which a car uses an emergency braking system equipped with an ML-based object detection model (e.g., YOLO). In this scenario, a pedestrian crosses the street coming behind other objects. The ML model is responsible for detecting all objects that enter a safety-critical region in front of the ego-vehicle. Regarding the safety-critical region, the emergency braking system reacts to objects of interest that enter the region, in our case, the pedestrians. The scenario is configured with CARLA scenario runner [NVidia, 2022].

4.3.1 Experiment settings

Regarding the object detector algorithm, we use one of the most applied: YOLO v6 [Jocher et al., 2021], a model based on stacked convolutional neural networks (CNN). The ML model was trained on the COCO dataset [Lin et al., 2014b] with 10% of augmented data. We use the same augmentation techniques for generating noise and distributional shift perturbations in our approach. We also used data perturbations default in the YOLOv6 library, such as geometric ones.

We applied 15 types of OOD perturbations presented in [Secci and Ceccarelli, 2020, Buslaev et al., 2020], each one with its own levels of intensity ("no effect" included), resulting in 175 levels of intensity.

Table 8: ***OOD perturbation parameters.*** *15 types of OOD transformation with its levels of intensity.*

OOD perturbation	**Levels of intensity**
Shifted pixels	11
Gaussian noise	8
Gaussian blur	24
Grid dropout	9
Coarse dropout	31
Channel dropout	3
Snow	6
Broken lens	11
Dirty	11
Condensation	11
Sun flare	11
Brightness	11
Contrast	11
Rain	6
Smoke	11

For novelty class experiments, we inject a fallen tree on a random road point since it is a familiar object in the operational design domain, but the ML model was not trained with this object in the training set. For anomaly experiments, we inject a vehicle overturned on a random point of the road since it is a known object to the ML model but in a way that is not expected to be found in the original dataset. More details about all OOD perturbations can be found in our repository [Ferreira, 2022].

Regarding the GA parameters, each individual has size $m = 2$. It is worth mentioning that one can change the size of the individual to $m = 1$ in case to find just the single perturbations that lead to hazards instead of a combination of them. Besides, the number of k selected individuals per generation is also fixed in $k = 2$, that is, we select the best pair of individuals per generation, replacing the worst ones. The mAP metric is applied as the ML metric. Therefore, the fitness score includes the worst mAP obtained from a given pair of perturbations.

Regarding the crossover and mutation probabilities, we follow the recommendations of Yang [2014] that showed from empirical results and theoretical studies that it is a good choice to set a relatively higher probability for crossover (e.g., in the range of 0.6 to 0.99). In contrast, the mutation probability can be very low (e.g., around 0.001 to 0.1). Therefore, we set our crossover probability as 0.99 and the mutation as 0.1. Finally, regarding the other GA parameters: a number of generations, population size, and smoothness term ω were chosen after performing sensitivity analysis experiments.

4.3.2 Robustness of SiMOOD regarding its parameters

This subsection has the objective to measure the robustness of SiMOOD regarding its parameters rather than performing a hyper-parameter optimization analysis. Moreover, for this analysis, we prioritize finding hazards with a reasonable amount of different perturbations rather than finding a huge number of hazards with repeated perturbations. That is, finding more hazards while having more diversity indicates better quality results.

We vary the number of generations g between $[10, 20, 30, 50]$, the initial population size n between $[10, 20, 30, 50]$, (each individual has 2 genes), and we also vary the value of ω between $[0.01, 0.1, 0.25, 0.5, 0.66, 0.75, 0.99]$. Next, we perform a diversity analysis, and a hazard analysis when varying the parameters of SiMOOD.

4.3.2.1 Diversity analysis on the influence of ω

The objective is to analyze how many unique individuals are generated when we change the smoothness term ω across the different amount of generations and population sizes. More unique individuals indicate more diversity in the results. Hence, as illustrated in Figure 10, the best value for better diversity tends to be $\omega = 0.5$. However, SiMOOD tends to be robust regarding this parameter, since we observe small but steady variations in the diversity in which $\omega \leq 0.5$ leads to slightly better results.

It is worth mentioning that in the next analysis, we keep ω with the best value. The reason is that the amount of time to perform all simulation tests with different values of ω increases quite fast. That is, the simple range of values chosen for these analyses leads to 3080 different individuals, which would require 6,160 minutes of graphical simulation.

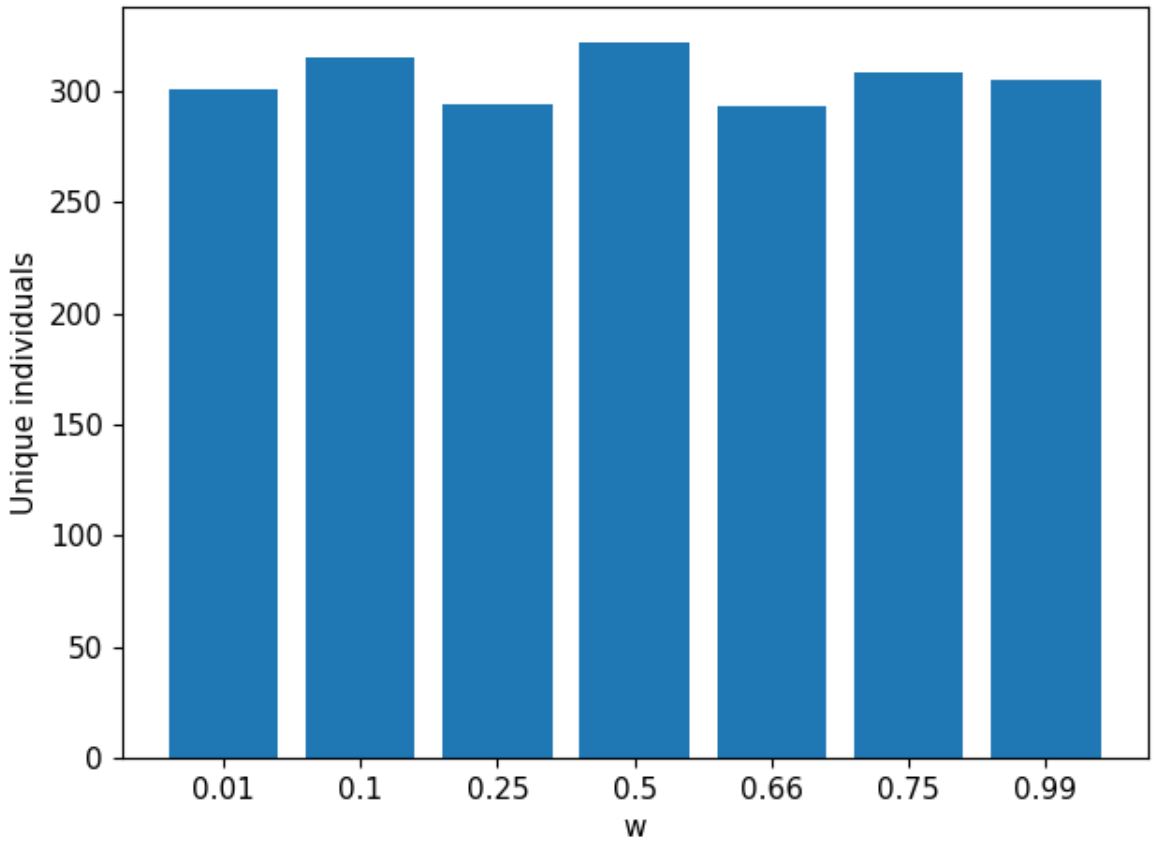


Figure 10: ***Diversity analysis in the SiMOOD generation.*** *Unique individuals per ω, generated across all variations of generations and population size (440 individuals).*

4.3.2.2 Hazard analysis on the influence of the population size and the number of generations when $\omega = 0.5$

Table 9 shows the number of unique genes, while Table 10 shows the number of hazards posteriorly found in the simulation. The best parameter values are the ones that lead to higher values of unique genes (e.g., more diversity) and hazards. The best values of each column are marked with (*).

Table 9: ***Number of unique genes in the selected population.*** *SiMOOD generates between 12% and 59% of diversity.*

Generations	**Population size**			
	10	**20**	**30**	**50**
10	7 (35%)	16 (40%)	34 (57%)*	59 (59%)*
20	8 (40%)*	20 (50%)*	28 (47%)	48 (48%)
30	6 (30%)	5 (12%)	26 (43%)	42 (42%)
50	6 (30%)	7 (17%)	19 (31%)	32 (32%)

Tables 9 and 10 show that the number of unique genes increases gradually along with the

Table 10: ***Number of hazards.*** *SiMOOD was capable of uncovering hazards in the simulation up to 100% of the time.*

Generations	Population size			
	10	**20**	**30**	**50**
10	9 (90%)	11 (55%)	12 (40%)	12 (24%)
20	6 (60%)	20 (100%)*	23 (77%)	21 (42%)
30	10 (100%)*	20 (100%)*	30 (100%)*	25 (50%)
50	10 (100%)*	20 (100%)*	15 (50%)	23 (46%)

population size when performed over a low number of generations, but the number of hazards seems to be steady. Considering the best results in Table 9 and 10, it seems that SiMOOD achieve a good compromise between diversity and hazards when performing with 20 generations g and population size $n = 20$. Besides, the number of predominant genes (e.g., repeated genes) increases with the number of generations. It indicates that SiMOOD does not need too many generations to find a good balance between diversity and uncovered hazards.

Regarding the variability of the experiments, we also run SiMOOD 10 times using the configuration that achieved the best values in both Tables (generation=20, population=20). Such experiments yielded standard deviation $\sigma = 2.5$ regarding the number of unique genes, and $\sigma = 5.5$ regarding the number of hazards, which is acceptable for our experiments.

4.3.3 Results and analysis

In this subsection, we show the results of two main experiments: single OOD perturbations that lead to hazards; and combinations of OOD perturbations that lead to hazards. These experiments were performed with the best parameters presented in the previous subsection: 20 generations g, population size $n = 20$, and $\omega = 0.5$.

4.3.3.1 Hazards provoked by single OOD perturbations

Figure 11 shows three examples. The first sub-figure resulted from a vehicle crash due to a fallen tree not detected by the ML model. Since the safety-critical system depends on the ML model to "see" objects in the safety-critical region in front of the ego-vehicle, it does not trigger the brake action, and the vehicle collides with the tree. In the second sub-figure, since the emergency braking system has the rule to react when a bounding box of a pedestrian intersects a warning region, the emergency braking system stops the vehicle in the middle of the street due to a "ghost" pedestrian detected by the ML model. Such a false detection was provoked by condensed water on the camera lens. Finally, in Figure 5c), a new accident occurs; this time, the vehicle does not avoid a crash with a pedestrian not detected by the ML model when exposed to heavy smoke in the environment. Even though the ML model was capable of detecting the pedestrian, the detection was not made at the right time to avoid a crash after the brake action was triggered. This case also shows the importance of testing on simulations instead of testing on static datasets.

4.3.3.2 Hazards provoked by combined OOD perturbations

Figure 12 shows an example of how the combination of different OOD perturbations can uncover new hazards. In Figure 6c), we show a combination (smoke, 3) and (grid dropout, 1) found by

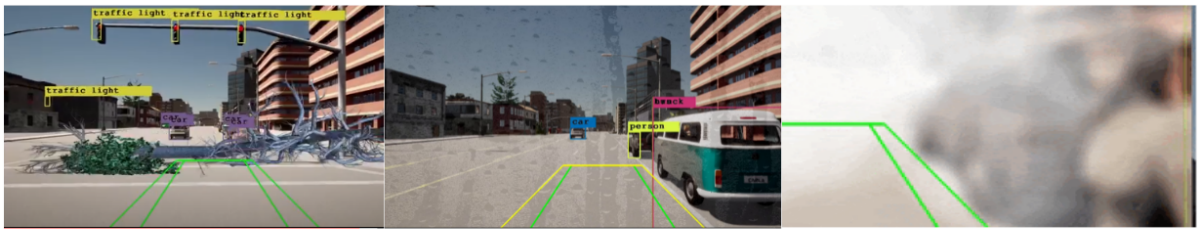


Figure 11: ***Hazards uncovered by applying single OOD perturbations.*** *a) An accident due to a unknown object (tree) not detected by the ML model;* ***b)*** *A dangerous stop due to a false detection (ghost pedestrian) provoked by condensed water on the camera lens;* ***c)*** *An accident due to a known object (pedestrian) not detected by the ML model when exposed to heavy smoke in the environment.*

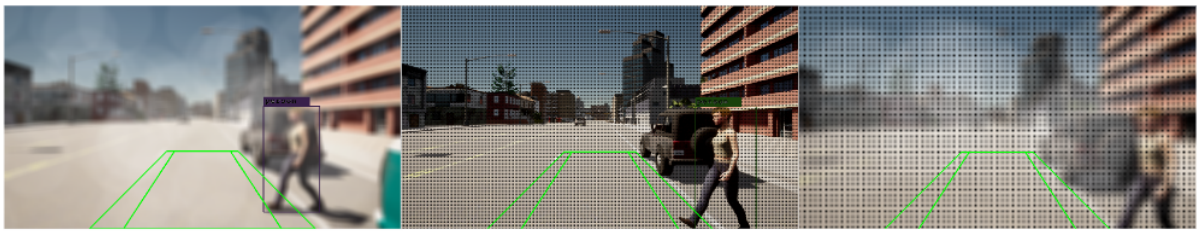


Figure 12: ***Hazards uncovered by combining two types of OOD perturbations.*** *a) YOLO detects a pedestrian in an environment with light smoke (intensity 0.25);* ***b)*** *YOLO detects a pedestrian despite a light grid dropout failure on the camera sensor (intensity 2);* ***c)*** *A collision happens due to a failure of YOLO on detecting a pedestrian when both conditions happen at the same time even with smoke transformation having a lower intensity than before.*

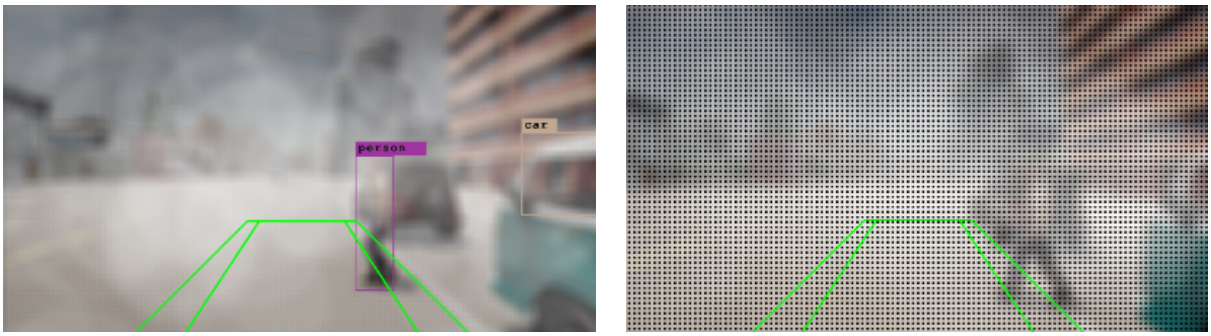
SiMOOD that led to a collision between the vehicle and the pedestrian due to the failure of YOLO in detecting the pedestrian, leading to incorrect behavior of the emergency braking system. However, when one of these combinations happens alone (sub-figures a) and b)), the ML model can correctly detect the pedestrian, making the emergency braking system correctly trigger the brakes.

Simple combinations such as a sensor failure in a moment with a small amount of smoke or haze during runtime can be enough to provoke a hazard. Moreover, as illustrated in Figure 13, the order of the perturbations also matters. The reason is that the same perturbations combined in a different order produce subtle differences in the image outcome, which can be enough to hinder the performance of ML models. Next, we analyze the required processing time and memory to use SiMOOD.

4.3.3.3 Processing time and memory usage

the experiments were performed in an Intel(R) i5-10500 CPU @ 3.10GHz, with 32GB of memory, six cores, and a GPU (Quadro RTX 4000). Below, we show time and memory analysis for the two parts of the approach: *a) generation, and b) simulation.*

a) Generation: SiMOOD can reduce the time necessary for finding hazards during the simulation due to its approach of performing the GA algorithm on data instead of applying it directly to the simulation. In our experiments, there are 175 possible perturbations T with



(a) *Grid dropout (1) + smoke (0.3).* **(b)** *Smoke (0.3) + grid dropout (1).*

Figure 13: ***Same OOD perturbations but in different order produce different image outcomes, which may uncover new hazards.***
a) *YOLO detects a pedestrian in the scenario with the OOD combination grid dropout + smoke.* ***b)*** *YOLO does not detect a pedestrian in the same scenario but with OOD perturbations in inverted order (e.g., smoke + grid dropout).*

their respective intensity levels, and the number of simultaneous perturbations for each combination (individual) is $m = 2$. Since the order of transformations matters, the set of possible combinations is given by $\frac{175!}{2!\times(175-2)!\times 2} = 30,450$ possible combinations.

Besides, applying the GA directly into the simulation is suitable for the classes exposed during the test (in our case, pedestrians and cars). On the other hand, SiMOOD uses a GA-on-data approach, which considers all 80 classes in the COCO dataset. It allows SiMOOD to be extended to other types of scenarios and objects on it. Below, we provide a theoretical time analysis considering our GA and its parameters.

First, the GA starts with a time complexity of $O(nt)$ which n is the size of the initial population, and t is the amount of time to compute the fitness function. Second, for each subsequent generation, the GA will transform the k best individuals of the population with crossover and mutation operators, and posteriorly compute the fitness function, which adds $k * t * g$ to the processing time. Finally, the theoretical processing time when performing the GA is given by $nt + ktg$, and its required memory space is given by $m * k$.

Considering the aforementioned processing time formula, we can calculate the expected amount of time, *in seconds*, to find a population of hazards when *performing GA directly on the simulation*. Therefore, if we chose a simple configuration with an initial population $n = 20$ (0.0001% of the entire population), with 20 generations g, and the amount of time to render our scenario ($t = 120$), and the number of selected individuals per generation $k = 2$, then the expected processing time is given by $20 * 120 + 2 * 120 * 20 = 7,200$ seconds (*120 minutes*).

Regarding *performing the GA-on-data-first approach*, the time t to compute a fitness function is replaced by the time spent generating the datasets (5 seconds per dataset) + the time applying the fitness function over each one of the generated datasets (7 seconds per dataset). That is, $t = 12$. Hence, the amount of time to perform GA over data is $20 * 12 + 2 * 12 * 20 = 720$ seconds. Finally, we add the time spent when running the simulation with the final population, that is, $120 * 20$. Therefore the total amount of time is given by $720 + 120 * 20 = 960$ seconds (*16 minutes*).

Besides, even SiMOOD reduced the processing time almost 10 times the GA-approach processing time, this difference could be even higher if we use a bit more complex scenarios such as longer scenarios with more objects to render, or if more than one different scenario needs to be

tested.

b) Simulation: Table 11 shows the processing time (seconds) and the memory usage (MB) *when performing the simulation* with and without using SiMOOD.

Table 11: ***Comparison of processing time and memory.*** *Amount of memory added to the simulation is significant due to image transformations performed by the framework.*

Time	**Time (with SiMOOD)**	**Overhead**
101.94	123.10	20.75%
Memory	**Memory (with SiMOOD)**	**Overhead**
3975.58	6708.09	68.73%

It is worth mentioning that the number of perturbations does not increase the average simulation time or the total average memory used in the simulation. All perturbation functions are performed over a matrix with a fixed image size. However, since SiMOOD applies perturbations on high-resolution images (1280x720), it is necessary an extra amount of memory (2.7 GB) to perform the task without severely impacting the simulation speed. Therefore, SiMOOD requires a computer with 8GB of memory, which is quite normal for the current computational requirements. Moreover, SiMOOD can be optimized to perform better processing and consume less memory by performing parallelization and data compression.

4.3.4 Threats to validity

In order to analyze and mitigate threats to the validity of the results, we present below a summary of arguments for external and internal validation.

4.3.4.1 External validity

Regarding the simulated sensor failures and weather conditions, there is always a gap between simulation and reality Stocco et al. [2021]. However, the tested OOD perturbations were already applied in several other papers from the literature with different levels of intensity.

Regarding the chosen simulator to perform our experiments, it is worth mentioning that different simulators can yield different outcomes. However, the CARLA simulator is an open-source simulator widely applied in the industry and in the literature. Besides, our approach can be extended to other simulators.

4.3.4.2 Internal validity

What could have led us to the wrong conclusions in our study?

Variability of simulated safety-critical scenarios we tested our approach over a single but relevant safety-critical scenario. However, a safety-critical scenario can greatly vary depending on the expert's needs. Therefore, it is not feasible to guarantee in a theoretical way that our approach will provide similar results across any safety-critical scenario. However, it is worth mentioning that this study is the first approach to this challenging problem. Thus, other scenarios can be added and tested in the future since our open-source code can easily be adapted/extended to new scenarios.

Choice of OOD perturbation levels despite the choice of perturbations on images capable of being interpreted by human eyes, the amount of intensity considered valid is a choice of the experts. Thus, the range of intensity over the perturbations can change the outcomes in different simulated scenarios.

Choice of parameters of the GA different values for the parameters of the GA algorithm (e.g., number of selected candidates per generation, number of newly generated individuals, crossover/mutation probability) can lead to different outcomes in both diversity and number of hazards in the simulation. However, we followed a traditional way to develop the GA approach and we also followed the recommendations from the literature [Mirjalili, 2019].

Choice of datasets for the ML model and for the GA even though we chose a dataset widely applied in the computer vision literature, the choice of the amount of data accessible for training/validating and testing the ML model, and performing the GA, can influence the fitness score and maybe also change the selected perturbations. However, we followed traditional ways to divide data (e.g., 80/20 for training and testing).

4.4 Conclusion

In this work, we proposed SiMOOD, an evolutionary simulation testing for ML-based perception systems. SiMOOD is open-source [Ferreira, 2022] and is already integrated with CARLA, an open-source simulator for autonomous vehicles. SiMOOD was capable of finding out-of-distribution image perturbations that lead to hazards during simulation. Moreover, it was possible without performing such optimization algorithms loops directly into the simulation. Such a strategy is able to save, up to 10 times, the amount of time needed to find a set of hazards in safety-critical simulation tests.

We also showed that combinations of perturbations could expose different hazards. Moreover, these perturbations are not commutative. Hence, the order of occurrence of such perturbations can also uncover new hazards in the simulation.

The main difference in findings between this chapter and Chapter 3 is that when benchmarking unit tests the aspect of hazards is reduced to the ML errors regarding the classification, being difficult to assess the outcomes of these hazards in the system usage, since there is no real-time simulated data. However, in this chapter, we went further and proposed a framework that finds hazards during the simulation. Both approaches could be combined into a pipeline for assessing the safety of ML-based functions, starting by evaluating the ML model as a single component and after, evaluating the entire ML-based component in a safety-critical system.

A limitation of this work is the additional memory usage added to the simulation. However, this is the first version of SiMOOD, and it will be improved in the subsequent versions since it is open to the community. As a next step, we intend to add a scenario generation process similar to Abdessalem et al. [2018], capable of varying the scenarios, and parameters. Adding such a generation process to SiMOOD is important to produce a more robust analysis.

The research presented in Chapters 3 and 4 covered the test and evaluation of SM at the unitary and system levels. We showed that new SM methods should be proposed in order to tackle the high amount of false positives and negatives in monitor detection. Therefore, in the next chapter, we propose a hybrid SM called SENA. Since our research showed that not all OOD data leads to failures, we focus on detecting when the ML is likely to fail. That is, instead

of trying to detect when an image is OOD data or a hard-to-predict ID data (which *will not always result in a wrong output*), we focus on detecting if the combination incoming image+ML model is an "error-prune" combination (which *will most of the time result in a wrong output*). Besides, we use information from both correct and incorrect data during ML model training.

Chapter 5

SENA: Similarity-based Error-checking of Neural Activations

In Chapter 3, state-of-the-art SMs were empirically demonstrated to yield a considerable number of false negatives (i.e., prediction errors not detected by the monitor) and false positives (i.e., correct predictions rejected by the monitor). On the one hand, false positives have a negative impact on the performance of the ML model for ID data. One of the main reasons for the high number of false positives is that current monitors treat OOD data as data that the ML model should avoid. Hence, such monitors tend to be activated for all OOD data. However, not all OOD data lead to a failure in the ML model [Guérin et al., 2023]. In addition, recent works showed that OOD detectors solely based on uncertainty [Schwaiger et al., 2020], generative models [Zhang et al., 2021] and its densities [Le Lan and Dinh, 2021], cannot safely guarantee that the OOD detection is correct, except for specific combinations of datasets. On the other hand, false negatives tend to decrease the safety of the ML system as unsafe data instances are not detected. They are mostly due to the fact that ML models still have to deal with possible wrong predictions for ID data as well, and recent works theoretically demonstrated that wrong ML predictions for both ID and OOD data may be linked to the same phenomena of model misestimation problem [Zhang et al., 2021].

All previous empirical and theoretical works seem to point to the importance of approaches that can detect when the ML model can correctly deal with the data, independently if it is ID or OOD. Therefore, in this work, we propose SENA, a data-based monitor focused on detecting unreliable ML model predictions. The main idea is that instead of trying to detect when an image is OOD, which *will not always result in a wrong output*, we focus on detecting if the prediction from the ML model is not reliable, which *will most of the time result in a wrong output*, independently of whether it is ID or OOD. In this chapter, we propose preliminary work to validate that this approach is worthy of investigation. Therefore, we do not investigate the optimization of our monitor, which will be future work.

To detect when a prediction is reliable or not, SENA differs from related works in two main aspects: 1) It uses information from both true positives and false negatives collected during the ML model training, which allows SENA to automatically choose thresholds that are better adapted for a specific dataset. 2) It uses a statistical concept known as core support extraction [Ferreira et al., 2019] combined with a simple distance metric, which makes SENA less sensitive to ID outliers. The first results show that by applying the aforementioned approaches an SM can achieve results comparable to state-of-the-art solutions without requiring any prior OOD information, and without hyperparameter tuning (using only default values). Besides, the code is publicly available for easy reproducibility [1].

Therefore, this chapter is organized as follows: In Section 5.1, we detail our approach by showing how our SM is built with training data, how it automatically chooses thresholds, and how it performs the ML monitoring at runtime. In Section 5.2, we show the experiments, and

[1] https://github.com/raulsenaferreira/SENA

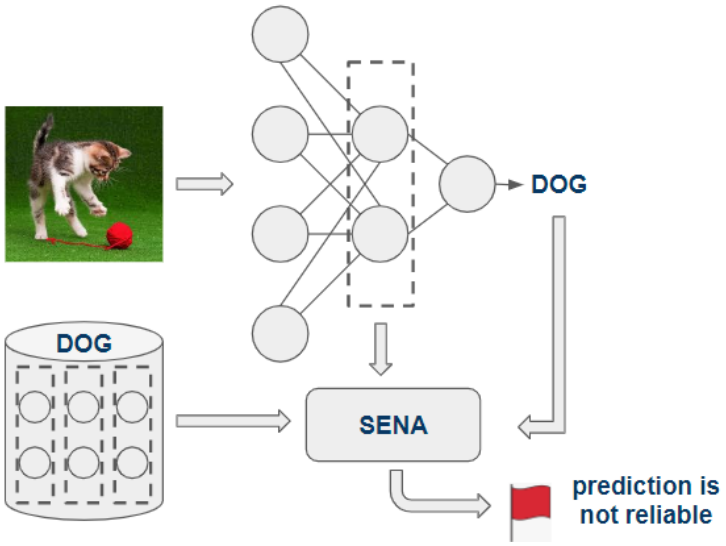


Figure 14: ***Similarity-based Error-checking of Neural Activations.*** *SENA checks neural activations that may lead to erroneous ML model predictions by comparing them to representative neural activations from the training set.*

the analysis of our results using classification metrics and safety ones. Finally, in Section 5.3, we present our final considerations, limitations, and possible improvements to our method.

5.1 Similarity-based Error-checking of Neural Activations

SM approaches usually monitor an ML model at runtime through one of three types of source of information: 1) the input (e.g., image [Sabokrou et al., 2018]); 2) the internal activation values (e.g., model logits [Jiang et al., 2018], activation patterns [Henzinger et al., 2020b]); 3) the outputs from the last layer (e.g., confidence values [Hsu et al., 2020]).

As illustrated in Figure 14, SENA uses the internal activation values (e.g., activation functions) of the ML model combined with its prediction as its source of information to monitor a specific input image at runtime. The verification is done by checking the similarity between the neural activations of an incoming input and a set of representative neural activations recorded during training. SENA uses information from both true positive and false negative examples collected during training to verify if a prediction is reliable or not. The motivation for using correct and incorrect examples from the training dataset is to help the monitor to deal with not only OOD data but also incorrectly classified ID data.

SENA is built as a one-class classifier, such as one-class support vector machines [Li et al., 2003], class-reconstruction-based methods [Sabokrou et al., 2018], or methods based on class-activation-function patterns [Henzinger et al., 2020b]. Therefore, an independent monitor is produced for each output class of the monitored model, and the ML prediction given at runtime is used to choose which monitor will be used. In the remainder of this section, we explain the processes to build a SENA monitor, and how to use it at runtime.

5.1.1 Monitor building

In this section, we explain how to build a SENA monitor for a specific class c. The complete SENA monitor is composed of n such monitors, where n is the number of output classes of the

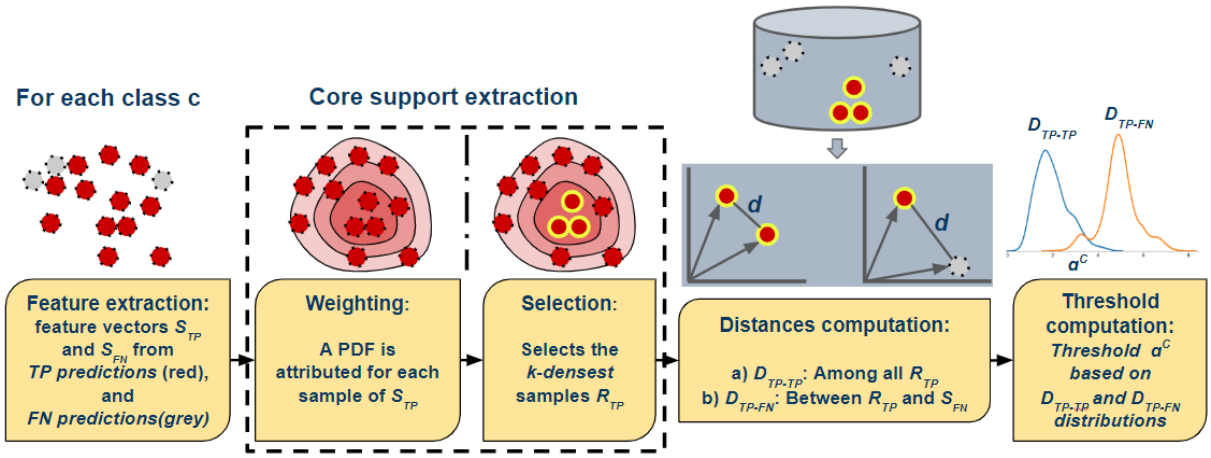


Figure 15: ***SENA monitor building.*** *1) A core support extraction algorithm is applied for weighting and selection of the most representative true positive neural activations to be stored and used at runtime. 2) It calculates the neural activation similarities among representative true positives, and also between true positives and false negatives.*

monitored model. SENA is built for a specific class c by using the information of true positives and false negatives outputted from the model for this specific class.

Worth mentioning that we do not use information from true negatives and false positives for the simple reason that for a single class c, it is not possible to know in advance the true negative objects, i.e. $\neg c$, that will appear at runtime. Without knowing the true negatives, we cannot calculate the false positives in advance as well since false positives are the incorrect predictions from the ML model when exposed to true negatives. Finally, as illustrated in Figure 15, four steps are required to build a SENA monitor:

1. **Training features extraction**. We feed the trained neural network with all the training images labeled as c. Then, we extract the neural activation vectors from these images and store them in two distinct sets: S_{TP}, containing all the feature vectors corresponding to correctly classified data (i.e., predicted as c), and S_{FN}, containing the incorrectly classified feature vectors. S_{TP} represents the features corresponding to the true positives for class c, while S_{FN} contains the false negatives. We note that these features are extracted from the same images that were used previously to train the model itself.

2. **Core support extraction**. As the monitored neural network is mostly correct on its training data, we usually have $|S_{TP}| >> |S_{FN}|$, and the number of samples in S_{TP} can quickly become very large. Hence, the second step of SENA intends to reduce the size of S_{TP} by selecting a small subset of representative samples. To do so, we apply a core support extraction algorithm [Ferreira et al., 2019]. Core support samples are the ones that contain the most informative characteristics of the underlying distribution of the set, i.e., they can represent most of the entire original distribution. Besides reducing the size of S_{TP}, core support extraction helps SENA to filter out outliers. A core support extraction algorithm has two steps:

    - *Weighting*: In this step, the probability distribution from which elements in S_{TP} were drawn is estimated using any multivariate density estimator, such as Gaussian mixture models (GMM) [Reynolds, 2015] or kernel density estimation (KDE) [Zhang

et al., 2006]. In this work, we chose KDE for its flexibility since it does not require knowing the parameters of the distribution to be fitted. In this work, the KDE algorithm uses the Gaussian kernel, and the bandwidth value is automatically tuned using a grid search algorithm. Hence, we train the KDE algorithm with S_{TP} and use the same trained estimator to calculate the probability density function value (PDF) for each sample in S_{TP}. Thus, each sample in S_{TP} is weighted based on its PDF.

- *Selection*: Then, we select the k densest samples in S_{TP}, i.e., the feature vectors corresponding to the k highest PDF values attributed for S_{TP}. A feature vector h with a high PDF value means that the sample is more likely to be drawn from the distribution learned by the KDE. Hence, these samples are the ones that best represent the original distribution of S_{TP}. The selected samples are stored in a set R_{TP}, containing the k most representative elements of S_{TP}. The choice of k represents a trade-off between accuracy in the training set and the amount of memory required to store the samples to be used at runtime. Previous works showed that such density estimation selection can discard up to 90% of the original dataset without a drastic performance drop [Ferreira et al., 2018]. Therefore, in this work, we also choose k to be 10% of the training set.

3. **Distances computation**. Next, SENA uses the set of true positive representative activations R_{TP} and the set of false negative activations S_{FN}, to calculate two sets of distances: 1) the set D_{TP-TP} contains all the distances among elements of R_{TP}, and 2) the set D_{TP-FN} contains all the distances between elements of R_{TP} and elements of S_{FN}. In this work, we use Euclidean distances between vectors.

4. **Automatic threshold computation**. Finally, the overlapping region between the distributions of D_{TP-TP} and D_{TP-FN} are analyzed. A threshold α belonging to this region is estimated by the algorithm (explained in Section 5.1.2). This threshold will be used at runtime to determine if an ML model prediction is reliable or not.

After iterating over all training images labeled as c, we produce a SENA monitor for this class, composed of a set of representative true positive feature vectors R_{TP}, and a distance threshold α^c. By repeating this process for every possible output class of the monitored neural network, we can build a complete SENA monitor.

5.1.2 Automatic threshold selection

As mentioned earlier, for a specific class, SENA generates the threshold α by analyzing the distributions of D_{TP-TP} and D_{TP-FN}. To understand how D_{TP-TP} and D_{TP-FN} are distributed, Figure 16 illustrates the densities of D_{TP-TP} (in blue) and D_{TP-FN} (in orange) for the first four classes of the CIFAR-10 dataset. Activation features were extracted from a ResNet model.

In the CIFAR-10 dataset, we can observe overlapping areas between D_{TP-TP} and D_{TP-FN}. It indicates that monitors that rely on activation function vectors might experience false positives (i.e., rejecting valid predictions) coming from the regions where there exist outliers, that is, false negatives with a higher distance density in the original distribution than true positives. It means that a threshold should be chosen considering the existence of such outliers to take a decision when flagging a possible unreliable prediction or not.

The same analysis is illustrated for the SVHN dataset in Figure 17. The distribution overlapping level observed in CIFAR-10 is not present in the SVHN dataset. Activation functions

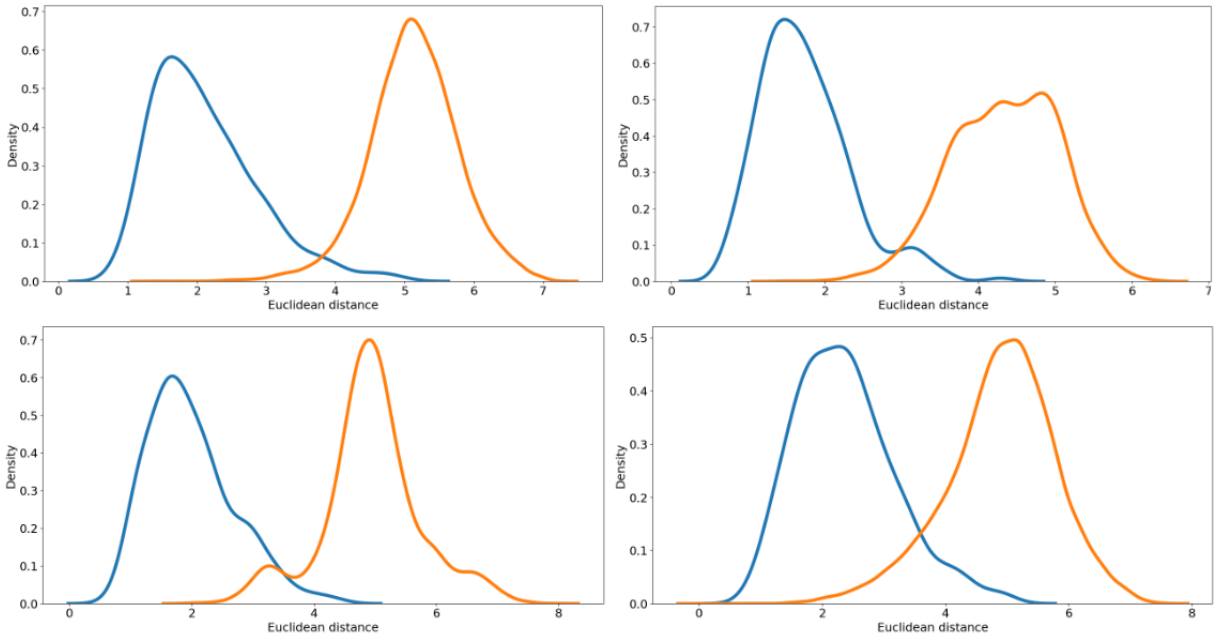


Figure 16: ***Activation vectors distances of the first 4 classes of CIFAR-10.*** *The x-axis represents the Euclidean distance values, and the y-axis represents the distribution density (e.g., amount of points). The blue line represents the distance distribution between true positives, and the orange line represents the distance distribution between true positives against false negatives.*

were extracted from the ResNet model. As can be noted, the margin for accepting a prediction as reliable should be increased if compared to the same threshold in the CIFAR-10 dataset. Therefore, it is clear that there is no single threshold that fits all datasets, and such a threshold needs to be chosen according to a specific analysis of the dataset during training. To decrease the human supervision from the daunting task of choosing a threshold, we apply a simple strategy of varying the threshold α based on the overlapping area of the distributions.

As illustrated in Figure 18, the threshold α is set to $\mu + \sigma$, where μ and σ are respectively the mean and standard deviation of the intersection set, i.e., all the elements in D_{TP-TP} and D_{TP-FN} that are greater than $\min(D_{TP-FN})$ and smaller than $\max(D_{TP-TP})$. The standard deviation is very useful for measuring the data dispersion in the distributions regardless of whether data are normally distributed. This allows α to assume different values depending on how the data in S_{TP} and S_{FN} are spread. Therefore, in smaller overlapping regions, α will naturally shift away and be more flexible to accept the predictions as reliable ones, which makes sense since less distribution overlapping means better separation between the distributions. If no FN exists for a specific class, the value of α will be the maximum value of D_{TP-TP}, and if an incoming average distance $\bar{d}$ falls outside the range of D_{TP-TP} it will be considered an unreliable prediction.

Next, we present how the monitor behaves at runtime.

5.1.3 Monitor at runtime

For a given class c, SENA starts with two artifacts previously calculated in the monitor building process: a threshold α^c, and a set of TP representatives (R^c_{TP}). As illustrated in Figure 19, SENA verifies if the incoming prediction might be unreliable. It extracts the neural activation

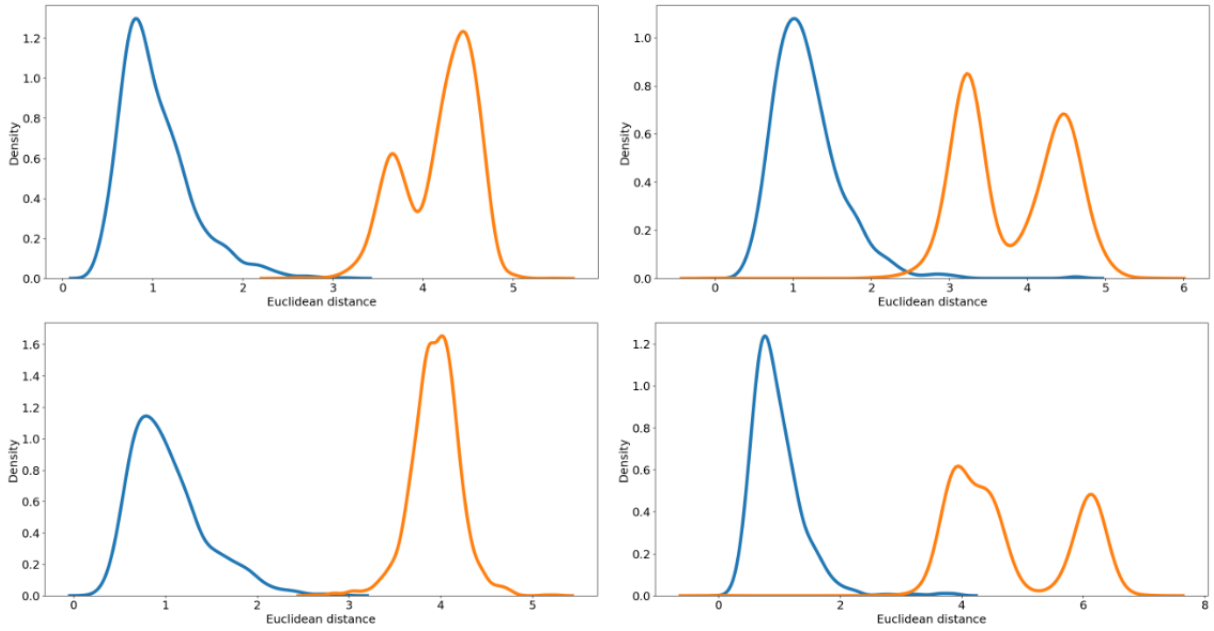


Figure 17: ***Activation function vectors distance of the first 4 classes of SVHN.*** *The x-axis represents the Euclidean distance values, and the y-axis represents the distribution density (e.g., amount of points). The blue line represents the distance distribution between true positives, and the orange line represents the distance distribution between true positives against false negatives.*

vector h from a particular neural-network layer during the ML model prediction on an image X. In this work, we take it from the last layer since it is the most informative one [Henzinger et al., 2020b]. Then, SENA calculates the average distance $\bar{d}^c$ between h and elements of R^c_{TP}. If $\bar{d}^c$ is higher than α^c, then SENA considers the prediction unreliable, otherwise, it considers the prediction reliable. The steps mentioned above are presented in Algorithm 1.

Algorithm 1 SENA monitoring at runtime

Input: ML model f; Image X; Prediction c; threshold α^c; set of representative TP R^c_{TP}.
Output: Returns *true* if an unreliable prediction detected, $false$ otherwise.

$h \leftarrow extract\_feature\_vector(X, f)$
$\bar{d}^c \leftarrow average\_distance\_by\_class(h, R^c_{TP})$

if $\bar{d}^c \geq \alpha^c$ **then**
    return $True$ *(i.e., reject NN prediction)*
else
    return $False$ *(i.e., accept NN prediction)*
end if

Next, we present an evaluation of our approach across image classification datasets.

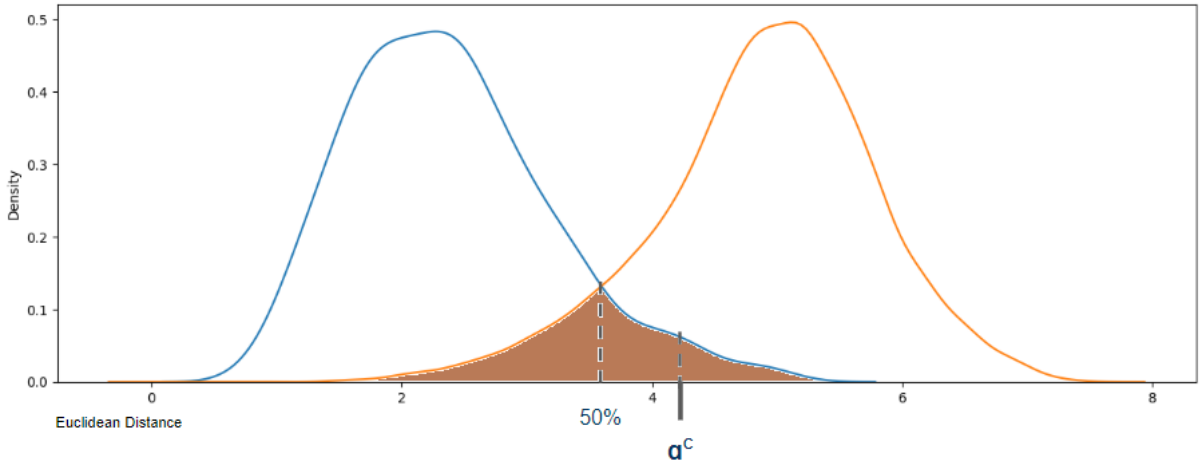


Figure 18: ***Flexible thresholds.*** *SENA threshold for true positives and false negatives.*

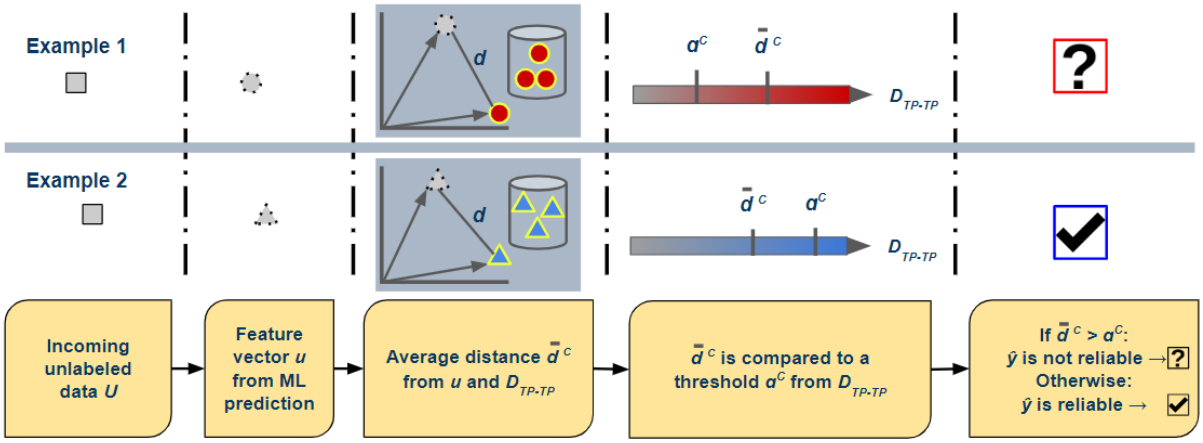


Figure 19: ***Two examples of SENA monitoring a class at runtime.*** *The average distance $\bar{d}^c$ is calculated by comparing the feature vector from the incoming image to D_{TP-TP}*
1) SENA triggers an alarm since $\bar{d}^c > \alpha^c$ (unreliable prediction);
2) SENA does not interfere in the ML prediction since $\bar{d}^c \leq \alpha^c$ (reliable prediction).

5.2 Experiments

Our experiments are conducted on several image classification tasks. Besides, since we want our experiments to be as close as possible to real-world scenarios, we set some important constraints, which can lead to different results from what has been reported in the literature previously. We use the concept of out-of-model-scope (OMS) introduced in this section and in Guérin et al. [2023]. Here are the details of our experiments:

- **OOD vs OMS**. In the OOD evaluation scenario, the monitor has to trigger an alarm when an OOD input data is encountered. However, in our experiments, the goal is to detect unreliable predictions instead of OOD data and we consider that a monitor is correct if it rejects unreliable predictions, independently of their ID/OOD status. This evaluation type is also known as out-of-model-scope detection (OMS).

- **Stream of random images**. We consider a scenario in which the images are randomly

given to the ML model in a streaming fashion, that is, the data stream will feed one image at a time to the ML model. Therefore, SMs based on time series, or using information from batches of data to perform the detection are not considered.

- **No hyperparameter tuning**. Datasets are divided into ID data (training/test) and OOD data (test), and the SMs are not allowed to use information or assumptions from test data (ID and OOD) to fine-tune rejection thresholds or to model their detection strategies. That is, SMs must be built using only the training data since in a real situation, it is unrealistic to consider that we know which kind of data will be fed to the ML model at runtime. Hence, related works that use information from test data (e.g., [Liang et al., 2018a]) are not considered.

- **No optimal performance metrics**. Several works in the literature test their solutions on the entire dataset with different rejection thresholds, and display the results corresponding to the best results according to some metric (e.g., F1, ROC curves, etc) [Guérin et al., 2023]. Although this approach can be relevant to demonstrate the optimal performance of a detector over all possible threshold choices, in a real-world scenario, one cannot perform such threshold optimization using the test data. Therefore, in this work, all analyses of the results are performed using objective metrics that do not depend on multiple runs using the test data. Thus, we decrease possible bias and conflicts between parameter fine-tuning and observed results.

Next, we present the experiment settings, the chosen datasets and OOD scenarios, and the comparative results.

5.2.1 Experiment settings

Regarding the experiments, we follow the FARM approach previously introduced in Chapter 3:

5.2.1.1 Fault model

We perform 38 experiments using three popular image datasets as ID, of which two are RGB: CIFAR-10 [Krizhevsky et al., 2009], and SVHN [Netzer et al., 2011]; and one is grayscale: MNIST [Deng, 2012]. For each ID dataset, we split it into train and test, in which the train is used to fit the monitors, and the test is used to evaluate the monitor under ID data. Except for novelty tasks, the ID test data is used to generate the OOD datasets through image transformations. Below, we present the tested OOD scenarios:

- **16 class novelty experiments**:

  1. For CIFAR-10 we use CIFAR-100 [Krizhevsky et al., 2009], GTSRB [Houben et al., 2013], SVHN, LSUN [Yu et al., 2015], Fractal [Hendrycks et al., 2022], and TinyImageNet (subset of ImageNet [Deng et al., 2009]) to represent the novel data.
  2. For SVHN we use CIFAR-10, CIFAR-100, Fractal, GTSRB, LSUN, and TinyImageNet.
  3. For MNIST [Deng, 2012], we use Fashion-MNIST [Xiao et al., 2017], E-MNIST (letters) [Cohen et al., 2017], American sign language (ASL) MNIST [Tecperson, 2017], and Simpsons MNIST [Attia, 2018].

- **16 distributional shift experiments**: for each RGB ID dataset, we apply eight image transformations from the AugLy library [Papakipos and Bitton, 2022]: brightness (factor=5), blur (radius=4), pixelization (ratio=0.1), shuffled pixels (factor=0.3), contrast (factor=9), opacity (level=0.2), rotate (degrees=25), and saturation (factor=17).
- **6 adversarial attack experiments**: for each RGB ID dataset, we apply three adversarial attacks from Torchattacks [Kim, 2020], with default parameters: fast gradient sign method (FGSM), DeepFool, and projected gradient descent (PGD).

5.2.1.2 Activity

Once the scenarios are set, we choose two different deep learning architectures to test with the SMs: the ResNet models from [Lee et al., 2018a] for RGB images, and a custom CNN [Deng, 2012] model for grayscale images.

Regarding the SMs, we compare our proposal to four other related works: outside-the-box (OTB) [Henzinger et al., 2020a], max softmax probability (MSP) [Hendrycks and Gimpel, 2017], max logits [Hendrycks and Gimpel, 2017], and energy [Liu et al., 2020b].

It is worth mentioning that the OTB method tested in this work is an improved version of the original paper that we developed for comparison purposes. In their original paper, the OTB method creates *2D boxes* by getting the max and min activation function values *from two indices* of the feature vector. However, in this work, the optimized OTB produces *n-dimensional boxes* by getting the max and min activation function values *from all indices* of the feature vector. Another important note is that, except for SENA and OTB, the SMs tested in our experiments require selecting a rejection threshold on the monitoring scores. The best strategy for choosing thresholds is not addressed in this work. However, in our experiments, we conducted two simple steps: 1) We fit the monitors using the same training data as the ML model. 2) We choose a threshold for the fitted monitors based on the best Matthews Correlation Coefficient (MCC) value for detecting correct/incorrect ML predictions from the training set. Such threshold tuning based on the training set is challenging but realistic.

If the ML model has no error for a specific class then the threshold is chosen anyway, since we select the threshold that resulted in the best MCC. Thus, it means that the first threshold value will be chosen since the model performs a perfect MCC every time for that class.

5.2.1.3 Readouts

The oracle for the monitor evaluation is as follows:

- True positive: the monitor triggers an alarm and the ML model prediction is wrong.
- True negative: the monitor is not triggered and the ML model prediction is right.
- False positive: the monitor is triggered and the ML model prediction is right.
- False negative: the monitor is not triggered and the ML model prediction is wrong.

5.2.1.4 Metrics

For the metrics, we apply four classification metrics for imbalanced datasets, also applied in the benchmark experiments presented in Chapter 3: Matthews Correlation Coefficient (MCC), false positive rate (FPR), false negative rate (FNR), and macro-F1 scores regarding the monitor's

output. For the OOD categories with more than 15 experiments, it is possible to perform Wilcoxon signed-rank tests with statistical guarantees [Dwivedi et al., 2017] for each pair of tested SM. Thus, we are able to compare if the SM methods are significantly different from each other across multiple experiments in a specific OOD category [Demšar, 2006].

5.2.2 Results

Below we show the results separated by novelty class, distributional shift, and adversarial attack. The best results in the tables are written in **bold**.

5.2.2.1 Novelty class

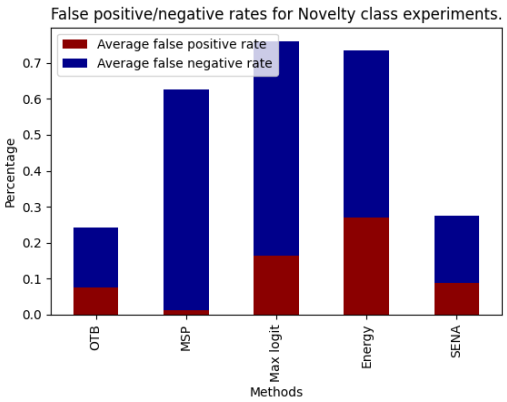
Table 12: ***MCC results for novelty class****: organized by ID data - OOD data.*

Experiments	OTB	MSP	Max Logit	Energy	SENA
CIFAR 10 - CIFAR 100	**0.41**	0.27	0.29	0.28	0.38
CIFAR 10 - Fractal	**0.76**	0.54	0.42	0.35	0.71
CIFAR 10 - GTSRB	**0.66**	0.34	0.28	0.25	0.55
CIFAR 10 - LSUN	0.78	0.52	0.44	0.38	**0.80**
CIFAR 10 - SVHN	0.48	0.33	0.23	0.17	**0.60**
CIFAR 10 - Tiny Imagenet	0.69	0.51	0.43	0.38	**0.75**
SVHN - CIFAR 10	0.81	0.29	0.45	0.51	**0.85**
SVHN - CIFAR 100	0.80	0.29	0.45	0.52	**0.84**
SVHN - Fractal	**0.85**	0.42	0.53	0.58	0.77
SVHN - GTSRB	0.67	0.24	0.33	0.38	**0.72**
SVHN - LSUN	0.78	0.38	0.53	0.59	**0.79**
SVHN - Tiny ImageNet	0.81	0.40	0.54	0.60	**0.82**
MNIST - ASL MNIST	**0.93**	0.87	0.60	0.90	0.90
MNIST - EMNIST	**0.67**	0.30	0.46	0.37	0.60
MNIST - Fashion MNIST	**0.74**	0.51	0.50	0.63	0.70
MNIST - Simpsons MNIST	**0.91**	0.82	0.64	0.88	0.83
Average rank	**1.5**	4	3.7	3.2	**1.5**

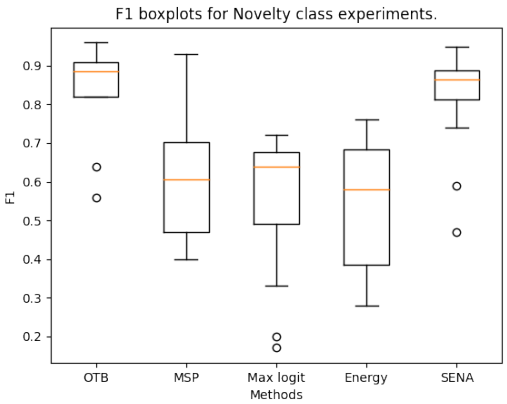
The MCC results corresponding to novelty OOD scenarios are reported in Table 12. Looking at the MCC results and the average rank, both SENA and OTB obtained the best results to avoid wrong predictions in the novelty class scenario. All the other three SMs (MSP, Max logit, and energy) performed similarly between themselves.

These results indicate that deploying SMs along with ML models for this scenario is a promising research direction. Besides, the SM methods based on activation functions (OTB, SENA) provided the best MCC results among the SMs.To understand better such behavior, we illustrate a positive and negative analysis for all novelty class experiments in Figure 20.

In Figure 20a, the average false negative rate of activation-function-based methods is way lower than the other SMs. This shows a better capacity of such a strategy in identifying data that is clearly OMS.The figure also shows that the softmax probability can be very useful in the verification of possible false positives in the novelty class scenario. The aggregated analysis can be seen in the Sub-figure 20b. On average, the macro-F1 values for OTB and SENA show stable and superior results.



(a) *FPR / FNR for novelty class experiments.*



(b) *F1 for novelty class experiments.*

Figure 20: ***False positives and negative analysis for novelty class experiments.*** *Figure 20a shows a lower number of false positives and false negatives for OTB and SENA. Figure 20b illustrates a good balance and stability for OTB and SENA regarding recovery and precision over new classes.*

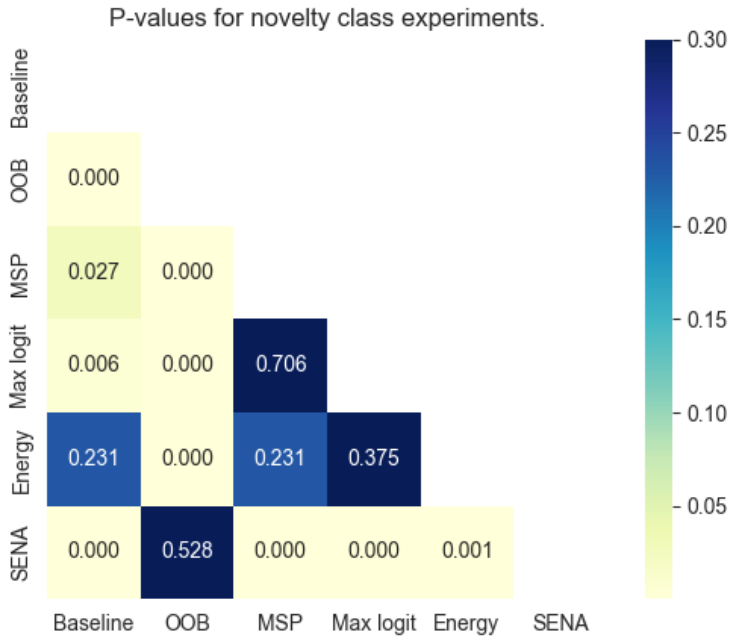


Figure 21: ***Novelty class experiments.*** *SENA statistically differs from other methods except for outside-of-the-box.*

Finally, we perform a statistical analysis of the results. Blue boxes mean the MCC values measured through the experiments, from two paired algorithms, were not significantly different from each other. On the contrary, p-values lower than 0.05 indicates that two paired algorithms have a statistical difference between their results for an OOD category. Figure 21 shows that SENA and OTB results are significantly different than the other SMs. Since Table 12 shows

Table 13: ***MCC results for distributional shift****: organized by ID data - Transformation.*

Experiments	OTB	MSP	Max Logit	Energy	SENA
CIFAR 10 - Blur	**0.65**	0.20	0.12	0.09	0.57
CIFAR 10 - Brightness	0.32	0.33	0.38	0.38	**0.48**
CIFAR 10 - Contrast	0.42	0.39	**0.44**	**0.44**	0.36
CIFAR 10 - Opacity	0.39	**0.46**	**0.46**	0.44	0.43
CIFAR 10 - Pixelization	**0.62**	0.21	0.14	0.10	0.58
CIFAR 10 - Rotate	**0.43**	0.37	0.36	0.34	0.38
CIFAR 10 - Saturation	**0.42**	0.36	**0.42**	0.41	0.35
CIFAR 10 - Shuffled pixels	0.37	0.33	0.35	0.33	**0.56**
SVHN - Blur	0.58	0.24	0.35	0.40	**0.60**
SVHN - Brightness	**0.72**	0.16	0.19	0.21	0.55
SVHN - Contrast	0.50	0.46	0.52	**0.53**	0.28
SVHN - Opacity	**0.55**	0.31	0.42	0.47	0.29
SVHN - Pixelization	0.66	0.20	0.31	0.36	**0.67**
SVHN - Rotate	**0.45**	0.31	0.40	0.43	0.27
SVHN - Saturation	**0.57**	0.32	0.40	0.44	0.34
SVHN - Shuffled pixels	**0.53**	0.29	0.39	0.43	0.39
Average rank	**1.6**	3.8	2.6	2.6	2.5

that OTB and SENA obtained the best average ranks, this combined analysis indicates that in fact, OTB and SENA were the best methods in the novelty class scenario.

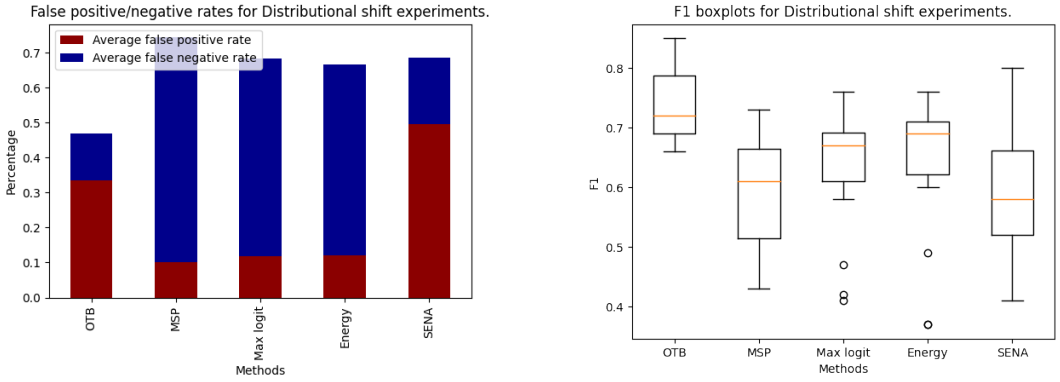
5.2.2.2 Distributional shift

The parameter values for the distributional shift transformations (e.g., level of blur, opacity, etc) were chosen such that the ML alone achieves MCC results between 0.2 and 0.8. It means that the ML model is placed in a challenging scenario that justifies the use of an SM since the ML is not capable of having strong results (MCC < 0.8), but at the same time, having reliably better results than a random classifier (MCC > 0.2).

According to Table 13, all methods, except the one based on softmax, have comparable results, in which SENA achieved the best MCC results in four times. The overall performance in the distributional shift scenario was worse than in the novelty class scenario. This decrease in the results indicates that activation function methods are better for novelty and less good for distribution shift. Figure 22 illustrates the analysis of positives and negatives regarding the distributional shift experiments.

We can see in Figure 22a the huge amount of false positives yielded by OTB and SENA since the activation function-based methods tend to have difficulties in distributional shift scenarios. On the other hand, MSP, Max logit, and energy, have lower FP rates since their thresholds are entirely based on the performance of the ML model over the ID data. Thus, since ID data support overlaps with OOD data, these methods tend to have lower FP rates. However, the amount of FN is as high as the other SMs. Such phenomenon leads to not-so-good overall macro-F1 results (Figure 22b).

Below, we summarize the SM performance over distributional shifts experiments by doing a statistical analysis of the presented MCC results.



(a) *FPR/FNR for distributional shift.* **(b)** *F1 for distributional shift experiments.*

Figure 22: ***False positives and negative analysis for distributional shift experiments.*** *Figure 22a shows a high rate of false negatives for all methods but a low rate of false positives for the methods not based on activation functions.*
Figure 22b shows that despite yielding more false positives, OTB still achieves a better balance between precision and recovery between the tested methods.

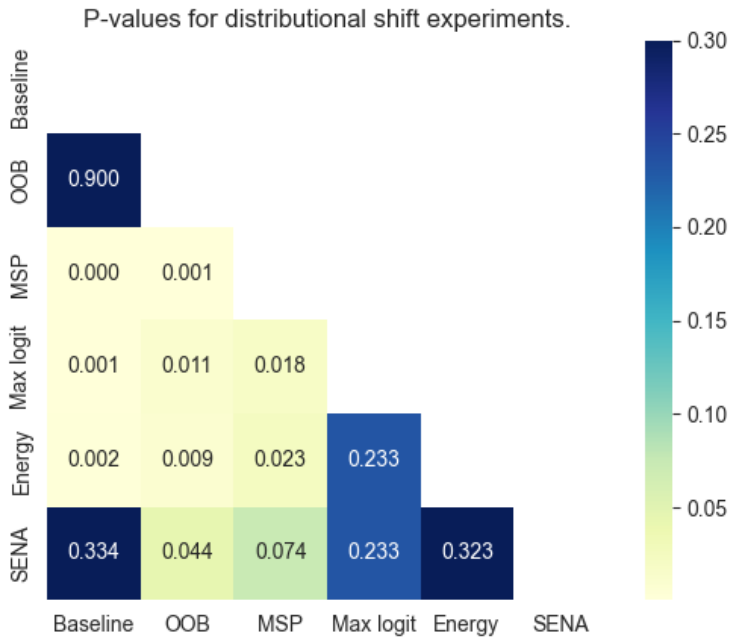


Figure 23: ***Distributional shift experiments.*** *SENA does not achieve results statistically better than related works.*

Figure 23 combined with the average ranks in Table 13 shows that just OTB was significantly better than the other methods, while SENA and the other methods had similar performance.

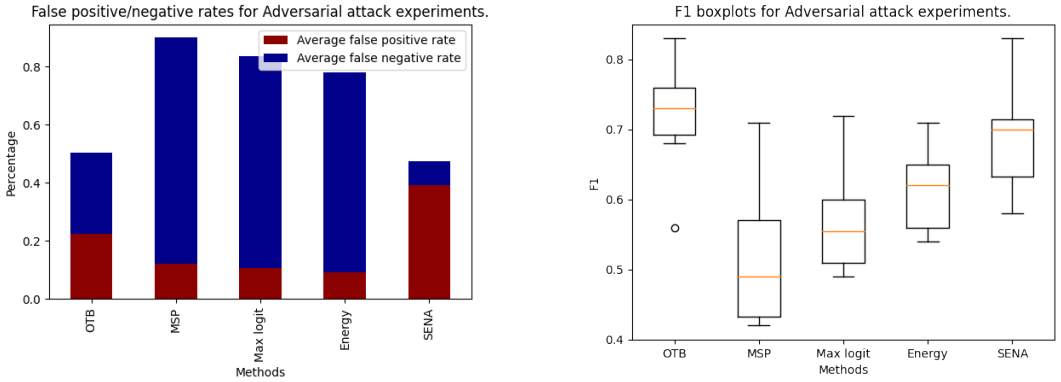
Table 14: ***MCC results for adversarial attack****: organized by ID data - Attack.*

Experiments	OTB	MSP	Max Logit	Energy	SENA
CIFAR 10 - Deep fool	0.39	0.16	0.14	0.20	**0.47**
CIFAR 10 - FGSM	0.48	0.44	0.45	0.43	**0.62**
CIFAR 10 - PGD	0.20	0.06	0.20	0.21	**0.68**
SVHN - Deep fool	**0.50**	0.13	0.05	0.12	0.48
SVHN - FGSM	0.63	0.24	0.33	0.37	**0.38**
SVHN - PGD	**0.54**	0.09	0.15	0.38	0.50
Average rank	1.5	4.3	4	3.3	**1.2**

5.2.2.3 Adversarial attacks

An important note is that the tested methods were not originally developed to detect adversarial images, and to perform such detection requires particular defense strategies for each type of attack which is an open research problem. However, we find it relevant to experiment on these three adversarial scenarios to give possible insights for future work and to complement the analysis of all OOD categories mentioned early on in this work.

Table 14 shows the MCC results in the adversarial attack scenario in which the monitoring strategy applied by the two activation-function-based SM were consistently better than the other methods. Figure 24 illustrates this analysis under the optics of positives and negatives.



(a) *FPR/FNR for adversarial attack.* **(b)** *F1 for adversarial attack experiments.*

Figure 24: ***False positives and negative analysis for adversarial attack experiments.*** *Figure 24a shows that methods not based on activation functions interfere less in the ML decision, which explains the high rate of false negatives and low rate of false positives.*
Figure 24b shows that activation-function-based methods are able to have better macro-F1 on average compared to the other methods.

Figure 24b shows good macro-F1 values for OTB and SENA. However, such results reflect the precision and recall of the data with equal importance. It means that even though their results are not bad when looking for both ID and OOD, it will depend on the amount of OOD data exposed at runtime. Such phenomenon is better understood by seeing Figure 24a. All tested SM yielded considerable amounts of false negatives. However, SENA obtained the lowest value among the tested methods.

5.3 Conclusion

In this work, we proposed SENA, an SM approach that uses a similarity-based error-checking of neural activations to monitor ML predictions at runtime. SENA uses a statistical weighting and filtering method to select the minimal set of the most representative samples from the distribution which is used as a reference at runtime. This statistical method is combined with a simple similarity algorithm such as a Euclidean distance. Moreover, the SENA threshold is chosen automatically based on the distributions of true positives and false negatives in the training set.

Differently from most previous works, SENA focuses on detecting when the ML prediction is error-prune, independently if the data is ID or OOD, instead of detecting when an image is OOD data or not. Our objective in this chapter was to demonstrate that such an approach is realistic and comparable to other monitors. Finally, SENA and the optimized version of OTB achieve the best overall results. Thus, SENA achieves a similar performance to OTB for novelty class scenarios, a slightly worst performance in distributional shift scenarios, and better results for adversarial attack scenarios. Our main lessons learned are:

- *Neural activation vectors from true positives and false negatives can be very similar:* most related works rely on the assumption that incorrect classifications can be spotted by inspecting all neural activation vectors that are not similar to the distribution of correct ones. Most of the time this is true for new classes (novelty), but not entirely true for ID data or other types of OOD data. We showed that feature vectors of TP and FN can be very similar to each other inside ID data. The necessity of filtering the right data to be used to model a better separation between TP and FN data led us to the second finding.

- *Not all training data is needed to build a monitor:* just a part of true positive samples is needed to extract the necessary information to perform monitoring of a class. As shown, there are several outliers present in the training data when comparing neural activation vectors. By outliers, we mean activation function values from false negatives that are similar to true positives, sometimes even more similar than other true positives compared between themselves. Therefore, a good filtering mechanism, such as the core support extraction algorithm applied in this work, can help to exclude samples that contribute negatively to make a better separation between false negatives and true positives. However, it can be hard to select different thresholds for different ID datasets. This showed the importance to investigate further sources of information during training, which led us to the third finding.

- *Extracting information from both true positives and false negatives is advantageous:* the use of both, when possible, allowed us to check different boundaries for the monitor's thresholds by checking the distribution of false negatives (e.g., error-prune ID data) instead of just checking what is normal (e.g., ID data that the ML model gives correct classification).

We believe that our approach can be improved in several ways. For example, better similarity algorithms could be applied in place of Euclidean distance, or even the choice of better strategies to set up the detection thresholds. There is also the possibility of applying pre-processing algorithms to the sparse neural activation vectors, increasing the level of information to be fitted in the monitor. Finally, most works in the literature, including ours, just use information from the penultimate layer of neural networks, but the combination of more layers can also be investigated [Hornauer and Belagiannis, 2023].

An important future direction is to move from *OOD/misclassification detection* to *error classification* since most works in the literature do not provide solutions on how to react after an unreliable prediction is detected. Knowing such a reaction is fundamental to practical applications of autonomous systems. However, to be able to do that, researchers need to know how to recognize which type of error-prune data is being detected at the moment. Finally, one can also think about how to apply more generic safety metrics [Guerin et al., 2022b] to capture the safety benefits of the evaluated monitor, availability costs, or negative impacts of the monitor on the system's availability.

In the next chapter, we conclude this thesis by highlighting a resume of our contributions, limitations, and perspectives for future work.

Chapter 6

Conclusion

Computer vision models are commonly applied in safety-critical tasks on autonomous systems. These models are generally built with deep learning algorithms due to their capacity on learning patterns from complex data such as images. However, such deep learning models can output wrong outputs even with high confidence. Safety monitors can be employed to inspect the model inputs, internal values, and outputs to ensure the safety of such deep learning-based functions. In this thesis, we presented the main steps and challenges to build a framework for monitoring and handling the safety of ML-based vision functions for autonomous systems at runtime. This research covered the current literature regarding safety monitors, and the necessary steps to build, test, and evaluate them. We also proposed a new hybrid SM for ML-based classifiers. Next, we summarize all the contributions originating from this research.

6.1 Resume of contributions

We presented a complete literature review in Chapter 2, in which we addressed what threats are relevant for such ML-based vision functions, how to model the safety properties, how to detect ML failures, how to recover from these failures, and how to evaluate such SMs. To the best of our knowledge, this was the first study applied to the safety monitoring of ML-based vision functions for safety-critical autonomous systems.

Next, we presented in Chapter 3, a framework for benchmarking safety monitors for image classifiers. To the best of our knowledge, this was the first benchmark framework focused on evaluating SM for vision-based functions covering the entire pipeline: data generation, system testing, and evaluation. Besides, we also showed the main flaws in current state-of-the-art OOD detectors, highlighting the necessity of new directions in the existing safe AI research.

In Chapter 4, we went one step further to the evaluation of SM approaches, going from the unit level to the analysis of SM at the system level. We proposed an evolutionary OOD-image-generation approach capable of finding relevant threats in a safety-critical system depending on ML models. Our approach has the advantage of finding the minimal perturbations capable of provoking failures in object detectors at runtime. Moreover, we could reduce the amount of time to find such relevant perturbations up to ten times compared to other approaches.

Finally, in Chapter 5, we presented SENA, an SM that uses information from both true positive and false negative activation functions combined with a statistical approach to select the most representative feature vectors from training data. Unlike most works in the literature, SENA is designed to detect incorrect predictions, independently if exposed to ID or OOD data. Our proposal was able to achieve SOTA performance regarding the detection of wrong ML predictions. The first results showed that selecting representative instances from the training set can lead to better separation between two types of feature vectors: feature vectors that represent the distribution of TP and FN. This work is a preliminary validation for exploring new solutions that do not only focus on the OOD paradigm.

6.2 Limitations

We recall the main limitations for each chapter and the main overall limitations regarding this thesis. Hence, here are the limitations for each chapter:

- *Chapter 2:* we provided a picture of the current state-of-the-art but the domain evolves very quickly.
- *Chapter 3:* we did not test more and bigger DNN architectures in our benchmark framework. Such experiments could produce a more robust evaluation.
- *Chapter 4:* the first version of SIMOOD adds a significant memory overhead to the simulation. Besides, SIMOOD does not use a scenario generation capable of varying the scenarios, and parameters which is important to produce a more robust analysis.
- *Chapter 5:* despite our SM having good preliminary results compared to related works, it still has room for being optimized, and probably more analysis needs to be done.

Regarding the overall limitations of this work, we can highlight the fact that our work mainly focuses on classification functions (except in Chapter 4). It means that existent solutions for other ML-based vision functions such as object detectors were not deeply investigated neither new SM methods proposed for this category of computer vision tasks. However, we focused deeply on classification tasks in order to validate and provide a solid basis for further research on more complex tasks such as object detection functions.

A second main limitation is the absence of proposals regarding possible recovery mechanisms. Currently, the tested data-based SMs (including our proposal) are limited to detection only. In real scenarios, triggering an alarm for possible model misclassification is critical and challenging but still, it is just a part of the solution. An SM should also know how to recover once the detection is made. However, developing and testing such a recovery mechanism combined with the detection is also very challenging. The reason is that the reaction triggered by the recovery mechanism can worsen the current state of the autonomous system. For example, a denoise filter as a recovery mechanism after the detection of a dangerous noise/haze in the image can result in an image even more difficult to read by the ML model. One possible direction to solve this issue is to develop frameworks that allow quick testing of combinations of detection and reaction mechanisms and choose the ones that lead to satisfactory safety outcomes.

Finally, a third main limitation is the lack of SM proposals for the system level. However, as one could realize in Chapter 2, such SMs are also scarce in the literature due to many reasons especially the one mentioned in the preceding paragraph. A possible alternative could be the adaptation of the SENA monitor to work with neural-network-based object detectors, in which the SM should be tested to handle multiple ML predictions at once.

6.3 Perspectives

Here we present short and long-term perspectives regarding the research performed during this thesis. A short-term perspective is the development of recovery methods after an unreliable ML prediction is detected. Since all resources to implement such recovery were mapped in Chapter 2, and a proposal to detect ML misclassification at runtime was also implemented in Chapitre 5, it is worth implementing a recovery mechanism to work with data-based monitors. Another short-term perspective is to improve the SiMOOD framework to allow usage in industrial applications.

The potential of SiMOOD in industrial applications has attracted some scientific partners such as ANITI [1], and industrial companies from the automotive sector such as Continental Engineering Services [2], resulting in invitations from these entities to present SiMOOD.

Regarding long-term perspectives, one of the most important ones is the development of an entire SM solution (detection and recovery) applied to an industrial case. However, developing such a full solution still requires a long-time research commitment. In the current state, SM for perception tasks is very scarce in industrial cases since the solutions are normally not fully disclosed and the amount of false positives/false negatives still is high, which slows the adoption of such SM. Other important constraints for embedded systems are also a challenge for these SMs such as memory, and processing time. Besides, other important tools for verifying, and validating such solutions should be defined/developed as well.

[1] https://aniti.univ-toulouse.fr/en/
[2] https://conti-engineering.com/

Bibliography

Raja Ben Abdessalem, Shiva Nejati, Lionel C Briand, and Thomas Stifter. Testing vision-based control systems using learnable evolutionary algorithms. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 1016–1026. IEEE, 2018. 52

Karrar Hameed Abdulkareem, Nureize Arbaiy, AA Zaidan, BB Zaidan, Osamah Shihab Albahri, MA Alsalem, and Mahmood M Salih. 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*, 33:1029–1054, 2021. 18

Mukhlas Adib. CARLA 2D Bounding Box Annotation Module. `https://github.com/MukhlasAdib/CARLA-2DBBox/`, 2022. [Online; accessed 23-June-2022]. 44

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018. 8

Jalal Al-Afandi and András Horváth. Class retrieval of adversarial attacks. *Workshop on Adversarial Machine Learning in Real-World Computer Vision Systems and Online Challenges (AML-CV)*, 2021. 18

Fadi Al-Khoury. Safety of machine learning systems in autonomous driving. Master's thesis, KTH Royal Institute of Technology School of Industrial Engineering and Management, Stockholm, Sweden, 2017. 10

Suad A Alasadi and Wesam S Bhaya. Review of data preprocessing techniques in data mining. *Journal of Engineering and Applied Sciences*, 12(16):4102–4107, 2017. 5

Andrej Karpathy. Tesla autonomous driving talk at cvpr 2021. `https://pharath.github.io/self%20driving/Karpathy-CVPR-2021/`, 2021. Online; accessed 07 June 2022. 19

Anonymous. Repository of the paper: benchmarking safety monitors for image classifiers with machine learning, 2021. URL `https://anonymous.4open.science/r/benchmark-sut-BE9B`. 31

Paolo Arcaini, Alessandro Calò, Fuyuki Ishikawa, Thomas Laurent, Xiao-Yi Zhang, Shaukat Ali, Florian Hauer, and Anthony Ventresque. Parameter-based testing and debugging of autonomous driving systems. In *2021 IEEE Intelligent Vehicles Symposium Workshops (IV Workshops)*, pages 197–202. IEEE, 2021. 40

Jean Arlat, Martine Aguera, Louis Amat, Yves Crouzet, J-C Fabre, J-C Laprie, Eliane Martins, and David Powell. Fault injection for dependability validation: A methodology and some applications. *IEEE Transactions on software engineering*, 16(2):166–182, 1990. 26

Alexandre Attia. Sinpson mnist., 2018. URL `https://github.com/alexattia/SimpsonRecognition`. 61

Algirdas Avizienis, J-C Laprie, Brian Randell, and Carl Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE transactions on dependable and secure computing*, 1(1):11–33, 2004. 1

Mukund Balasubramanian, Eric L Schwartz, Joshua B Tenenbaum, Vin de Silva, and John C Langford. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002. 32

George A Bekey. *Autonomous robots: from biological inspiration to implementation and control*. MIT press, 2005. 1

Alan Joseph Bekker and Jacob Goldberger. Training deep neural-networks based on unreliable labels. In *IEEE International Conference on Acoustics, Speech and Signal Processing, China*, pages 2682–2686. IEEE, 2016. 5

Hermann Blum, Paul-Edouard Sarlin, Juan Nieto, Roland Siegwart, and Cesar Cadena. Fishyscapes: A benchmark for safe semantic segmentation in autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 7

Daniel Bogdoll, Maximilian Nitsche, and J Marius Zöllner. Anomaly detection in autonomous driving: A survey. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4488–4499, 2022. 39

Adith Boloor, Karthik Garimella, Xin He, Christopher Gill, Yevgeniy Vorobeychik, and Xuan Zhang. Attacking vision-based perception in end-to-end autonomous driving models. *Journal of Systems Architecture*, 110:101766, 2020. 8, 41

Markus Borg, Cristofer Englund, Krzysztof Wnuk, Boris Durann, Christoffer Lewandowski, Shenjian Gao, Yanwen Tan, Henrik Kaijser, Henrik Lönn, and Jonnas Törnqvist. 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(1):1–19, 2019. 23

George EP Box and George C Tiao. *Bayesian inference in statistical analysis*, volume 40. John Wiley & Sons, 2011. 12

Eric Breck, Shanqing Cai, Eric Nielsen, Michael 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)*, pages 1123–1132. IEEE, 2017. 5

Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018. 1

Cornelius Buerkle, Florian Geissler, Michael Paulitsch, and Kay-Ulrich Scholl. Fault-tolerant perception for automated driving a lightweight monitoring approach. *arXiv preprint arXiv:2111.12360*, 2021. 11, 17, 23

Alexander Buslaev, Vladimir I Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A Kalinin. Albumentations: fast and flexible image augmentations. *Information*, 11(2):125, 2020. 45

Feiyang Cai and Xenofon 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)*, pages 174–183. IEEE, 2020. 11, 14, 20

Feiyang Cai, Jiani Li, and Xenofon Koutsoukos. Detecting adversarial examples in learning-enabled cyber-physical systems using variational autoencoder for regression. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 208–214. IEEE, 2020. 14

Michele Giovanni Calvi. *Runtime Monitoring of Cyber-Physical Systems Using Data-driven Models*. PhD thesis, University of Illinois at Chicago, 2019. 1

J. Cartucho, R. Ventura, and M. Veloso. Robust object recognition through symbiotic deep learning in mobile robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2336–2341, 2018. 45

Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018. 5, 8

Valerie Chen, Man-Ki Yoon, and Zhong Shao. Task-aware novelty detection for visual-based deep learning in autonomous systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11060–11066. IEEE, 2020. 11, 15, 20

Wenhu Chen, Yilin Shen, Hongxia Jin, and William Wang. A variational dirichlet framework for out-of-distribution detection. *arXiv preprint arXiv:1811.07308*, 2018. 12

Yuhang Chen, Chih-Hong Cheng, Jun Yan, and Rongjie Yan. Monitoring object detection abnormalities via data-label and post-algorithm abstractions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6688–6693. IEEE, 2021. 11, 15, 20

Chih-Hong Cheng, Georg Nührenberg, and Hirotoshi Yasuoka. Runtime monitoring neuron activation patterns. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy*, pages 300–303. IEEE, 2019. 11, 15, 21, 25, 27

Gunjan Chhablani, Abheesht Sharma, Harshit Pandey, and Tirtharaj Dash. Superpixel-based domain-knowledge infusion in computer vision. *arXiv preprint arXiv:2105.09448*, 2021. 11, 12

Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC genomics*, 21(1):6, 2020. 29

Darren Cofer, Isaac Amundson, Ramachandra Sattigeri, Arjun Passi, Christopher Boggs, Eric Smith, Limei Gilham, Taejoon Byun, and Sanjai Rayadurgam. Run-time assurance for learning-based aircraft taxiing. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1–9. IEEE, 2020. 11, 17, 18, 22

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)*, pages 2921–2926. IEEE, 2017. 61

Filipe Condessa, José Bioucas-Dias, and Jelena Kovačević. Performance measures for classification systems with rejection. *Pattern Recognition*, 63:437–450, 2017. 7

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 6, 7

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Boosting with abstention. *Advances in Neural Information Processing Systems*, 29:1660–1668, 2016. 11, 13

Gabriele Costante and Michele Mancini. Uncertainty estimation for data-driven visual odometry. *IEEE Transactions on Robotics*, PP:1–20, 06 2020. 11, 12

Tirtharaj Dash, Sharad Chitlangia, Aditya Ahuja, and Ashwin Srinivasan. Incorporating domain knowledge into deep neural networks. *arXiv preprint arXiv:2103.00180*, 2021. 12

Erwin De Gelder and Olaf Op Den Camp. Procedure for the safety assessment of an autonomous vehicle using real-world scenarios. *arXiv preprint arXiv:2012.00643*, 2020. 22

Erwin De Gelder, Hala Elrofai, Arash Khabbaz Saberi, Jan-Pieter Paardekooper, Olaf Op Den Camp, and Bart De Schutter. Risk quantification for automated driving systems in real-world driving scenarios. *IEEE Access*, 2021. 22

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research,*, 7(Jan):pp. 1–30, 2006. 63

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6, 20, 61

Li Deng. The MNIST database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 20, 61, 62

Taylor Denouden, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance. *arXiv preprint arXiv:1812.02765*, 2018. 11, 14, 20

I Donadello, L Serafini, and AS d'Avila Garcez. Logic tensor networks for semantic image interpretation. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 1596–1602. IJCAI, 2017. 11, 13

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Xiaolin Hu, and Jun Zhu. Discovering adversarial examples with momentum. *arXiv preprint arXiv:1710.06081*, 2017. 30

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 20, 39

Tommaso Dreossi, Alexandre Donzé, and Sanjit A Seshia. Compositional falsification of cyber-physical systems with machine learning components. *Journal of Automated Reasoning*, 63(4): 1031–1053, 2019a. 2, 9, 41

Tommaso Dreossi, Daniel J Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A Seshia. Verifai: A toolkit for the formal design and

analysis of artificial intelligence-based systems. In *International Conference on Computer Aided Verification*, pages 432–442. Springer, 2019b. 9

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016. 1

Alok Kumar Dwivedi, Indika Mallawaarachchi, and Luis A Alvarado. Analysis of small sample size studies using nonparametric bootstrap test with pooled resampling method. *Statistics in medicine*, 36(14):2187–2205, 2017. 63

Hazem Fahmy, Fabrizio Pastore, and Lionel Briand. Hudd: A tool to debug dnns for safety analysis. In *2022 IEEE/ACM 44st International Conference on Software Engineering*, 2022. 41

José M Faria. Machine learning safety: An overview. In *Proceedings of the 26th Safety-Critical Systems Symposium, York, UK*, pages 6–8, 2018. 1

Di Feng, Ali Harakeh, Steven L Waslander, and Klaus Dietmayer. A review and comparative study on probabilistic object detection in autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 12

Yeli Feng and Arvind 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*, pages 225–226, 2021. 14, 20

Raul S Ferreira, Geraldo Zimbrão, and Leandro GM Alvim. Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency. *Information Sciences*, 488:219–237, 2019. 3, 7, 54, 56

Raul Sena Ferreira. Code repository for the paper: SiMOOD: evolutionary safety simulation testing with out-of-distribution images, 2022. URL `https://github.com/raulsenaferreira/SiMOOD`. 40, 46, 52

Raul Sena Ferreira, Bruno MA da Silva, Wendell Teixeira, Geraldo Zimbrao, and Leandro Alvim. Density-based core support extraction for non-stationary environments with extreme verification latency. In *2018 7th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 181–187. IEEE, 2018. 57

John Fox and Subrata Das. Safe and sound. *Artificial intelligence in hazardous applications*, 307, 2000. 2

Giorgio Fumera and Fabio Roli. Support vector machines with embedded reject option. In *International Workshop on Support Vector Machines*, pages 68–82. Springer, 2002. 13

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning (ICML), New York, United States*, pages 1050–1059, 2016. 1, 11, 12, 17

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. *Advances in neural information processing systems*, 30, 2017. 12

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021. 12

Yonatan Geifman and Ran El-Yaniv. Selectivenet: A deep neural network with an integrated reject option. In *International Conference on Machine Learning*, pages 2151–2159. PMLR, 2019. 11, 13

Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer, 2020. 12

Lovedeep Gondara. Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th international conference on data mining workshops (ICDMW)*, pages 241–246. IEEE, 2016. 18

Raphael Gontijo-Lopes, Yann Dauphin, and Ekin D Cubuk. No one representation to rule them all: Overlapping features of training methods. *arXiv preprint arXiv:2110.12899*, 2021. 14, 16, 19

Federica Granese, Marco Romanelli, Daniele Gorla, Catuscia Palamidessi, and Pablo Piantanida. Doctor: A simple method for detecting misclassification errors. *Advances in Neural Information Processing Systems*, 34, 2021. 5, 11, 15, 21

Joris Guérin, Anne Magaly de Paula Canuto, and Luiz Marcos Garcia Goncalves. Robust detection of objects under periodic motion with gaussian process filtering. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 685–692. IEEE, 2020. 11, 16

Joris Guérin, Kevin Delmas, and Jérémie 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)*, pages 55–62, 2021a. 1, 22

Joris Guérin, Stephane Thiery, Eric Nyiri, Olivier Gibaru, and Byron Boots. Combining pretrained cnn feature extractors to enhance clustering of complex natural images. *Neurocomputing*, 423:551–571, 2021b. 14

Joris Guerin, Kevin Delmas, and Jérémie Guiochet. Evaluation of runtime monitoring for UAV emergency landing. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2022a. 17, 18, 20, 22

Joris Guerin, Raul Sena Ferreira, Kevin Delmas, and Jérémie Guiochet. Unifying evaluation of machine learning safety monitors. In *The 33rd IEEE International Symposium on Software Reliability Engineering (ISRRE)*. IEEE, 2022b. 69

Joris Guérin, Kevin Delmas, Raul Sena Ferreira, and Jérémie Guiochet. Out-of-distribution detection is not all you need. In *Proceedings of the AAAI conference on artificial intelligence (2023)*, 2023. 54, 60, 61

Jeremie Guiochet, Damien Martin-Guillerez, and David Powell. Experience with model-based user-centered risk assessment for service robots. In *2010 IEEE 12th International Symposium on High Assurance Systems Engineering*, pages 104–113. IEEE, 2010. 9

Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020. 3

Corina Gurău, Dushyant Rao, Chi Hay Tong, and Ingmar Posner. Learn from experience: probabilistic prediction of perception performance to avoid failure. *The International Journal of Robotics Research*, 37(9):981–995, 2018. 14

Sami Haddadin, Michael Suppa, Stefan Fuchs, Tim Bodenmüller, Alin Albu-Schäffer, and Gerd Hirzinger. Towards the robotic co-worker. In *Robotics Research*, pages 261–282. Springer, 2011. 2

Tamás Haidegger. Autonomy for surgical robots: Concepts and paradigms. *IEEE Transactions on Medical Robotics and Bionics*, 1(2):65–76, 2019. 1

Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel 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)*, pages 85–95. IEEE, 2020. 22

Fitash Ul Haq, Donghwan Shin, Shiva Nejati, and Lionel Briand. Can offline testing of deep neural networks replace their online testing? *Empirical Software Engineering*, 26(5):1–30, 2021. 40, 41

Christopher Harper, Greg Chance, Abanoub Ghobrial, Saquib Alam, Tony Pipe, and Kerstin Eder. Safety validation of autonomous vehicles using assertion-based oracles. *arXiv preprint arXiv:2111.04611*, 2021. 11, 15

Martin E Hellman. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics*, 6(3):179–185, 1970. 13

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019. 7, 20, 30, 39

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136, 2016a. URL `http://arxiv.org/abs/1610.02136`. 25

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016b. 11, 12, 15, 20, 21, 39

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017. 62

Dan Hendrycks, Andy Zou, Mantas Mazeika, Leonard Tang, Bo Li, Dawn Song, and Jacob Steinhardt. Pixmix: Dreamlike pictures comprehensively improve safety measures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16783–16792, 2022. 61

Thomas A Henzinger, Anna Lukina, and Christian Schilling. Outside the box: Abstraction-based monitoring of neural networks. In *24th European Conference on Artificial Intelligence-ECAI 2020*, pages 2433–2440, 2020a. 11, 15, 20, 62

Thomas A. Henzinger, Anna Lukina, and Christian Schilling. Outside the box: Abstraction-based monitoring of neural networks. In *ECAI*, volume 325 of *Frontiers in Artificial Intelligence and Applications, Santiago de Compostela, Spain*, pages 2433–2440. IOS Press, 2020b. doi: 10.3233/FAIA200375. 25, 27, 31, 37, 55, 59

Julia Hornauer and Vasileios Belagiannis. Heatmap-based out-of-distribution detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2603–2612, 2023. 68

Sebastian Houben, Johannes Stallkamp, Jan Salmen, Marc Schlipsing, and Christian Igel. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. In *The 2013 international joint conference on neural networks (IJCNN)*, pages 1–8. Ieee, 2013. 61

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt 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*, pages 10951–10960, 2020. 11, 15, 20, 55

Po-Yu Huang, Wan-Ting Hsu, Chun-Yueh Chiu, Ting-Fan Wu, and Min Sun. Efficient uncertainty estimation for semantic segmentation in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 520–535, 2018. 11, 12

Christian Hubschneider, Robin Hutmacher, and J Marius Zöllner. Calibrating uncertainty models for steering angle estimation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1511–1518. IEEE, 2019. 11, 12

Gunel Jahangirova. Oracle problem in software testing. In *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 444–447, 2017. 23

Arpan Jain, Apoorva Mishra, Anupam Shukla, and Ritu Tiwari. A novel genetically optimized convolutional neural network for traffic sign recognition: A new benchmark on belgium and chinese traffic sign datasets. *Neural Processing Letters*, 50(3):3019–3043, 2019. 30

Heinrich Jiang, Been Kim, Melody Y Guan, and Maya Gupta. To trust or not to trust a classifier. *arXiv preprint arXiv:1805.11783*, 2018. 55

Glenn Jocher, Alex Stoken, Ayush Chaurasia, Jirka Borovec, NanoCode012, TaoXie, Yonghye Kwon, Kalen Michael, Liu Changyu, Jiacong Fang, Abhiram V, Laughing, tkianai, yxNONG, Piotr Skalski, Adam Hogan, Jebastin Nadar, imyhxy, Lorenzo Mammana, AlexWang1900, Cristi Fati, Diego Montes, Jan Hajek, Laurentiu Diaconu, Mai Thanh Minh, Marc, albinxavi, fatih, oleg, and wanghaoyang0106. ultralytics/yolov5: v6.0 - YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support, October 2021. 45

Joint Authorities for Rulemaking of Unmanned Systems (JARUS). JARUS guidelines on Specific Operations Risk Assessment (SORA) v2.0. Guidelines, JARUS, 2019. 22

Gholamreza Kakamanshadi, Savita Gupta, and Sukhwinder Singh. A survey on fault tolerance techniques in wireless sensor networks. In *2015 international conference on green computing and internet of things (ICGCIoT)*, pages 168–173. IEEE, 2015. 18

Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. Model assertions for debugging machine learning. In *NeurIPS MLSys Workshop*, volume 3, page 10, 2018. 11, 15, 21

Daniel Kang, Deepti Raghavan, Peter Bailis, and Matei Zaharia. Model assertions for monitoring and improving ml models. *Proceedings of Machine Learning and Systems*, 2:481–496, 2020. 24

Yiannis Kantaros, Taylor Carpenter, Sangdon Park, Radoslav Ivanov, Sooyong Jang, Insup Lee, and James Weimer. Visionguard: Runtime detection of adversarial inputs to perception systems. *arXiv preprint arXiv:2002.09792*, 2020. 11, 16, 20

Eliahu Khalastchi, Meir Kalech, and Lior Rokach. Sensor fault detection and diagnosis for autonomous systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 15–22. Citeseer, 2013. 7

Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020. 62

Seong Soo Kim and AL Narasimha Reddy. Image-based anomaly detection technique: algorithm, implementation and effectiveness. *IEEE Journal on Selected Areas in Communications*, 24 (10):1942–1954, 2006. 11, 13

Peter Klein. The safety-bag expert system in the electronic railway interlocking system elektra. In *Operational Expert System Applications in Europe*, pages 1–15. Elsevier, 1991. 2

Trupti M Kodinariya and Prashant R Makwana. Review on determining number of cluster in k-means clustering. *International Journal*, 1(6):90–95, 2013. 32

János Kontos, Ágnes Vathy-Fogarassy, and Balázs 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)*, pages 000031–000036. IEEE, 2021. 24

A Krizhevsky. Learning multiple layers of features from tiny images. *Master's thesis, University of Toronto*, 2009. 20

Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *Unpublished manuscript*, 40(7):1–9, 2010. 30

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. In *Technical report, University of Toronto*. Toronto, ON, Canada, 2009. 61

Christopher B Kuhn, Markus Hofbauer, Goran Petrovic, and Eckehard Steinbach. Introspective black box failure prediction for autonomous driving. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1907–1913. IEEE, 2020. 14

Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018. 8

Charline Le Lan and Laurent Dinh. Perfect density models cannot guarantee anomaly detection. *Entropy*, 23(12):1690, 2021. 54

Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015. 31

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, 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*, pages 4681–4690, 2017. 18

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018a. 62

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018b. 11, 12, 15, 20

Binbin Li, Xuesu Xiao, Yulin Zhang, Haifeng Li, and Hongpeng Wang. Camera-imu extrinsic calibration quality monitoring for autonomous ground vehicles. *IEEE Robotics and Automation Letters*, 2022. 17

Chau Yi Li, Ricardo Sánchez-Matilla, Ali Shahin Shamsabadi, Riccardo Mazzon, and Andrea Cavallaro. On the reversibility of adversarial attacks. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 3073–3077. IEEE, 2021. 18

Guanpeng Li, Yiran Li, Saurabh Jha, Timothy Tsai, Michael Sullivan, Siva Kumar Sastry Hari, Zbigniew Kalbarczyk, and Ravishankar Iyer. Av-fuzzer: Finding safety violations in autonomous driving systems. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–36. IEEE, 2020. 40

Kun-Lun Li, Hou-Kuan Huang, Sheng-Feng Tian, and Wei Xu. Improving one-class svm for anomaly detection. In *Proceedings of the 2003 international conference on machine learning and cybernetics (IEEE Cat. No. 03EX693)*, volume 5, pages 3077–3081. IEEE, 2003. 55

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018a. 11, 15, 20, 31, 61

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations (ICLR)*, 2018b. 39

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(4):569–583, 2013. 11, 13, 18

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014a. 6

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014b. 45

Jiangchao Liu, Liqian Chen, Antoine Mine, and Ji Wang. Input validation for neural networks via runtime local robustness verification. *arXiv preprint arXiv:2002.03339*, 2020a. 11, 16, 20, 25

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020b. 62

Yaoyao Liu, Bernt Schiele, and Qianru Sun. Adaptive aggregation networks for class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2544–2553, 2021a. 19

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. Swin transformer v2: Scaling up capacity and resolution. *arXiv preprint arXiv:2111.09883*, 2021b. 6

Zhen Liu, Wenjie Lin, Xinpeng Li, Qing Rao, Ting Jiang, Mingyan Han, Haoqiang Fan, Jian Sun, and Shuaicheng Liu. Adnet: Attention-guided deformable convolutional network for high dynamic range imaging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 463–470, 2021c. 18

Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters*, 5(2):3153–3160, 2020. 11, 17, 23

Anna Lukina, Christian Schilling, and Thomas A Henzinger. Into the unknown: Active monitoring of neural networks. In *International Conference on Runtime Verification*, pages 42–61. Springer, 2021. 11, 14, 20

Mathilde Machin, Jérémie Guiochet, Hélène Waeselynck, Jean-Paul Blanquart, Matthieu Roy, and Lola Masson. SMOF: A safety monitoring framework for autonomous systems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 48(5):702–715, 2018. 2, 9, 10

Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018. 12

Razvan Marinescu, Daniel Moyer, and Polina Golland. Bayesian image reconstruction using deep generative models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 18

Marcin Marszalek and Cordelia Schmid. Semantic hierarchies for visual object recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–7. IEEE, 2007. 12

José Mena, Oriol Pujol, and Jordi Vitrià. A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. *ACM Computing Surveys (CSUR)*, 54(9): 1–35, 2021. 12

Christian Micheloni and Gian Luca Foresti. Active tuning of intrinsic camera parameters. *IEEE transactions on automation science and engineering*, 6(4):577–587, 2009. 7

Dimitrios Milios, Raffaello Camoriano, Pietro Michiardi, Lorenzo Rosasco, and Maurizio Filippone. Dirichlet-based gaussian processes for large-scale calibrated classification. *Advances in Neural Information Processing Systems*, 31, 2018. 12

Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pages 43–55. Springer, 2019. 52

Sina Mohseni, Mandar Pitale, Vasu Singh, and Zhangyang Wang. Practical solutions for machine learning safety in autonomous vehicles. In *Proceedings of the Workshop on Artificial Intelligence Safety*, pages 162–169, 2020. 1, 13

Mukesh C Motwani, Mukesh C Gadiya, Rakhi C Motwani, and Frederick C Harris. Survey of image denoising techniques. In *Proceedings of GSPX*, volume 27, pages 27–30. Proceedings of GSPX, 2004. 8

Khan Muhammad, Amin Ullah, Jaime Lloret, Javier Del Ser, and Victor Hugo C de Albuquerque. Deep learning for safe autonomous driving: Current challenges and future directions. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 4

Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018. 12

Robert Myers and Zeyn Saigol. Pass-fail criteria for scenario-based testing of automated driving systems. *arXiv preprint arXiv:2005.09417*, 2020. 22

Joseph Ndong and Kavé Salamatian. Signal processing-based anomaly detection techniques: a comparative analysis. In *Proc. 2011 3rd International Conference on Evolving Internet*, pages 32–39, 2011. 11, 13

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`. 61

Zhangkai Ni, Kai-Kuang Ma, Huanqiang Zeng, and Baojiang Zhong. Color image demosaicing using progressive collaborative representation. *IEEE Transactions on Image Processing*, 29: 4952–4964, 2020. 18

NVidia. CARLA scenario runner. `https://carla-scenariorunner.readthedocs.io/en/latest/`, 2022. [Online; accessed 23-June-2022]. 45

Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, John Duchi, and Russ Tedrake. Scalable end-to-end autonomous vehicle testing via rare-event simulation. In *Advances in neural information processing systems (NeurIPS)*, 2018. 40

Umit Ozguner, Christoph Stiller, and Keith Redmill. Systems for safety and autonomous behavior in cars: The darpa grand challenge experience. *Proceedings of the IEEE*, 95(2):397–412, 2007. 10

Conrad Pace and Derek Seward. A safety integrated architecture for an autonomous safety excavator. In *International Symposium on Automation and Robotics in Construction*, volume 2, 2000. 2

Zoe Papakipos and Joanna Bitton. Augly: Data augmentations for robustness, 2022. 62

Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China*, pages 1–18. ACM, 2017. 40

Dung Phan, Nicola Paoletti, Radu Grosu, Nils Jansen, Scott A. Smolka, and Scott D. Stoller. Neural simplex architecture, 2019. 17

Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215 – 249, 2014. ISSN 0165-1684. 5

Cristiano Premebida, Rares Ambrus, and Zoltan-Csaba Marton. Intelligent robotic perception systems. In Efren Gorrostieta Hurtado, editor, *Applications of Mobile Robots*, chapter 6. IntechOpen, Rijeka, 2018. 1

Frédéric Py and Félix Ingrand. Dependable execution control for autonomous robots. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 2, pages 1136–1141. IEEE, 2004. 2

Rui Qian, Robby T Tan, Wenhan Yang, Jiajun Su, and Jiaying Liu. Attentive generative adversarial network for raindrop removal from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2482–2491, 2018. 18

Stephan Rabanser, Stephan Günnemann, and Zachary Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. *Advances in Neural Information Processing Systems*, 32, 2019. 23

Sadegh Rabiee and Joydeep Biswas. Ivoa: Introspective vision for obstacle avoidance. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1230–1235. IEEE, 2019. 16

Quazi Marufur Rahman, Niko Sünderhauf, and Feras 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)*, pages 3748–3753. IEEE, 2019. 11, 14, 20, 21

Quazi Marufur Rahman, Peter Corke, and Feras Dayoub. Run-time monitoring of machine learning for robotic perception: A survey of emerging trends. *IEEE Access*, 9:20067–20075, 2021. 4

Manikandasriram Srinivasan Ramanagopal, Cyrus Anderson, Ram Vasudevan, and Matthew Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018. 11, 17

Vignesh Ramanathan, Congcong Li, Jia Deng, Wei Han, Zhen Li, Kunlong Gu, Yang Song, Samy Bengio, Charles Rosenberg, and Li Fei-Fei. Learning semantic relationships for better action retrieval in images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1100–1109, 2015. 11, 12

Amir Rasouli, Iuliia Kotseruba, and John 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*, pages 0–0, 2018. 8

Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015. 56

Vincenzo Riccio and Paolo Tonella. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 876–888, 2020. 37

Stephen Roderick, Brian Roberts, Ella Atkins, and Dave Akin. The ranger robotic satellite servicer and its autonomous software-based safety system. *IEEE Intelligent Systems*, 19(5): 12–19, 2004. 2

Alina Roitberg, Ziad Al-Halah, and Rainer Stiefelhagen. Informed democracy: voting-based novelty detection for action recognition. *British Machine Vision Conference*, 2018. 11, 16, 20

Simone Rossi, Pietro Michiardi, and Maurizio Filippone. Good initializations of variational bayes for deep models. In *International Conference on Machine Learning*, pages 5487–5497. PMLR, 2019. 12

Giulio Rossolini, Federico Nesti, Gianluca D'Amico, Saasha Nair, Alessandro Biondi, and Giorgio Buttazzo. On the real-world adversarial robustness of real-time semantic segmentation models for autonomous driving. *arXiv preprint arXiv:2201.01850*, 2022. 41

Anirban Roy, Adam Cobb, Nathaniel D Bastian, Brian Jalaian, and Susmit Jha. Runtime monitoring of deep neural networks using top-down context models inspired by predictive processing and dual process theory. *AAAI 2022 Workshop on Designing Artificial Intelligence for Open Worlds*, 2022. 16

John Rushby. Kernels for safety. *Safe and Secure Computing Systems*, pages 210–220, 1989. 2

Mohammad Sabokrou, Mohammad Khalooei, Mahmood Fathy, and Ehsan Adeli. Adversarially learned one-class classifier for novelty detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3379–3388, 2018. 11, 14, 20, 31, 37, 39, 55

Chitwan Saharia, William Chan, Huiwen Chang, Chris A Lee, Jonathan Ho, Tim Salimans, David J Fleet, and Mohammad Norouzi. Palette: Image-to-image diffusion models. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 18

Rick Salay, Matt Angus, and Krzysztof Czarnecki. A safety analysis method for perceptual components in automated driving. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 24–34. IEEE, 2019a. 10

Rick Salay, Matt Angus, and Krzysztof Czarnecki. A safety analysis method for perceptual components in automated driving. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, pages 24–34. IEEE, 2019b. 18, 22, 24

Christoph Schorn and Lydia Gauerhof. Facer: A universal framework for detecting anomalous operation of deep neural networks. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020. 11, 15, 20

Adrian Schwaiger, Poulami Sinhamahapatra, Jens Gansloser, and Karsten Roscher. Is uncertainty quantification in deep learning sufficient for out-of-distribution detection? In *AISafety@ IJCAI*, 2020. 54

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. In *Advances in neural information processing systems (NeurIPS), Montreal, Canada*, pages 2503–2511, 2015. 5

Francesco Secci and Andrea Ceccarelli. On failures of rgb cameras and their effects in autonomous driving applications. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 13–24. IEEE, 2020. 39, 45

Matthias Seeger. Bayesian modeling in machine learning: A tutorial review. *infoscience.epfl.ch*, 2006. 12

Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 3183–3193, 2018. 12

Pierre Sermanet and Yann LeCun. Traffic sign recognition with multi-scale convolutional networks. In *The 2011 International Joint Conference on Neural Networks*, pages 2809–2813. IEEE, 2011. 30

Alireza Shafaei, Mark Schmidt, and James J Little. A less biased evaluation of out-of-distribution sample detectors. *30th British Machine Vision Conference, Cardiff, Wales*, 2019. 20, 39

Sina Shafaei, Stefan Kugele, Mohd Hafeez Osman, and Alois Knoll. Uncertainty in machine learning: A safety perspective on autonomous driving. In *International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, pages 458–464, 2018. 6

Zheyan Shen, Jiashuo Liu, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey. *arXiv preprint arXiv:2108.13624*, 2021. 5, 7

Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. 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*, pages 1874–1883, 2016. 18

Vasu Singh, Siva Kumar Sastry Hari, Timothy Tsai, and Mandar Pitale. Simulation driven design and test for safety of ai based autonomous vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 122–128, 2021. 40

Amolika Sinha, Sai Chand, Vincent Vu, Huang Chen, and Vinayak Dixit. Crash and disengagement data of autonomous vehicles on public roads in california. *Scientific data*, 8(1):1–10, 2021. 7

Andrea Stocco and Paolo Tonella. Towards anomaly detectors that learn continuously. In *2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW)*, pages 201–208. IEEE, 2020. 14, 22

Andrea Stocco, Michael Weiss, Marco Calzana, and Paolo Tonella. Misbehaviour prediction for autonomous driving systems. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, pages 359–371, 2020. 11, 14, 18, 22, 23

Andrea Stocco, Brian Pulfer, and Paolo Tonella. Mind the gap! a study on the transferability of virtual vs physical-world testing of autonomous driving systems. *arXiv preprint arXiv:2112.11255*, 2021. 51

Rob Stumpf. Autopilot blamed for tesla's crash into overturned truck, Jun 2020. URL `https://www.thedrive.com/news/33789/autopilot-blamed-for-teslas-crash-into-overturned-truck`. Online; accessed 07 June 2022. 8

Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 144–157. Curran Associates, Inc., 2021. 11, 14, 20, 39

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)*, pages 1526–1533. IEEE, 2018. 7

Tecperson. Sign language mnist, version 1., 2017. URL `https://www.kaggle.com/datasets/datamunge/sign-language-mnist`. 61

Sergios Theodoridis. *Machine learning: a Bayesian and optimization perspective*. Academic press, 2015. 16

Chunwei Tian, Lunke Fei, Wenxian Zheng, Yong Xu, Wangmeng Zuo, and Chia-Wen Lin. Deep learning on image denoising: An overview. *Neural Networks*, 2020. 8

Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018. 40

Dennis Ulmer. A survey on evidential deep learning for single-pass uncertainty estimation. *arXiv preprint arXiv:2110.03051*, 2021. 12

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019. 19

Kush R Varshney and Homa Alemzadeh. On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. *Big data*, 5(3):246–255, 2017. 1

Huiyan Wang, Jingwei Xu, Chang Xu, Xiaoxing Ma, and Jian Lu. Dissector: Input validation for deep learning applications by crossing-layer dissection. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 727–738. IEEE, 2020a. 11, 15, 21

Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1245–1256. IEEE, 2019. 11, 16, 20

Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1905–1914, 2021. 14

Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020b. 18

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011. 12

Olga Wichrowska, Niru Maheswaranathan, Matthew W Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando Freitas, and Jascha Sohl-Dickstein. Learned optimizers that scale and generalize. In *International Conference on Machine Learning*, pages 3751–3760. PMLR, 2017. 31

Changshun Wu, Yliès Falcone, and Saddek Bensalem. Customizable reference runtime monitoring of neural networks using resolution boxes. *arXiv preprint arXiv:2104.14435*, 2021. 11, 15, 20

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 61

Jingyi Xu, Zilu Zhang, Tal Friedman, Yitao Liang, and Guy Broeck. A semantic loss function for deep learning with symbolic knowledge. In *International conference on machine learning*, pages 5502–5511. PMLR, 2018. 11, 13

Salisu Wada Yahaya, Ahmad Lotfi, and Mufti Mahmud. A consensus novelty detection ensemble approach for anomaly detection in activities of daily living. *Applied Soft Computing*, 83:105613, 2019. 11, 16

Qingsen Yan, Dong Gong, Javen Qinfeng Shi, Anton van den Hengel, Chunhua Shen, Ian Reid, and Yanning Zhang. Dual-attention-guided network for ghost-free high dynamic range imaging. *International Journal of Computer Vision*, pages 1–19, 2021. 18

Qingsen Yan, Dong Gong, Javen Qinfeng Shi, Anton van den Hengel, Jinqiu Sun, Yu Zhu, and Yanning Zhang. High dynamic range imaging via gradient-aware context aggregation network. *Pattern Recognition*, 122:108342, 2022. 18

Jian Yang, David Zhang, Alejandro F Frangi, and Jing-yu Yang. Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 26(1):131–137, 2004. 32

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *preprint arXiv:2110.11334*, 2021. 6, 7

Xin-She Yang. Chapter 5 - genetic algorithms. In Xin-She Yang, editor, *Nature-Inspired Optimization Algorithms*, pages 77–87. Elsevier, Oxford, 2014. ISBN 978-0-12-416743-8. 46

Haotian Ye, Chuanlong Xie, Tianle Cai, Ruichen Li, Zhenguo Li, and Liwei Wang. Towards a theoretical framework of out-of-distribution generalization. *Advances in Neural Information Processing Systems*, 34, 2021. 3

Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 61

Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation. In *European Conference on Computer Vision (ECCV)*, volume 1, 2021. 6

Lars Zandbergen. Evaluating the simulation gap for training off-road self-driving. Master's thesis, University of Amsterdam, Amsterdam, The Netherlands, 2021. 23

Eleni Zapridou, Ezio Bartocci, and Panagiotis Katsaros. Runtime verification of autonomous driving systems in carla. In *International Conference on Runtime Verification*, pages 172–183. Springer, 2020. 40

Cheng Zhang and Pan Gao. Countering adversarial examples: Combining input transformation and noisy training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 102–111, 2021. 18

Jialiang Zhang, Lixiang Lin, Jianke Zhu, Yang Li, Yun-chen Chen, Yao Hu, and CH Steven Hoi. Attribute-aware pedestrian detection in a crowd. *IEEE Transactions on Multimedia*, 2020. 8

Lily Zhang, Mark Goldstein, and Rajesh Ranganath. Understanding failures in out-of-distribution detection with deep generative models. In *International Conference on Machine Learning*, pages 12427–12436. PMLR, 2021. 54

Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz 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)*, pages 132–142. IEEE, 2018. 16, 20, 40

Xibin Zhang, Maxwell L King, and Rob J Hyndman. A bayesian approach to bandwidth selection for multivariate kernel density estimation. *Computational Statistics & Data Analysis*, 50(11): 3009–3031, 2006. 56

Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2. IEEE, 2018. 31

Wei Zhou, Julie Stephany Berrio, Stewart Worrall, and Eduardo Nebot. Automated evaluation of semantic segmentation robustness for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 21(5):1951–1963, 2019. 11, 17, 20

Wei Zhou, Ling Zhang, Shengyu Gao, and Xin Lou. Gradient-based feature extraction from raw bayer pattern images. *IEEE Transactions on Image Processing*, 30:5122–5137, 2021. 18

Yuqian Zhou, Jianbo Jiao, Haibin Huang, Yang Wang, Jue Wang, Honghui Shi, and Thomas Huang. When awgn-based denoiser meets real noises. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13074–13081, 2020. 18