Can we reconcile safety objectives with machine learning performances?

Lucian Alecu, Continental Lucian.Alecu@continental-corporation.com
Hugues Bonnin, Continental hugues.bonnin@continental-corporation.com
Thomas Fel, SNCF thomas.fel@sncf.fr
Laurent Gardes, SNCF l.gardes@sncf.fr
Sébastien Gerchinovitz, IRT Saint Exupéry and IMT sebastien.gerchinovitz@irt-saintexupery.com
Ludovic Ponsolle, Apsys Airbus ludovic.ponsolle@apsys-airbus.com
Franck Mamalet, IRT Saint Exupéry franck.mamalet@irt-saintexupery.com
Eric Jenn, IRT Saint Exupéry and Thalès Avionics eric.jenn@irt-saintexupery.com
Vincent Mussot, IRT Saint Exupéry vincent.mussot@irt-saintexupery.com
Cyril Cappi, SNCF cyril.cappi@sncf.fr
Kevin Delmas, ONERA Kevin.Delmas@onera.fr
Baptiste Lefevre, Thalès baptiste.lefevre@fr.thalesgroup.com

Abstract— The strong demand for more automated transport systems with enhanced safety, in conjunction with the explosion of technologies and products implementing machine learning (ML) techniques, has led to a fundamental questioning of the trust placed in machine learning. In particular, do state-of-the-art ML models allow us to reach such safety objectives? We explore this question through two practical examples from the railway and automotive industries, showing that ML performances are currently far from those required by safety objectives. We then describe and question several techniques aimed at reducing the error rate of ML components: model diversification, monitoring, classification with a reject option, conformal prediction, and temporal redundancy. Taking inspiration from a historical example, we finally discuss when and how new ML-based technologies could be introduced.

Keywords— machine learning, safety, certification, probabilistic assessment, trustworthiness.

I. INTRODUCTION

Recent efforts in developing next generation transportation systems bet on significant contributions from machine learning-based predictive models to implement highly automated functions. Yet, despite the impressive progress seen in recent years, many questions remain open in regard to the capacity of such statistical models to comply with existing safety standards. Here we focus on one such central question: is the error rate of current state-of-the-art ML models low enough to allow the implementation of safety-critical functions?

In this paper, we explore this question through two examples from the railway and automotive industries and make the following contributions:
- In Section II, we note that the performances of ML-based detection systems used in each of the two use-cases is orders of magnitude lower than that required to reach safety objectives (or at least for now, or to reach it directly, i.e., without non-ML software components).
- In Section III, we describe potential directions for safe integration of ML models: can we use several redundant ML models in parallel? Can the ML component be monitored by rejecting specific inputs or changing the (confidence of the) outputs when needed? Can we improve ML performances with temporally redundant inputs? We report experimental results from the ML literature indicating that these solutions can be useful, but are currently not at all sufficient to reach safety objectives. Since some of these techniques reduce the availability of the ML component, we also discuss the availability-reliability trade-off that naturally appears.
- Finally, in Section IV, we take a historical perspective on how technologies were introduced in the past and try to relate it to (and differentiate it from) how ML-based technologies can be introduced.

From a pedagogical viewpoint, this paper targets a mixed audience of safety engineers and ML researchers. We hope it can help highlight some key concepts and refocus research efforts on relevant topics from the safety perspective.
In this document we focus our analysis exclusively on the performance of ML models, since they have recently shown considerable success in many complex tasks, and they currently represent the dominant AI paradigm used in many industrial applications. Other AI-flavored techniques are not discussed here, even if they may partly address some of the issues raised in this paper.

Related works. The question of ML reliability and trustworthiness for integration to safety-critical systems has received a lot of attention recently. Many projects, institutes or working groups (e.g., ANITI, DEEL, SafeAI, the EUROCAE WG-114 and SAE G-34, ISO TC22/SC32/WG14, ISO-IEC JTC1/SC42/WG3, RISE SMILE), as well as workshops or conferences (e.g., AAAI-SafeAI, MLCS, WAISE, ERTS, Future Intelligence) were created to address these challenges. Though several reports and papers have been published very recently (e.g., white paper from DEEL [1], CoDANN II [2], [3]), many questions remain open.
Authors from the DEEL project [3] provide a thorough review of existing methods aimed towards the certification of ML-based software in critical systems.
Work by authors from RISE SMILE project series [4] provides another review of verification and validation methods that aim to embed ML technologies into automotive critical systems. The authors highlight the gap between current standards and the ML-based software engineering and conclude that "ISO 26262 largely contravenes the nature of DNNs". They enumerate and discuss several challenges and directions towards new methods and norms for certifying critical systems based on ML.

A similar assessment is provided by [5], which claims that ISO 26262 cannot appropriately manage ML-based software and propose new measures to adapt the current norms.
From the side of ML literature, many papers have addressed properties such as generalization, robustness, or explainability [6], [7], [8], [9], [10], yet we argue that none of these results alone can layout the necessary safety foundations for ML-based critical systems, at least for now.

II. FROM ML PERFORMANCE TO SAFETY OBJECTIVES

A. ML performance and safety objectives

In ML parlance a model refers to a software implementation of some stochastic function which provides predictions on unseen samples from a given input domain. In the context of system engineering, a function implements a decision to be taken, given the available information and constraints. Its role is to implement the necessary logic in order to alter or to maintain the current state of the system, according to a given specification.
Safety analysis seeks to establish causal links between the erroneous components, the decisions taken by the system and the foreseeable failures.
Safety objectives can be formulated in terms of acceptable failure rates for each critical function implemented by a system. In industrial standards the acceptable failure rates are expressed as number of average failures per hour of operation. In addition, assurance levels are considered: e.g., DAL (Design Assurance Level) for aviation industry, as introduced by ARP4754 [11] [12] [13] [14], SIL (Safety Integrity Level) for industry in IEC 61508 standard [15], adapted for Railway systems in EN 50126 [16] [17] [18], and tailored in ASIL (Automotive Safety Integrity Level) in ISO 26262 [19]. Least stringent levels of performance require less than 10^{-3} failures per hour for such functions, while the strictest one generally require less than 10^{-9} failures per hour on average[1].
In order to assess whether an ML-based function complies with the safety objectives it is thus essential to estimate its failure rate. While everyone can acknowledge that there is a close dependency between the error rate usually reported in the ML literature and the failure rate of the system relying on ML predictions in operation, the exact relationship is far from obvious to describe analytically. Moreover, this relation is use-case dependent.
In this section we consider ML-based components for which every decision relies on a single prediction provided by the ML model. To conduct a rigorous safety assessment of a ML-based component it is key to estimate its error rate per hour of operation of the system, based on some set of assumptions and empirical observations.
We can draw here a parallel between the methods used to compute error rates for hardware components and for ML-based components. At first glance the comparison seems appropriate as both types of components are characterized by stochastic processes.
For most ML models these estimates can be computed from observational data, by sampling the operational domain. Depending on the sampling schemes and other assumptions used to compute it, statistical guarantees can be obtained with respect to the gap between the empirical error rate (i.e. computed from the sampled observations) and the true error rate (the one that would have resulted by performing the computation on all possible observations of the entire operational domain).
On the other hand, error rates of hardware components are reported after performing experiments in various regimes (normal operation, operation under stress, accelerated aging, etc.) and calibrating some expected distributions for the error rates as to closely match this empirical evidence. Failure rates of the system embedding them are then estimated according to various failure analysis tools and methods (FTA, FMEA, Human Factor Analysis, Functional Hazard Analysis, etc.). Most of these methods rely on assumptions and practices that must be used with a lot of attention for ML components (e.g., sub-components may not be independent as shown in Section III.A, the error distribution in operation is unknown, etc). Thus, in order to make accurate estimations of error rates for ML-based functions, one must rely almost exclusively on empirical observations (for now) of the actual behavior of these functions in operation-like scenarios. In order for these to be statistically significant, they require a large number of samples to be collected in real-life scenarios. For example, this would require at least 100 000 hours of operational testing (obviously simulation can help to accelerate the time) to decide whether the expected failure rate of a ML-based system function is lower than 10^{-5} errors per hour, and this under the stringent assumption that all the predictions provided by the ML model are i.i.d. over training and operation. The further we are from these assumptions, the more samples we would need to gather in order to reach such conclusions. Unless prior knowledge regarding mathematical properties of the operational domain and / or of the ML model, we are required to perform extensive testing in order to be able to guarantee statistically sound estimations of the error rate. In spite of these challenges, it can be useful to estimate the error rate under idealized assumptions. If the safety objectives are met in these conditions, further refinements (i.e., under realistic assumptions) should be conducted, but if they are far from being satisfied, the negative conclusion is a good indication that safety objectives are difficult to prove in realistic conditions.

B. Two examples from the railway and automotive domains

To illustrate the large gap between the performance of ML system and the targeted safety objectives, let us consider two examples: a computer vision system used in the railway domain, and a weather prediction system used in the automotive domain.

1) Railway and computer vision

Functional description. In our first example, we consider a railway automatic signal reading system. This system aims at recognizing the state of a light signal applicable to a train, a task that is currently performed by train drivers.
Figure 1 gives an overview of the actions performed by a train driver. The system shall determine the indication of the signal automatically. The system must be able to operate in any environmental conditions in the open world of a typical railway infrastructure. This includes scenarios where the signal can be partially occluded by vegetation or due to some weather condition (fog, snow, etc.) or damaged. The recognition system relies on a ML-based computer vision model. The system takes in input an image (coming from a camera in front of the train) in which the approximate location of the signal is

[1] ML technologies are not completely covered by any safety standard/norm today, even recent ones like SOTIF [53]. In this paper, we consider that, due to its probabilistic nature, it makes sense to allocate to the ML components some probabilistic objectives, based on the analogy with HW components, or the system level components.

known. This allows us to use a bounding box and thus to limit the detection effort of the signal in the complete image. In the end, the ML model has to cope with a small image containing only one signal that shall be classified.
This use-case features two interesting properties with respect to the objective of our study: it implements a simple task (recognizing a light signal) and it operates in an open and weakly structured environment.

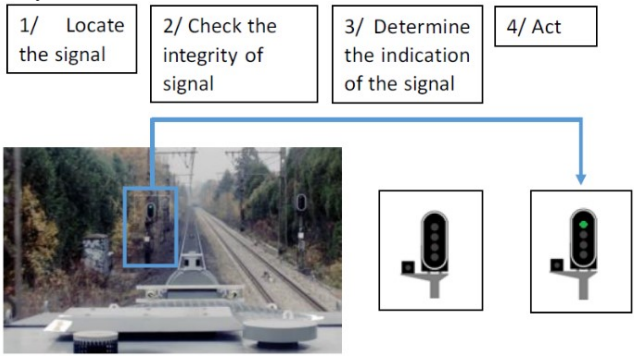

Figure 1 : Action performed by a train driver.

Safety objective. The analysis of the current human-operated system leads to an average target of maximum 10^{-5} failures per hour, which is the observed number of failed recognitions of a red signal per hour of driving on average. This in turn corresponds to 10^{-5} failures for red signal recognition if we consider the number of red signals that a train driver usually encounters during a journey, which is one red signal per hour on average. The red signals are the most critical ones, and this consideration determines the safety objectives that we consider here for the ML system that shall recognize the signals, in particular red signals.
ML performances compared with safety objective. As of today, in the performance results carried out on test sets deemed representative of real-world situations the best classification models achieve on average 10^{-2} errors per signal.
Compared with the safety objective, the order of magnitude of the observed performance is clearly insufficient.

2) Automotive and weather prediction

Functional description. Let us consider an assisted driving service using weather data to enhance the safety of the driving experience. Concretely, this service predicts the state of the road surface (icy, wet or dry) from various weather-related indicators (e.g., air temperature, humidity and water level on the road surface) available at any given location. This ML-based predictive system is used to implement an assisted speed management system as a highly automated driving function (HAD). The system takes control from the driver and handles the vehicle accordingly in scenarios deemed safe, while handling the control to the driver in the remaining situations.

Safety objective. The proposed ML-based service is expected to introduce some risks, especially when the predictions of the road conditions are erroneous (e.g., the road surface is deemed dry instead of wet or icy and the vehicle moves at high speeds). In such scenarios, due to its autonomous nature, the HAD function reduces the controllability of the vehicle, which potentially increases the risks of collisions. For this reason, the expected failure rate that would qualify as minimally safe by current automotive standards is 10^{-5} errors per hour[2].

ML performances compared with safety objective. The described service relies on an ML-based prediction model which computes periodically the road conditions in the near future based on recently observed weather-related data, for each square tile of size S of the Earth's surface. Each vehicle having enabled such a service would make a request for a new prediction periodically, depending on its geographical position. A very simplistic model in which the vehicle is considered to move at a constant speed, a periodic tile change, and an error rate per prediction (ML error rate) of 0.01, leads to a failure rate of 0.5 failure per hour. Again, to reach the required safety objectives, it would be necessary for this error rate to be orders of magnitude lower.

C. The Gap

We concluded in the two use-cases above that the ML performances are several orders of magnitude below those that would be required to reach the safety objectives. Such a gap should not be omitted by the new guidelines aiming for the integration of ML into safety-critical systems.
In addition, neither rigorous software development efforts nor properly documented data collection rules are sufficient to reduce the gap between the actual performance of ML components and the safety objectives. Even a properly coded ML model trained on properly collected data could lead to unacceptable error rates—at least for now.
Does this mean there are no hopes for integration of ML components into safety-critical systems? We believe not but urge to pursue research efforts.

III. Reconciling Safety Objectives with Machine Learning Performances?

Generally speaking, the prediction performances of ML models can be enhanced either by improving the data management and processing or the model's engineering.
Enhancing performances with data can be done by collecting more data, by improving their quality, by including more diverse sources of data (e.g., sensors), by exploiting certain constraints or properties of the operational domain, etc. Obviously, more data usually means dealing with more uncertainty, therefore obtaining overall better performances is not guaranteed.
From the model's engineering viewpoint, we can apply various techniques to improve, e.g., robustness, generalization or explainability of the models.
Next, we focus on possible solutions to decrease the error rate of ML components. We first question the possibility of using several redundant ML systems in parallel (Section A). Then, we describe methods that either detect inputs potentially leading to erroneous predictions (Sections B and C.1), or that are allowed to output less precise predictions for hard-to-classify inputs (Section C.2). We finally question the possibility of using temporally redundant inputs (Section D).
Part of these methods reduce the ML model's availability by, e.g., ignoring its predictions in some situations. We thus also discuss the availability-reliability trade-off that arises there.

A. Model diversification

Using several redundant components in parallel is a classical way to reduce the failure rate of a system when the

[2] This requirement mainly arises from two assumptions: 1) as explained in II.A a probabilistic objective is allocated to the ML component 2) the predicted state of the road is not the only information used by the envisaged system, this particular contribution being considered as ASIL C or B.

components' failures are independent (and when the costs incurred by the additional components are justified from the safety perspective). Though it is very natural to implement redundant ML systems, it is now known that the independence assumption is hard to ensure in practice. As an illustration, Figure 2 below shows on the ImageNet dataset how the joint error rate of n ML models in parallel would decrease as a function of n if the models' errors were independent (exponential decrease, in blue), and how this joint rate actually decreases in practice (slow decrease, in red). For any value of n, we considered the first n models in the list {EfficientNet,DenseNet121,ResNet50V2, MobileNet,VGG16models}, which are state-of-the-art deep learning models for this dataset. We say we have a joint error if all n models make a prediction error simultaneously.

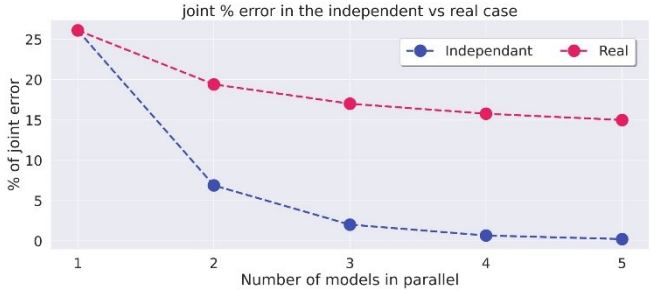

Figure 2: joint error rate versus number of redundant ML models

Of course, we cannot conclude from Figure 2 that it is impossible to build independent (and accurate) deep learning models. What if we tried to get more diverse models, such as models trained with different hyperparameters, loss functions or optimization procedures, models with different architectures, or models trained on different datasets? Though somewhat negative answers were formulated by [20] (about standard ImageNet deep learning models) and by [21] (about bootstrapping deep learning models), the very recent paper [22] provides a more positive answer. The authors show that significantly different training methodologies can in fact lead to models with partially uncorrelated errors or, equivalently, models that err on partially disjoint test points. More precisely, for a pair of models, let us call "error inconsistency" the proportion of test points for which only one model errs (while the other is correct). On ImageNet, the authors show that two different optimization objectives (supervised versus contrastive learning) can yield an error inconsistency of around 13%, while it can go up to approximately 20% for models that are trained on two different datasets.

Though the results of [22] indicate that it is possible to build diverse models, they do not reach the diversity level that independent models would do: for two models with around $p = 25\%$ test error rate (as was the case in that paper), if the two models were independent, the error inconsistency would be of $2p(1-p) = 0.375$, which is larger than the proportions of 13% or 20% mentioned above. Nonetheless, as shown by [22], producing diverse ensembles of deep learning models can help increase the ensemble test accuracy. For instance, two very diverse models trained with supervised learning on the one hand and contrastive learning on the other hand, both with about 75-76% test accuracy on ImageNet, were shown to achieve about 83% after ensembling. The main reason seemed that these diverse models specialize to two (partially) different subdomains of the data. Note that the gain in accuracy is however not as high as the one we could hope to get with independent models and remains orders of magnitude worse than that prescribed by safety objectives.

Two natural yet important-for-safety remarks should be made. First, the fact that two ML models do not have independent test errors is not at all surprising, since the two models are evaluated on the same test points, with the same inputs. There can be test points that are easy to classify with both models, while other points can be hard to classify for both models (e.g., occluded images for which the ground truth is not at all readable), even if these two models were trained on different datasets. This indicates that their failure modes are typically not independent. Second, if we accept to work only with `partially independent' models, estimating their error correlation accurately seems at least as hard as estimating the error rate of the ensemble itself (that is, the model after consolidation). It is thus not obvious a priori how one could leverage partial independence to prove safety at system level by combining low-reliability ML components. While the independence assumption is common in many safety analysis methods when combining various components, and helps prove a small system failure rate by, e.g., multiplying individual error rates, the formula to combine individual error rates is not known a priori when ML is involved.

B. Monitoring

Monitoring techniques are common means for detecting unexpected inputs or conditions that could lead to erroneous output of a system. Once those events are detected, prevention, recovery, or passivation actions can be taken to prevent an unsafe failure of the system. Those events may be due to the activation of intentional (cyber attack with malicious purpose) or unintentional errors, as shown in the following table.

Category	**External**	**Internal**
Intentional	Spoofing, adversarial attacks	*Not relevant*
Unintentional	Out Of Distribution inputs	Lack of robustness, Inconsistent behavior

Different monitoring approaches can be used:

- **ODD (Operational Design Domain) Monitoring** [23] verifies that the ML-based system is operated in its usage domain (e.g., range of brightness, temperature, speed…)
- **OOD (Out Of Distribution) Monitoring** [24] [25] ensures that the ML Model operates in the distribution defined during the training process. Monitoring techniques include distance-based approaches, One-class classification, probabilistic approaches, reconstruction approaches, etc.
- **Attacks monitoring** [26] allows to detect adversarial attacks. It can be based on Input source diversification, properties imposed at design phase, ML adversarial detector, prediction inconsistency, etc.
- **Robustness monitoring** ensures that the ML Model is used in a stable area. Robustness can be verified using constraint programming, abstract interpretation, geometrical approaches, statistical approaches, etc.
- **Consistency monitoring** analyzes the consistency of outputs with the inputs. In particular, inconsistent sequences of outputs can be detected using Detection of unstable states, Functional rules, etc.

In this paper, for reason of space, we will focus on OOD and Robustness monitoring, presenting an overview of State-of-the-Art techniques and their application for monitoring.

1) OOD Monitoring

The monitoring of Out-of-distribution (OOD) is a complex topic, often related to the detection of anomalies and to the underlying question of input normality. OOD inputs correspond to samples that seem to be drawn from a distribution different from the one used during training. Indeed, the use of a training data distribution representative of the real-world data is crucial for the quality of the learning process and the adequacy with the operational domain. However, the capabilities of the trained model to fulfill its task and the probabilistic guarantees on the model performances can only be ensured for the same type of data that it has seen during training. Therefore, OOD monitoring focuses on the detection of inputs that do not correspond to the normality represented by this training distribution. This led to families of OOD detection techniques relying on an estimation of the similarity between a sample and the ones seen during training, such as distance-based approaches [27] or one-class classification [28]. This kind of techniques can be used for obvious tasks like detecting major anomalies in the data, for instance when the expected task cannot be performed on the input (e.g., a picture of a dog given to classifier trained on digits). However, it can be harder to specify for elements belonging to the tail of the distribution, where we could also expect to rely on the generalization capability of the model. For this reason, these techniques always use a form of threshold on the similarity to discriminate OOD from ID inputs, and this choice of threshold will have a direct impact on the availability of the function.

Furthermore, we also need to consider the question of the performance of OOD detection methods, which is addressed in a few papers [29, 30]. These articles compare the various techniques of the literature on dedicated datasets, typically with various images corruptions and noises, or representing industrial defects. It is worth noting that one major result of these comparisons is the inconsistency of most techniques, which may perform exceptionally well on specific types of simple corruptions (up to 100% detection) but may perform below average on others. Moreover, on complex tasks, or in high dimension, the average accuracy of the best methods remains particularly low, typically below 80% for realistic settings.

In the railway use case, we considered primarily the detection of OOD inputs as a means to detect unreadable signals, which could be caused by several things:

- An occlusion by an environmental element (e.g., a pole, a bird, or a tree) which makes the signal partially or entirely unreadable. We also consider other environmental elements such as fog or brightness issues, even if these could be easily associated with other types of monitoring such as the ones based on robustness.
- A failure in the cropping algorithm (e.g., crop of a wrong area of the picture, an issue in the crop size, a crop of the wrong signal, etc.), which may result in the absence of the expected signal on the input
- A defective sensor, (e.g., a failure of the camera lens), could be detected with a robustness monitoring, but the resulting ML images will be unreadable, and the OOD detection should trigger as well in this case.

In fact, most of these root causes can be monitored separately from the ML component, but the OOD detection appears to be valuable, as it can cover all these cases independently from the cause. However, the rate of potential OOD inputs is an important factor, as it is considered to be very low in our case (less than 10^{-4}). Moreover, it is important to remember that an input classified as OOD does not necessarily mean that the model will not be able to perform its task correctly, only that the model performances cannot be guaranteed. This is especially true for partial occlusions, for which there is no way to define how much of the signal needs to be occluded before considering it unreadable.

If we take into account the poor performances of OOD detection methods, we can safely consider that a method providing a true positive rate of 80% or 90% with a detection threshold chosen to have 1% false alarms is an excellent method in this realistic setting. In our use case, if we consider an expected OOD rate of 10^{-4}, the benefits of the OOD detection appear to be very limited: the few OOD inputs detected will not change significantly the accuracy of the model, but the availability will drop from 100% to 99% due to the false alarms. In this situation, the use of OOD monitoring mechanism will be damaging the availability without improving the reliability, but it should not be the same for systems with higher expected OOD rates. A general formula will be detailed in the following section with a numeric analysis on Robustness monitoring for which the conclusions are transferable to other types of monitoring.

2) Robustness Monitoring

According to the CODANN [2], two types of robustness can be considered; either the capability of the learning algorithm to produce "similar" models for "close" training datasets; or the model stability e.g., if inputs, x and x' are close then predictions f(x) and f(x') are also close. In this paper, we address the runtime monitoring of a trained ML component hence we focus the remainder of this section on the state-of-the-art approaches enabling model stability assessment.

The model stability is classically specified w.r.t. a measure $|.|$ on inputs and outputs to formally capture the "closeness", thus a stable model ensures that for any delta-close inputs (x,x'), predictions (f(x),f(x')) are epsilon-close, that is:

$$|x - x'| < \delta \Rightarrow |f(x) - f(x')| < \epsilon$$

According to [31], a poor robustness radius may indicate a use of the component in an unstable decision area hence a potential erroneous inference. This instability can typically be exploited by errors adversarial attacks. For our use case this may indicate that slight modifications of the image may change its classification.

Numerous methods propose to perform at runtime a formal robustness checking centered on the received input. Many of these methods are founded on the abstract interpretation that provides an over-approximation of the ML-component output domain for a given input domain i.e., the studied robustness ball. The main challenge encountered by these methods is the computation resources and the computation time needed to compute the output domain. We can especially consider the works [32] proposing GPU-based implementation of the abstract interpretation to speed-up the analysis process (less than 1s second for MNIST). Other works like [33] promote new abstract domains to increase both the accuracy and efficiency of the computation (1-100s for MNIST). Such approaches suffer from over-approximation, thus may consider that an input is non-robust whereas it is.

Quite different approaches like [34] provide some statistical robustness guarantees of local robustness. These methods usually rely on the sampling of a set of inputs within the studied robustness ball. Such methods could provide a quite simple way to assess the local robustness. But as the authors of [35] has identified, the main problem is to justify the representativeness of the computed statistical bounds since the sampling must be performed in conformity with the operational input distribution.

To apply these techniques, one may specify the requested local model stability that will be monitored at runtime. At the best of our knowledge, very little works propose methods to identify the requested robustness radius. One may consider that the robustness radius should be derived from the specifications of the ML component. Nevertheless, in many cases the selection of a robustness radius is difficult to argue and even more to relate to the specification. Hopefully, the authors of [31] propose a method to identify the expected robustness radius out of the training and test data sets. The capability of the requested robustness radius to identify instability areas at inference time has been assessed in their experiments.

Just like OOD monitoring, the performance of robustness-based runtime monitoring is paramount to ensure that a safety benefit can be achieved with an acceptable impact on system's availability. To address this aspect, we propose a simple formulation of the safety/availability tradeoff problem and illustrate it thanks to the experiments made by [31].

3) Safety/Availability Tradeoff

Let *S* be a simple system containing: ml, a ML component (e.g., a classifier) and m a mechanism validating (or rejecting) ml decisions (*e.g.,* an external monitor). We consider that m can only be *triggered* (*i.e.,* the prediction of ml is not trusted) or not (*i.e.,* the prediction of *ml* is trusted).

So, for a given input the system generates three possible outcomes:

- a correct output
- an erroneous-safe output, which covers a) the situation where ml generates an incorrect but safe output while *m* has not triggered, and b) the situation where the monitor triggers.
- an erroneous-unsafe output.

Let us assume that the ML system is free of implementation errors and is executed on a error-free hardware. Even with a perfect implementation and hardware we consider here that:

a) *ml* can produce an incorrect output; b) the occurrence of an error (*e.g.,* an OOD input), denoted *F*, may prevent *ml* to properly process the input.

In addition, we consider that *m* is unable to detect any failure caused by something else than *F*. This assumption is conservative since a monitoring mechanism, typically output monitoring, *may* also detect other errors like implementation or hardware errors.

The issue is to find the appropriate *sensitivity level* for *m* allowing complying with both safety and availability requirements. To assess the appropriate sensitivity level, one must formalize the notion of *unsafe* and *unavailable* system.

Let us consider that *ml* can either produce a correct result, fail safely or unsafely (event U_{ml}) and *m* can either be triggered (denoted *T*) or not. Let

- $P(U_{ml}|\neg F) = \alpha_{ml}$ be the conditional probability that the component fails unsafely knowing the failure *F* does not occur.
- $P(F) = \lambda$ be the probability of occurrence of *F* on demand
- $P(T|\neg F) = \delta$ be the false positive rate
- $P(\neg T|F) = \gamma$ be false negative rate.

a) General case

The scenarios leading to an *unsafe failure* (Unsa) are (a.1) *F* occurs and (a.2) *m* fails to detect it and (a.3) *ml* fails unsafely or (b.1) *F* does not occur and (b.2) *ml* fails unsafely and (b.3) *m* does not spuriously detect *F*, *i.e.*

$$\begin{aligned} P(Unsa) &= P(F, U_{ml}, \neg T) + P(\neg F, U_{ml}, \neg T) \\ &= P(U_{ml}, \neg T|F)\lambda + P(U_{ml}, \neg T|\neg F)(1-\lambda) \end{aligned}$$

Similarly, *S* is *unavailable* (Unav) when *m* is triggered. Among these scenarios some are expected since *ml* must not be used when *F* occurs. So, we propose to consider the scenarios where *S* is unavailable even if *F* did not occur, that is:

$$P(Unav) = P(\neg F, T) = \delta(1-\lambda)$$

b) Simplifications

Correlation between U_{ml} and F. If *ml* is likely to produce an unsafe output when *F* occurs that is $P(U_{ml}, \neg T|F) \simeq P(\neg T|F)$, then:

$$P(Unsa) = \gamma\lambda + P(U_{ml}, \neg T|\neg F)(1-\lambda)$$

Correlation between U_{ml}, $\neg T$ and $\neg F$. Additionally if we consider that the ability of *ml* to produce an unsafe output when *F* does not occur is similar when *F* does not occur and the *m* confirms it, that is $P(U_{ml}|\neg F) \simeq P(U_{ml}|\neg T, \neg F)$ then:

$$P(Unsa) = \gamma\lambda + \alpha_{ml}(1-\delta)(1-\lambda)$$

Robustness monitoring. Let us consider that *F* is an *input in an unstable area for ml*. We will consider that the *ml* component is very likely to produce an erroneous output if *F* occurs. Additionally, we assume that the robustness monitor does not significantly reject robust data.

Hence one may use the following formula for Unsa and Unav:

$$P(Unsa) = \gamma\lambda + \alpha_{ml}(1-\delta)(1-\lambda)$$
$$P(Unav) = \delta(1-\lambda)$$

Concerning abstract interpretation-based monitoring; some numerical values can be extrapolated from the Figure 1a of [31]. Let us consider a robustness radius of 10^{-2}, if we assume that their experiments (made over 100 images) can be generalised then we have:

$$P(F|\neg U_{ml}) = 3.6 \cdot 10^{-2} \qquad P(F|U_{ml}) = 6.3 \cdot 10^{-1}$$

The initial accuracy of the FNN-MNIST is 95.8%, let us consider that any misclassification is unsafe we have:

$$P(U_{ml}) = 4.2 \cdot 10^{-2}$$

One may estimate λ as follows:

$$\begin{aligned} P(F) &= P(F|U_{ml})P(U_{ml}) + P(F|\neg U_{ml})\big(1 - P(U_{ml})\big) \\ &= 6.1 \cdot 10^{-2} \end{aligned}$$

The performance of the ML model without F can be computed as follows:

$$\alpha_{ml} = P(U_{ml}|\neg F) = P(\neg F|U_{ml})\frac{P(U_{ml})}{P(\neg F)} = 1.65 \cdot 10^{-2}$$

Let us consider that we are using the approximate robustness radius computation whose performance are depicted on the Figure 1b of [31]. This monitor will compute an under-

approximation of the exact robustness ball (i.e., it is a pessimistic monitor) so there are only false positives w.r.t. F that is images that are rejected even if their true robustness is above the threshold. So, we have

$$\alpha_{ml} = 1.65 \cdot 10^{-2} \qquad \delta = 1.5 \cdot 10^{-2}$$
$$\lambda = 6.1 \cdot 10^{-2} \qquad \gamma = 0$$

The resulting unavailability and safety measures are:

$$P(Unsa) = 1.53 \cdot 10^{-2} \qquad P(Unav) = 1.4 \cdot 10^{-2}$$

Thus, according to the experiments of [31], adding the online robustness monitoring enhances slightly the integrity ($P(U_{ml})$ / $P(Unsa)$ ratio is approximately 2.7) with a one percent *availability* loss. In this example, the measures quantify the probability of misclassification (and lack of classification) per image. Let us translate these results for the use-cases of the section II.B. For the signal recognition system, the measures are expressed per signal. If we assume that the errors are not correlated to the signal's color, we obtain an error rate of $1.53 \cdot 10^{-2}$ per red signal that is higher than the expected 10^{-2} rate. For the automotive use case, the resulting error rate with a very simple vehicle model is $7.6 \cdot 10^{-1}$ per hour that is not in the same order of magnitude than the expected error rate. Note that these numerical values have been obtained through experiments considering idealized assumptions [31] and there is no conclusive evidence that such detection performances could be met during the operation.

Even if the previous assessment of a robustness monitoring technique is overly simplistic, it provides and illustrates a simple method to assess the contribution of monitoring techniques to the safety-availability trade-off. This illustration shows that monitoring slightly improves the safety of the system but also affects the system's availability in a non-negligible way.

C. Reject option and conformal prediction

We now describe two other mechanisms that allow to detect or temper ML errors, by either filtering out some inputs (classification with a reject option) or by outputting less informative predictions (conformal prediction). From a system architecture viewpoint, the reject option is similar to OOD monitoring (it filters out inputs that can lead to erroneous predictions), though the reject option is meant to be used for typical (in-distribution) yet hard-to-classify inputs.

1) Classification with a reject option

This setting, also known as *selective classification*, is an extension of the multi-class classification setting and has been studied for many decades; e.g., [36], [37], [38].

Given a new input X (e.g., an image), the goal is to predict the associated label Y out of several possible labels. An algorithm with *reject option* can decide to predict or not. More formally, such an algorithm is given by a *selective function* g and a *classification model* f. When $g(X) = 1$ the algorithm predicts the unknown label Y with $f(X)$. When $g(X) = 0$, the algorithm refrains from predicting. There are two competing objectives, which correspond to the reliability-availability trade-off:

- minimize the *selective risk*, i.e., the average number of errors when the ML algorithm predicts:
$$P(Y \neq f(X) | g(X) = 1)$$
- maximize the *coverage*, i.e., the proportion of inputs for which the ML algorithm outputs a prediction:
$$P(g(X) = 1)$$

For instance, in the railway use-case described earlier, the algorithm could predict or not predict the state Y of the light signal on the input image(s) X. A small selective risk corresponds to making few errors among the predicted light signals. A large coverage corresponds to predicting most light signals, leading to a large availability of the ML system.

Intuitively, there is a trade-off between selective risk and coverage: we can reduce the selective risk by rejecting hard-to-classify inputs x, but this also reduces coverage. This trade-off has been studied theoretically for binary labels and sometimes simple models (e.g., [37], [39], [40], [41]), but also empirically for multiple labels and deep learning models. For instance, on classical benchmarks such as CIFAR-10 or Cats vs. Dogs, empirical results from [42], [43] typically show an improvement of up to a factor of 2 in the risk when coverage is reduced to around 90 – 95%, and up to a factor of 20 for around 70% coverage. For other datasets such as CIFAR-100, SVHN, and ImageNet, the empirical results from [42] are of the same order of magnitude, but the trade-off is less favorable: a smaller coverage is needed to achieve the same risk reduction. In any case, even on CIFAR-10 for which the SelectiveNet algorithm [43] is reported to reduce the risk from around 6.7% at full coverage to around 0.3% at around 70% coverage, the value of 0.3% is still orders of magnitude larger than what would be required from a safety perspective if the ML algorithm errors could not be compensated in a drastic way. Therefore, the reject option is likely not a sufficient solution to fill the gap between safety objectives and ML predictive performances, though it should be considered as an interesting component towards ML error reduction.

We should also note that in many safety-related applications some errors may be considered more costly than others (i.e. carry a higher risk). For example, in our railway use-case, if the problem is cast as a binary classification (detect whether the image contains a red signal or not), then the type II error (missing a red signal) has a far higher risk than the type I error (falsely reporting a non-existent red signal). The theoretical formalism presented above can be extended to take into account this distinction between the types of errors by associating a different cost (also known as risk or loss) with each one. The new risk minimization criterion then reads:

$$\text{minimize} \quad P(f(X) \neq red, Y = red | g(X) = 1) + \lambda P(f(X) = red, Y \neq red | g(X) = 1)$$

where λ is the relative cost of making a type I error as a fraction of the cost of making a type II error. The coverage criterion we seek to maximize remains unchanged. Statistical guarantees for this asymmetric cost or risk formulation have been explored in the literature and have been successfully applied to the medical domain [39].

2) Conformal prediction

Conformal prediction is another way to reduce the risk. It consists in post-processing an ML algorithm and predicting a set $C(x)$ of possible labels for each new input x, instead of a single prediction $f(x)$. Now, predictions are made at all times, but the fact that $C(x)$ can contain more than one element is a way to reduce the probability of making an error: $P(Y \notin C(X))$.

In our railway use-case, this means we allow the ML system to make predictions with less information (e.g., this light signal can be this or that), but with the benefit of making fewer errors. This can prove useful as long as labels in the predicted set $C(x)$ do not often correspond to very different decisions.

Several algorithms have been proposed; see, e.g., [44], [45]. Typically, $C(x)$ is defined as the set of labels with highest probability scores at the output of a deep learning model, using

a data-driven threshold, and possible regularization when tail probabilities are poorly calibrated [46].
The aforementioned methods enjoy theoretical guarantees that are typically of the following form: given a risk level $\alpha \in (0,1)$, we can tune the `size' of $C(X)$ so that

$$P\big(Y \notin C(X)\big) \leq \alpha$$

Importantly, this guarantee relies on statistical assumptions (i.i.d. or exchangeable data), the verification of which can be a very difficult task. Furthermore, the probability above is an average error which does not imply a guarantee for each image, but a guarantee for most images (as those seen in the calibration dataset).
The fact that virtually any ML model can be conformalized (as a post-processing step) to yield a probabilistic guarantee as above makes conformal prediction an appealing technique for safety purposes. This field is currently receiving a lot of attention from the ML community, and we hope efforts will be made to incorporate safety-specific considerations.

D. Temporal redundancy

If we focus on the case of systems that may process sequences of inputs (e.g., consecutive video frames), it is natural to consider exploiting temporal redundancy to consolidate decisions or to exclude sporadic errors. This principle is applicable to ML-based systems, where consecutive inputs can be fed to a single ML model. These inputs are usually correlated; therefore independence cannot always be claimed. Nevertheless, a gain can be expected from such approach when consecutive inputs vary (for instance when the train is running).

In order to understand that gain, let us denote by X_t the correct classification at time t. Then the probability of two consecutive failures can be decomposed as follows:

$$P(\overline{X_t} \cap \overline{X_{t+dt}}) = P(\overline{X_t}) \times P(\overline{X_{t+dt}}|\overline{X_t})$$

If $P(\overline{X_{t+dt}}|\overline{X_t}) < 1$, that is to say if the inputs are not fully correlated, then the consideration of two consecutive inputs instead of one single input can increase the reliability.

Considering the ML component, two categories of events can affect single input performance:

- Extrinsic events concerning the environment of the system as perceived by its sensor (here, a camera). A typical example is the masking of the signal by a pole, or a bird, etc. that lead to a reduction of the recognition performance and possibly to an erroneous decision. So, by considering a series of predictions with an appropriate interval, the effects of these events will be filtered out and the capability to take a good decision will be improved.
- Intrinsic events concerning the capability of the system to take a good decision for some input. In that case, the event lasts as long as the input stays in the domain where the ML performance is bad. Temporal redundancy does not improve the capability to take a good decision.

The simplest approach to take advantage of the potential gain on extrinsic events is to use a voting scheme mechanism: the decision resulting of a sequence of predictions is taken as the majority vote of the single predictions.

Alternately, recurrent neural networks (LSTM or GRU [47] [48]) can also be used and show encouraging results for taking decision with time series. The principle is to feed over time a Neural Network with a feedback connexion. The input can either be the raw data (for instance the video sequence), or features extracted from each step in the sequence. The gain in sequence classification accuracy is of several percentage points. For instance, [49] for an action video classification use case obtain a 3% accuracy gain with LSTM compared to a voting scheme.

For safety purpose, we suggest an algorithm based on Finite State Machines (FSMs): each time a new input is received, the output of the ML classifier is used to update the state of a FSM, according to the logic depicted on *Figure 3*. This logic requires N ($N = 4$ for illustration purpose only on the figure) consecutive consistent classifications to make a decision. Otherwise, the output remains undefined.

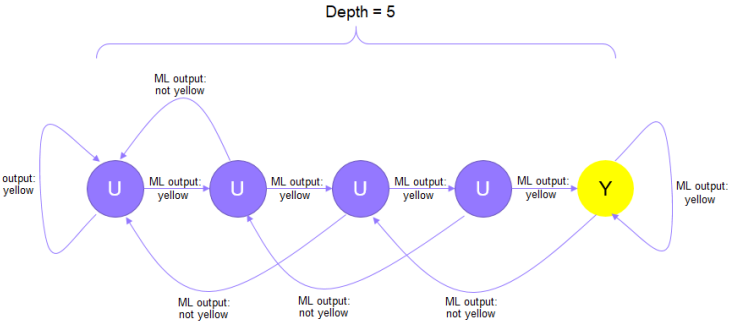

Figure 3: FSM for the classification of a yellow traffic light.

Temporal redundancy suffers from a correlation bias: it is not unusual to face situations where a ML model consistently misclassifies several consecutive inputs that are part of the same image sequence. Indeed, the variation of the input image depends essentially on the variation of the train pose. Unfortunately, if the speed of the train is low, or if the signal is close to the track, the variation of the pose will be low too. Additionally, the mechanism that extracts the relevant part of the image may also filter out variations of the image since it crops and scale it in order to keep the signal in a given bounding box.

In order to estimate the correlation bias and its impact, a simple testing approach is suggested. It consists in comparing the rate of occurrence of the following two events: (1) "N consecutive wrong outputs in the same sequence" and (2) "one wrong output". The former measures the failure rate with temporal redundancy, whereas the latter measures the failure rate without temporal redundancy. A reduced failure rate proves the added value in terms of reliability of this temporal redundancy mechanism.
Additionally, hybridization of various mitigation techniques could also be used. It consists in mixing the algorithm with other methods listed in this paper, in order to compensate for any identified weaknesses.

E. Concluding remarks

As we presented in this section, various methods can be considered (and even combined) to attempt to fill the gap identified between ML performances and safety requirements. Even if most of these methods rely to a certain degree on the independence hypothesis, which is often impossible to guarantee, these solutions still deserve a serious attention as their improvement and their potential combination could become sufficient in the future.
At ML level typically, structural redundancy remains interesting and could help exploiting the partial independence

of ML errors. Moreover, methods for monitoring ML components for instance, such as robustness or OOD monitoring, may be difficult to specify, but they also represent a promising direction to explore. However, for now, the variability of their performances and their poor reliability often negatively impacts the tradeoff between safety and availability. Other ML-based approaches may also be used, such as selective classification, or conformal prediction, which address the reliability-availability trade-off from a statistical viewpoint are able to provide specific statistical guarantees on ML predictions. Finally, when a system processes sequences of inputs, temporal redundancy can also be leveraged either through specific algorithms or ML architectures, to improve the overall results. However, here again, the fact that temporal independence cannot be guaranteed represents a serious impediment.

IV. DISCUSSION: A HISTORICAL PERSPECTIVE

A. Typical stages in the introduction of a new technology

At first sight, the performance gap stated above seems to indicate that ML-based systems are not able to be the core part of a safety-critical system at least for complex tasks [50] [51]. Nevertheless, looking carefully at the history of typical safety-critical systems, we can observe how new technologies can be introduced smoothly in safety-critical systems.

We can distinguish these main stages:

- First, the *bonus*: the new technology is only used to mitigate a risk that is not yet addressed at all, or to help address it beside some existing solution. The risks introduced by the technology itself are considered either negligible or mitigated by other means.
- Second, the *integration*: when the technology is used in the market, there are a lot of opportunities of feedbacks. In fact, there are two sides for this feedback: the overall risk reduction impact, and the adoption by the users. The feedback on the first aspect is given by the facts that some specific new risk could appear only during the operation phase, given that these new risks are (hoped to be) at a low level. This phase is the opportunity to accumulate data, in order to *quantify* the global balance between risk reduction and risk introduction. Meanwhile, the users have time to use the technology and to adopt it, to ask for modification or to reject it. It could be that he misuses it, too, which could generate a new type of risks, which will be added to the overall risk impact seen before.
- Third, the *acceptability change*: when a technology serving safety purposes is more and more used, it is more and more demanded, and can at some point become mandatory. It means that the mitigation of the risk that was an option at the beginning, becomes a requirement, which is asked by some *norms*. Afterwards, it could be that the acceptability level is raised (i.e., the failure rate has to be lowered).

In order to figure out this three-phases safety related systems evolution, let's take the example of the ABS (Anti-blocking Brake System) in the automotive field. In the last seventies, this system was introduced as an option in the premium cars. The market proposition was to reduce the risk of slipping because of wheels blocking due to a too strong braking regarding the road condition (e.g., driving on snow or very wet road). At this moment in the car's history, the road users accept the fact that, in certain conditions, a car could slip when the driver brakes too hard. Then having an equipment that has the ability to avoid this slipping situation was really a safety improvement, a *bonus* to the safety. At the same time, if this new equipment failed by not avoiding the slipping, then it was not less safe than the accepted current situation. Obviously, it was introduced from the beginning by taking care at least as safely as before to the other risks like keeping the ability to brake or to move (i.e., not blocking or releasing the brake unintendedly).

After this phase, it was the phase of *integration*, in the 80's and 90'. The product was more and more used, allowing to know how it really helped the drivers to overcome slipping situations, and what it brought as new effects, that were not expected in the first versions of the new product. For example, the first ABS equipments were not usable on roads with cobblestones, because of the periodic loss of wheel-road contact, which could be unfortunately at the same rhythm that the ABS order to release the brake: this behavior was not expected and was discovered with the usage on the road. More largely, this usage phase allowed to measure the impact of the ABS on the overall road safety, and to quantify all the expected and unexpected benefits and losses, and their balance.

This integration phase resulted in the *acceptability change* phase. This system has clearly been adopted by the users, and the balance of risks is positive. For this reason, for example in Europe in 2004, the ABS equipment was made mandatory by the regulation. The promulgation of this regulation shows that at this time the society considered that the ABS system was adopted by the users and had a positive risk balance. But it shows one more thing: it was not anymore accepted by the society that the accident due to slipping on the road because of a too strong braking application; in other words, the acceptability bar on the danger of slipping has been raised. It means that the risks that were accepted as *normal risk* in the early 80's, were not anymore accepted, and need to be mitigated, in the 2000's.

B. Are such stages applicable to ML?

In the case of ML-based systems, it is obviously too early to predict that this technology will follow this kind of cycle. In particular, the current rapid evolution of these technologies necessarily modifies the way they are integrated and adopted, since no sooner is the technology on the market than it is rendered obsolete by the next version. Furthermore, the nature of the products that can embody ML is much broader than in the case of ABS where the product is confused with the technology.

Having said that, the interesting point is that one can envisage ML-based products that improve overall safety in a domain, even if it is not a safety critical product. The underlying principle is that the new technology will not do worse than the existing one, and therefore if the existing one is acceptable then the new technology will be acceptable. This principle has obvious limitations related to the fact that important changes can also generate fears, and therefore the perceived risk could be different from the real risk. But let's leave that aside and take a few examples of what ML can bring today in a safety critical context.

The anticipation aids that ML can provide are an interesting category. They are based on two principles: it is about helping a human operator (pilot, driver, etc.), and therefore the operator will fully play his role as the main risk mitigation means; it is about helping anticipation, and therefore a potential failure of the system is far in the causal chain from the realization of the risk, which makes it completely legitimate to rely on the operator as a risk mitigation (i.e., he will have time and ability to react). In this case, as in all the others, we remind that we assume that the technology introduced to help mitigate a risk does not increase any other risk simultaneously. Incidentally, this category corresponds to

the category 1A that the EASA has identified in their recently released roadmap [2] as the first to be addressed.

As noted above, how precisely these technologies will be integrated into products, and even more how they will influence their acceptability, is unknown at this time. Typically, it seems unreasonable to think that their integration into products will fundamentally improve their reliability to such an extent that the orders of magnitude are changed, and that the compatibility of this criterion with the requirements of safety critical systems is made possible. But this conjecture is not to be totally rejected either, because an operational life can allow the development of a way to select training data whose representativeness and quantity would allow to reach this necessary reliability.

These considerations lead us to formulate a line of research. It appears that the cycle described above is not formalized at all today. It seems interesting to study a formalization so that these system evolutions can be anticipated, or even programmed, and that this would allow the identification of clear steps for a system to become "safety-critical", or even, why not, to obtain a continuum between non-critical systems and critical systems.

V. CONCLUSION

In this paper we address the complex problem of integrating ML predictive models into safety-critical systems. Through the lens of two practical use-cases we highlight the discrepancy between the performances of current ML models and the acceptable failure rates required by the industrial safety standards.

We observe that initiatives that are trying to propose adjustments of current practices to produce safety-critical software do not address the failure rate question. Therefore, we present several techniques from both domains (ML and safety) and analyze their potential application or extension to address the challenges raised by this assessment.

Throughout our analysis we note that many of these problems can be viewed as a reliability-availability trade-off and each of these techniques can address it from a different perspective.

We further investigate analogies with current practices and norms, as well as historical perspectives on the introduction of pioneering technical innovations.

While we take inspiration from many such precedents, we conclude that none of the enumerated techniques or norms offer satisfactory solutions, at least for now. The introduction of ML components in safety-critical systems remains an open question very much.

Nevertheless, we argue that ML can still contribute to the safety enhancement of current critical systems when implemented as "smart assistant solutions", which address otherwise unmitigated risks while ensuring they do not introduce additional ones (e.g., without any negative impact on the controllability of the system or the human capacity required for safe operation).

In addition to these technical aspects, our aim is to bring together the two communities (ML and safety), in order to build a common and solid foundation for the engineering of future intelligent safety systems. We hope that the discussions and the research directions presented here will motivate other contributors in this challenging endeavor.

VI. APPENDIX: DEFINITIONS OF KEY CONCEPTS

To help the reader unfamiliar with safety terminology, but also to avoid any ambiguities, we provide definitions of key safety concepts below. These definitions tend to be as generic as possible in order to be field-independent and focus on principles more than on normative (implementation) details. In fact, this paper is mainly about transportation systems, trying to be independent of its type (aeronautical, railway, automotive). We also refer the reader to [52] for further definitions.

1. **Failure**: Inability of a system or component to perform required function according to its specification and may have severe consequence on its usage.
2. **Risk**: in this document we only deal with safety risk. A safety risk is the potentiality of a system to provoke some injuries or even death to a person, due to its **failures** or insufficiencies. It is described essentially by its **severity** and **frequency** and can be associated to their mathematical product.
3. **Severity**: the impact level of a **risk**, in terms of number of deaths, number or type of injuries, number or type of other effect leading indirectly to injuries or death. The severity is a discrete (resp. scalar) value, on a finite (resp. bounded) set of values.
4. **Frequency**: the number of occurrences of the **failures** associated to a **risk** in a given time unit. This measured frequency is the **reliability**.
5. **Acceptability**: the fact that a society is keen to authorize the use of a product, because the residual **risk**s are considered sufficiently low (i.e., under the acceptability level).
6. **Risk mitigation means**: the means to decrease a **risk**, by acting on its **severity** or its **frequency**. It can be technical, organizational, or procedural.
7. **Norm/Standard**: a norm or a standard is a reference document (or set of documents) where the **acceptability** level is defined, and the recognized **risk mitigation means** are described. This document is written by a community of people that agree on what they accept as risk and what they do not accept, given the usage of a product. For example, the International community defines through the ISO 26262 what they accept as risk regarding the failures of the electric and electronic equipments for road vehicles (See [11] [12] [13] [14] [15] [16] [17] [18] [19] [53]).
8. **Error**: the occurrence of the state of a part of the system which is not compliant to the specified or intended state. An error can be the unique or the partial cause of a failure (or causes no failure at all).
9. **Exposure**: one aspect of the **frequency** of a failure, to help quantify operational situations for which it can occur. It allows to treat differently the rare and the frequent situations, from a safety point of view. This parameter is more used in the automotive field than in the others; the smaller the exposure, the smaller the frequency.
10. **Controllability**: the ability for a user of the product to avoid the dangerous situation provoked by a **failure**, or at least to decrease its effects. This parameter is more used in the automotive field than in the others; the higher the controllability, the smaller the frequency.

VII. REFERENCES

[1] H. Delseny, C. Gabreau, A. Gauffriau, B. Beaudouin, L. Ponsolle, L. Alecu, H. Bonnin, B. Beltran, D. Duchel, J.-B. Ginestet, A. Hervieu, G. Martinez, S. Pasquet, K. Delmas and Pag, "White Paper Machine Learning in Certified Systems," 2021.

[2] J. M. Cluzeau, X. Henriquel, G. Rebender, G. Soudain, L. van Dijk, A. Gronskiy, D. Haber, C. Perret-Gentil et R. Polak, *Concepts of Design Assurance for Neural Networks,* 2020.

[3] F. T. Laviolette, L. Gabriel, A. Le, N. Amin, S. N. M. Paulina, P. Yann, K. Foutse, A. Giulio and M. Ettore, *How to Certify Machine Learning Based Safety-critical Systems? A Systematic Literature Review,* 2021.

[4] M. Borg, C. Englund, K. Wnuk, B. Duran, C. Levandowski, S. a. T. Y. Gao, H. Kaijser, H. Lönn et J. 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,* n° %11, pp. 1-19, 2019.

[5] R. Salay et K. Czarnecki, *Using Machine Learning Safely in Automotive Software: An Assessment and Adaption of Software Process Requirements in ISO 26262,* 2018.

[6] A. B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins et et al., «Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI,» *Information Fusion,* vol. 58, pp. 82-115, 2020.

[7] C. Molnar, Interpretable Machine Learning, 2019.

[8] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay et D. Mukhopadhyay, «Adversarial Attacks and Defences: A Survey,» *CAAI Transactions on Intelligence Technology,* vol. 6, 2021.

[9] A. Mehta et S. Kumar, «A Survey on Resilient Machine Learning,» 2017. [En ligne]. Available: https://arxiv.org/abs/1707.03184.

[10] D. Carvalho, E. V., M. Pereira et J. Cardoso, «Machine learning interpretability: A survey on methods and metrics,» *Electronics,* vol. 8, n° %18, 2019.

[11] SAE/EUROCAE, *ARP4754A/ED-79A, Certification considerations for highly-integrated or complex aircraft systems,* 2010.

[12] SAE/EUROCAE, *ARP4761/ED-135, Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment,* 1996.

[13] RTCA/EUROCAE, *ED-80/DO-254, Design Assurance Guidance for Airborne Electronic Hardware,* 2000.

[14] RTCA/EUROCAE, *DO-178C/ED-12C, Software considerations in airborne systems and equipment certification,* 2012.

[15] IEC, *61508, Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems,* 2010.

[16] CENELEC, *EN-50127, Railway Applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS),* 2017.

[17] CENELEC, *EN-50128, Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems,* 2020.

[18] CENELEC, *EN-50129, Railway applications - Communication, signalling and processing systems – Safety related electronic systems for signalling,* 2020.

[19] ISO, *26262, Road vehicles -- Functional safety,* 2018.

[20] H. Mania, J. Miller, L. Schmidt, M. Hardt and B. Recht, "Model Similarity Mitigates Test Set Overuse," in *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)* , 2019.

[21] J. Nixon, B. Lakshminarayanan and D. Tran, "Why Are Bootstrapped Deep Ensembles Not Better?," in *ICBINB Workshop at NeurIPS 2020*, 2020.

[22] R. Gontijo-Lopes, Y. Dauphin and E. D. Cubuk, "No One Representation to Rule Them All: Overlapping Features of Training Methods," 2021.

[23] SAE, *J3016, Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems,* 2018.

[24] G. Pang, C. Shen, L. Cao et A. V. D. Hengel, «Deep learning for anomaly detection: A review,» *ACM Computing Surveys (CSUR),* vol. 54, n° %12, pp. 1-38, 2021.

[25] R. Lukas, R. K. Jacob, A. V. Robert, M. Grégoire, S. Wojciech, K. Marius, G. D. Thomas et M. Klaus-Robert, *A unifying review of deep and shallow anomaly detection,* 2021.

[26] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett et M. J. Kochenderfer, *Algorithms for verifying deep neural networks,* 2019.

[27] M. Breunig, H.-P. Kriegel, R. Ng et J. Sander, Lof: identifying density-based local outliers, ACM SIGMOD international conference on Management of Data, 2000.

[28] J. Hansi, W. Haoyu, H. Wenhao, K. Deovrat et C. Arin, Fast incremental svdd learning algorithm with the gaussian kernel, AAI Conference on Artificial Intelligence, 2019.

[29] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen et G. Montavon, A unifying review of deep and shallow anomaly detection, IEEE, 2021.

[30] S. Alireza, S. Mark et J. L. James, A Less Biased Evaluation of Out-of-distribution Sample Detectors, arXiv:1809.04729, 2019.

[31] J. Liu, L. Chen and A. Miné, "Input validation for neuralnetworks via runtime local robustness verification," in *CoRR*, 2020.

[32] C. Müller, F. Serre, G. Singh, M. Püschel et M. Vechev, «Scaling polyhedral neural network verification on gpus,» chez *Proceedings of Machine Learning and Systems*, 2021.

[33] M. Niklas, C. Müller, G. Makarchuk, F. Serre, G. Singh, M. Püschel et M. Vechev, «Prima: Precise and general neural network certification via multi-neuronconvex relaxations,» chez *arXiv preprint arXiv:2103.03638*, 2021.

[34] J. Cohen, E. Rosenfeld et Z. Kolter, «Certified adversarial robustness viarandomized smoothing,» chez *International Conference on Machine Learning*, 2019.

[35] A. Fromherz, K. Leino, M. Fredrikson, B. Parno et C. Pasareanu, «Fast geometric projections for local

robustness certification,» chez *International Conferenceon Learning Representations*, 2020.

[36] C. K. Chow, "An optimum character recognition system using decision functions," *IRE Transactions on Electronic Computers,* pp. 247-254, 1957.

[37] C. K. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on Information Theory,* pp. 41-46, 1970.

[38] M. E. Hellman, "The Nearest Neighbor Classification Rule with a Reject Option," *IEEE Transactions on Systems Science and Cybernetics,* vol. 6, no. 3, pp. 179-185, 1970.

[39] R. Herbei and M. H. Wegkamp, "Classification with Reject Option," *The Canadian Journal of Statistics / La Revue Canadienne de Statistique,* vol. 34, no. 4, pp. 709-721, 2006.

[40] R. El-Yaniv and Y. Wiener, "On the Foundations of Noise-free Selective Classification," *Journal of Machine Learning Research,* vol. 11, no. 53, pp. 1605-1641, 2010.

[41] J. Lei, "Classification with confidence," *Biometrika,* vol. 101, no. 4, pp. 755-769, 2014.

[42] Y. Geifman and R. El-Yaniv, "Selective Classification for Deep Neural Networks," in *Proceedings of NeurIPS 2017*, 2017.

[43] Y. Geifman and R. El-Yaniv, "SelectiveNet: A Deep Neural Network with an Integrated Reject Option," in *Proceedings of ICML 2019*, 2019.

[44] Y. Romano, M. Sesia and E. Candes, "Classification with Valid and Adaptive Coverage," in *Proceedings of NeurIPS 2020*, 2020.

[45] M. Cauchois, S. Gupta and J. C. Duchi, "Knowing what You Know: valid and validated confidence sets in multiclass and multilabel prediction," *Journal of Machine Learning Research,* vol. 22, no. 81, pp. 1-42, 2021.

[46] A. Angelopoulos, S. Bates, J. Malik and M. I. Jordan, "Uncertainty Sets for Image Classifiers using Conformal Prediction," in *Proceedings of ICLR 2021*, 2021.

[47] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation,* vol. 9, no. 8, p. 1735 1780, 1997.

[48] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk et Y. Bengio, «Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,» 2014.

[49] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia et A. Baskurt, «Sequential Deep Learning for Human Action Recognition,» chez *2nd International Workshop on Human Behavior Understanding (HBU)*, Amsterdam, Netherlands, 2011.

[50] G. Katz, G. W. Barrett, D. L. Dill, K. Julian et M. J. Kochenderfer, «Reluplex: An efficient SMT solver for verifying deep neural networks,» chez *CoRR*, 2017.

[51] M. Damour, F. De Grancey, C. Gabreau, A. Gauffriau, J.-B. Ginestet, A. Hervieu, T. Huraux, C. Pagetti, L. Ponsolle et A. Claviere, «Towards Certification of a Reduced Footprint ACAS-Xu System: A Hybrid ML-Based Solution,» chez *International Conference on Computer Safety, Reliability, and Security (SAFECOMP)*, 2021.

[52] A. Avizienis, J. Laprie and B. Randell, "Fundamental Concepts of Dependability," 2000. [Online]. Available: https://www.researchgate.net/publication/2408079_Fundamental_Concepts_of_Dependability.

[53] ISO, *PAS 21448, Road vehicles - Safety of the intended functionality (SOTIF),* 2019.