

Certification of embedded systems based on Machine Learning: A survey

Guillaume Vidot, Christophe Gabreau, Ileana Ober, Iulian Ober

▶ **To cite this version:**

Guillaume Vidot, Christophe Gabreau, Ileana Ober, Iulian Ober. Certification of embedded systems based on Machine Learning: A survey. 2021. hal-03252906v2

HAL Id: hal-03252906
https://hal.science/hal-03252906v2

Preprint submitted on 29 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Certification of embedded systems based on Machine Learning: A survey

GUILLAUME VIDOT, Airbus Opération S.A.S, France and University of Toulouse, IRIT, France
CHRISTOPHE GABREAU, Airbus Opération S.A.S, France
ILEANA OBER, University of Toulouse, IRIT, France
IULIAN OBER, University of Toulouse, IRIT, France

Advances in machine learning (ML) open the way to innovating functions in the avionic domain, such as navigation/surveillance assistance (e.g. vision-based navigation, obstacle sensing, virtual sensing), speech-to-text applications, autonomous flight, predictive maintenance or cockpit assistance. Current certification standards and practices, which were defined and refined decades over decades with classical programming in mind, do not however support this new development paradigm. This article provides an overview of the main challenges raised by the use ML in the demonstration of compliance with regulation requirements, and a survey of literature relevant to these challenges, with particular focus on the issues of robustness and explainability of ML results.

1 INTRODUCTION

Safety critical systems, and avionics in particular, represent an application field unenthusiastic in applying new software development methods [33], as shown by the fact that some aspects of the object-oriented programming, such as polymorphism and dynamic biding, never made their way to safety critical systems, mostly because the inconvenient balance between added value and increased development and validation costs. Nevertheless, the recent advances in machine learning triggered genuine interest, as machine learning offer promising preliminary results and open the way to a wide range of new functions for avionics systems, for instance in the area of autonomous flying. In this paper we investigate on how existing certification and regulation techniques, can (or cannot) handle software development that includes parts obtained by machine learning.

Nowadays a large aircraft cockpit offers many avionic complex functions: flight controls, navigation, surveillance, communications, displays... Their design has required a top down iterative approach from aircraft level downward, thus the functions are performed by systems of systems, with each system decomposed into subsystems that may contain a collection of software and hardware items. Therefore, any avionic development considers 3 levels of engineering: *(i)* Function, *(ii)* System/Subsystem and *(iii)* Item. The development process of each engineering level relies on several decades of experience and good practices that keep on being adapted today. These methods have been standardized through EUROCAE/SAE standards for system development (ED-79A/ARP4754A) and EUROCAE/RTCA standards for software items (ED-12C/DO-178C and supplements) and hardware items (ED-80/DO-254). They are recognized as applicable means of compliance with regulation requirements by worldwide Avionic Authorities (AAs). In this context, this survey actually reduces the domain of study to the item level and more precisely to the software item. Note that at the item level, the correct terminology for the demonstration of conformity to standards is “qualification”.

The item’s development workflow is usually represented by the V-cycle (refer to Figure 1). Today, the usual development paradigms are Requirement-based Engineering (RBE) or Model-Based Engineering (MBE). Indeed, avionic items are developed using *classical programming*, i.e. the

Authors’ addresses: Guillaume Vidot, eric-guillaume.vidot@airbus.com, eric.vidot@irit.fr, Airbus Opération S.A.S, Toulouse, France and University of Toulouse, IRIT, Toulouse, France; Christophe Gabreau, Airbus Opération S.A.S, Toulouse, France, christophe.gabreau@airbus.com; Ileana Ober, University of Toulouse, IRIT, Toulouse, France, ileana.ober@irit.fr; Iulian Ober, University of Toulouse, IRIT, Toulouse, France, iulian.ober@irit.fr.

developers explicitly use language instructions to implement requirements (or model) and thus are able to control each line of code of the embedded software item.

Recently, new avionic functions have emerged, aiming at developing new flight experiences: navigation/surveillance assistance (e.g. vision-based navigation, obstacle sensing, virtual sensing), speech-to-text applications, autonomous flight, predictive maintenance, cockpit assistance...

Contrary to *classical programming* which can hardly support these functions, Machine Learning (ML) which is a sub-domain of Artificial Intelligence (AI), is well known to show good results for most of them (e.g. [5, 14, 49]). Current industrial guidance have a strong focus on bespoke technologies in aeronautical applications thus, are not appropriate to support this paradigm change.

Indeed, ML techniques introduce a brand new paradigm in avionic development: the data-driven design of models (including supervised, unsupervised and reinforcement learning). *Data-driven* refers to the fact that the data rule the algorithm behavior through a learning phase. We choose to restrict our research to **offline supervised learning**, i.e. the training is done with *labelled data* and *before embedding the ML software*. As a first step to the qualification of a ML item, these restrictions seems reasonable. Besides, these restrictions do not affect our capability to find efficient algorithms that might solve the emerging functions mentioned above.

Even with this restriction, many significant issues remains regarding the current demonstration of conformity. Indeed, some fundamentals of the usual techniques (Requirement Based Engineering or Model Based Engineering) are jeopardized, challenging the classical safety guarantee argumentation:

- Specifiability: It sounds difficult, a priori, to capture the complete behavior of a ML model (for instance when the training dataset is composed of millions of images), decreasing the confidence that the model behaviour will always match the functional intent and be free of unintended behavior that may jeopardize the safety.
- Traceability: The relationship between item requirements and corresponding *learned parameters* of the ML algorithm cannot be established. This makes the ML item design less transparent and draw a need for explainability capabilities to add confidence that the algorithm correctly and safely implements the intended function.

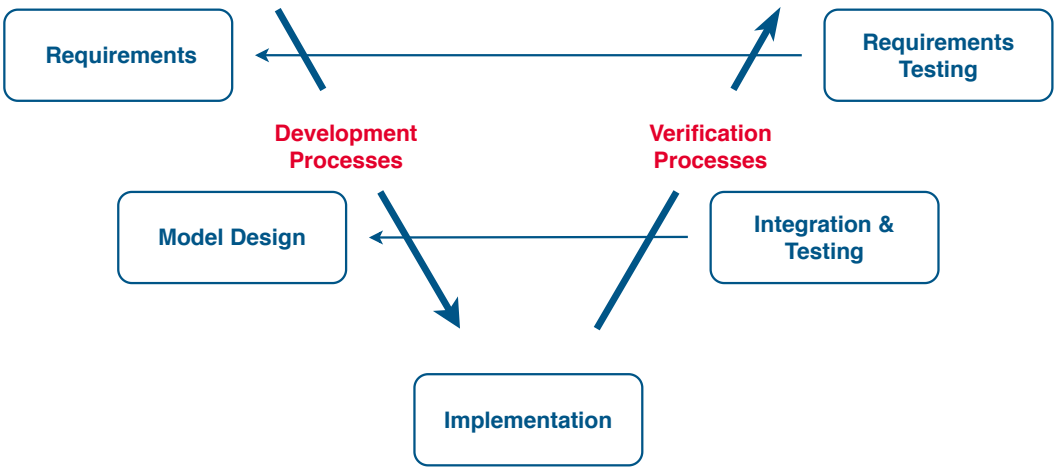


Fig. 1. Item development workflow

- Robustness: Perturbations in the design phase and the operational inference of the trained algorithm may lower the reliability on the predictions correctness and may degrade the functional and safety performances.

The motivation of this survey is two-fold: *(i)* identify the main challenges to the certification of ML based item (see Figure 2), *(ii)* overview the literature to know whether existing methods or techniques are able to solve the problems raised by the identified challenges. Compared with the survey of Huang et al. [26], which is also motivated by the certification of critical embedded systems that include Deep Neural Network (DNN), our survey is guided by aeronautics regulations and standards and is not limited to DNN, although many of the cited works deal with DNN.

Currently, the challenges for certifying ML algorithms are listed in Figure 2. The goal of this survey is not to provide a complete overview of possible techniques to solve these challenges, its scope is reduced to the trustworthiness and methodology considerations. Specifically, the survey focuses on Explanability and Robustness (more precisely adversarial robustness) because they address key issues introduced by the intrinsic characteristics of ML algorithms: the lack of transparency of the design (black-box aspect) and the lack of performances reliability when inputs are slightly perturbed in the range of the Operational Design Domain (ODD). In grey, in Figure 2, are the topics that are out of our scope: *Embeddability* and *Resiliency*. *Embeddability* raises classical implementation concerns such as the deterministic behavior of the item versus memory and computational constraints. Some resolution techniques are specific to ML, such as the optimization of the ML algorithm structure or implementation architecture to fit the targeted host platform. *Resiliency* monitors the behavior of the system when it is deployed, e.g. error detection or fault tolerance. This is an issue to be addressed at the system level of engineering.

This paper is structured as follows: in section 2, we overview the main challenges raised by the use ML in the demonstration of compliance with regulation requirements. In section 3 we introduce the Trustworthiness considerations which will possibly help to fill the existing gaps in future certification approaches (see Figure 3). Then the sections 4 and 5 focus on robustness and explainability, respectively. Eventually, we end the paper with section 6 which recaps the identified problems for the qualification of ML items.

2 IMPACT OF ML TECHNIQUES ON CERTIFICATION APPROACH

The European Union Aviation Safety Agency (EASA) provides regulatory material that defines and explains all the requirements due for developing safe avionic products [15]. Regarding software/hardware items embedded in avionic safety-related systems, the certification specification are described in CS 2x.1301 and CS 2x.1309. Roughly speaking, we would summarize these paragraphs as follows: *avionic systems should safely perform their intended function under all foreseeable operating and environmental conditions.* Through supplemental documents (AMC20-115D and AMC20-152A[1]), EASA recognizes the current avionic standards as acceptable means of compliance to the regulation text. The grey element in Figure 3 shows the relation between regulations and standards.

Assuming the ML techniques are reduced to offline learning, we do not anticipate any change to the regulation [15]. However, even if regulation requirements are unchanged, the current standards do not provide sufficient guidance to make a complete demonstration of conformity for a ML item. This section details these gaps.

Specifiability. One of the fundamentals of the RBE (or MBE) relies on the correct and complete capture of the item requirements: either they come from system allocated requirements (intent) or from its own behavior (emerging functions). In this context, the item can be verified to safely

[1] AMC stands for Acceptable Mean of Compliance

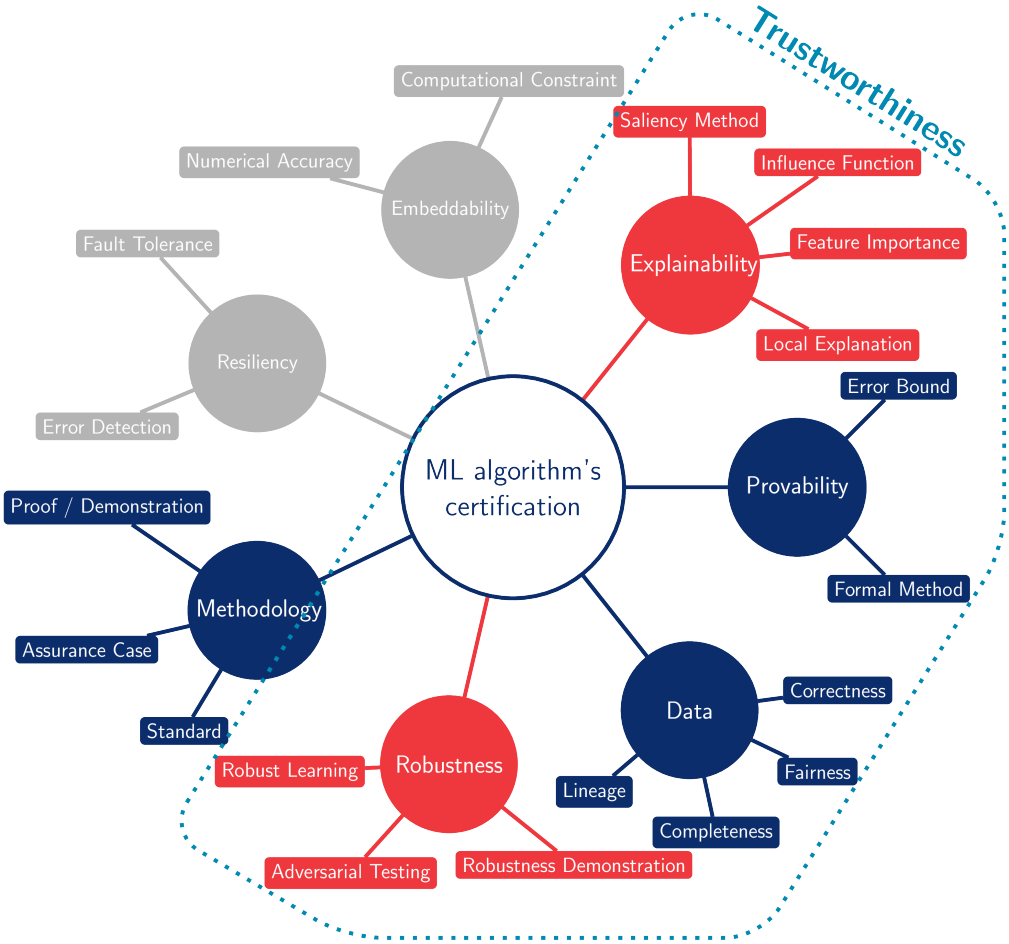


Fig. 2. AI certification key elements. In red, are the domain that we detail in this survey. In blue are the domain that we briefly present. In grey are the domain out of our scope.

perform the intended function under all foreseeable operating and environmental conditions. With ML items, it sounds difficult to fully specify the function with a classical requirement process. For instance, all the possible way to describe a runway whatever the operational (e.g. sensors own bias) and the environment conditions (e.g. weather, light conditions) cannot be defined using textual or

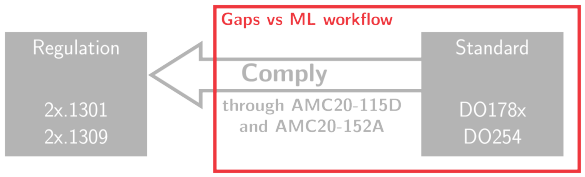


Fig. 3. Certification Process Overview

even modelling technique. This is the reason why we will prefer use millions of images instead, betting that the data experience will complete the requirement capture and that ML technique will enable the development of an acceptable runway detection function. Therefore the difficulty will be to describe as thoroughly as necessary the operational domain (ODD) in which the detection function is supposed to operate and then to validate that the collected data correctly and sufficiently represent this ODD.

Traceability. The current software standard requires the traceability between the requirements and the code (in both senses) so that each line of embedded code can be justified as implementing captured requirements. In a ML development context, this relationship that makes the design a white-box process, is lost. Actually the trained algorithm is a complex mathematical expression (made of arithmetic operations with weights and bias) that is not traceable to any upward functional requirements. Thus it becomes infeasible to demonstrate the completeness of implementation using traceability. From this lack of traceability comes the loss of transparency of the model, making the link from the input data to the output predictions not understandable by a human. This lowers the confidence that the safety properties of the intended function are preserved in its operational domain. In this context, explainable AI could be a means to enhance the confidence in these algorithms when safety is highly critical (see section 3 and 5).

Unintended behaviour. All the life-cycle processes (requirements capture, model design, implementation, integration and requirement-based testing) of the item development workflow (see Figure 1) are used to demonstrate the consistency with the intended function. In addition to the problems mentioned above, the learning phase could introduce some unexpected behavior that we cannot measure or prevent using usual activities. It means that we cannot guarantee the absence of unintended behavior during item operation and specifically those affecting aircraft safety. This may be due to several root causes such as a low-quality data management process (e.g. bias, mislabeled data) or inadequate learning process. We will see in the rest of the paper that there are methods to limit or to formally verify the occurrence of such effects.

3 TRUSTWORTHINESS CONSIDERATIONS

The qualification of ML items in the frame of avionic developments offers lots of research challenges in various domains. We selected the overview of 5 different domains that seem very promising and which will contribute to support the certification of ML-based systems. Thus, this section overviews the domains covering the Trustworthiness (see Figure 2) and the Methodology aspects.

Explainability and robustness of Machine Learning model have already been reviewed, the reader may therefore refer to [7, 26, 50], to get more details on these topics. Nevertheless, the originality of our work stands in the perspective of the conformity to the regulation requirements inherent to the development of safety-critical systems for avionic domain.

Explainability. Explainability is a topic inherent to ML technique. It is not needed for classical programming since the item development can be fully explained through the joint requirement and traceability processes which enable the interpretation of each line of embedded code. ML development breaks this understanding chain and make room for an undesirable black box effect. This is the reason why one can think that, when required by the safety level of the implemented function, explainability would be requested to add the necessary confidence to support the demonstration of conformity [16]. Phillips et al. [47] introduce four principles of explainable AI: explanation, meaningfulness, explanation accuracy, knowledge limits. Basically, these principles state that an explainable AI must: *(i)* provide an "evidence or reason for all outputs" *(ii)* be meaningful with respect to the audience, *(iii)* be representative of the way that the algorithm produce the output

and *(iv)* be aware of the domain of usage of the algorithm, i.e. it should not give an explanation when the input is out of scope since the algorithm is not design to work with it.

In the aeronautical context, four stakeholders could need explanation: the designer, the authorities, the pilot and the investigator. Considering the "meaningfulness" principle, each of them could receive different explanation. Indeed, the needs for explanation would be different for an engineer who knows the system and for an end-user who has no prior knowledge of the system. The explanations could also differ whether it is used for debugging purposes during the development (for designer), for investigation purposes in case of in-flight issues (for authorities or investigator) or for a user assessment of the model predictions when the system is deployed, e.g. a pilot who requests justification of the decision to enhance his confidence in the system before acting. In the literature, we find two kinds of explanation: local explanation [13, 17, 23, 24, 30, 51, 52, 65] and global explanation [38, 56, 67] (see Section 5).

Adversarial Robustness. Robustness comprises several sub-domains (adversarial examples, distributional shift, unknown classes, *physical* attacks, ...) but we focus only on *adversarial robustness* which is defined as the capability of algorithms to give the same outputs considering some variation of the inputs in a region of the state space. The issues addressed by the adversarial robustness domain are at the heart of the certification process; it addresses partially the demonstration of the intended function. The aim is to assess and/or enhance the behavior of the algorithm when dealing with *abnormal inputs* (noise, corner case, sensor malfunction...). The literature is two sided: one side works at enhancing the adversarial robustness of the algorithm [9, 20, 21, 35, 39, 44, 45, 59, 66], the other side at its verification [12, 19, 27, 28, 31, 42, 54, 57, 60]. Improving the adversarial robustness consists in finding robust learning procedure that is resilient against crafted exampled made to defeat the ML algorithm. The verification of this latter property is either based on formal methods [12, 28, 31, 54, 60] or optimization methods [19, 27, 42, 57].

Provability. The provability aspect is the capability to formally demonstrate that system properties are preserved. Formal methods provides mathematical evidences to support such demonstration. As stated above, formal methods are already used to verify well-defined properties of algorithms [28, 29, 62], such as the robustness [19, 27, 42, 57].

The error (or generalization) bound gives guarantees on the ability of the algorithm to generalize.. An algorithm generalizes well when it maintains its high performance on unseen data, i.e. the theoretical error, r is close to the empirical error $\hat{r}_S$ (computed from a training set S). The idea is to bound the gap between the theoretical error r and its empirical counterpart $\hat{r}_S$. However, r is not computable since it is the error on all possible data. Hence, the need for probabilistic bounds which give an upper bound on the gap between r and $\hat{r}_S$. The first generalization bounds appears with the PAC (Probably Approximately Correct) theory [61] and have the following forms:

$$\Pr_S\Big(|r - \hat{r}_S| \leq \epsilon(\text{model's complexity}, \#S, \delta)\Big) \geq 1 - \delta, \tag{1}$$

where $\epsilon(\cdot) \geq 0$ and $\delta \in]0, 1]$. ϵ is a function that models the upper bound that usually relies on the complexity of the model, the number of samples in S (denoted by $\#S$), and the probability δ. The ideal scenario is to have the gap between r and $\hat{r}_S$ small while having a high probability that the inequality holds, i.e. having $\epsilon(\cdot)$ and δ as small as possible.

Particularly, the PAC-Bayesian theory[41, 55], which interprets an algorithm as a *majority vote*, is well known to provide tight generalization bound [46]. A majority vote is defined as the weighted sum of several models where the weights of the sum constitute a distribution Q. Hence, a model follows a prior distribution P before the learning and posterior distribution Q after. A PAC-Bayesian

generalization bound could have the following form:

$$\Pr_S\Big(r \leq \hat{r}_S + \epsilon(KL(Q\|P), \#S, \delta)\Big) \geq 1 - \delta. \tag{2}$$

One can notice that both bounds have a similar structure but the particularity of the PAC-Bayesian theory is to use the Kullback-Leibler divergence between Q and P to quantify the complexity of the model. Besides, the differences stand also in the definition of r and $\hat{r}_S$ since in the PAC-Bayesian theory the algorithm is interpreted as a *majority vote*.

Data management. In the avionic context, data has been used for a long time with the use of databases or configuration files. However Machine Learning techniques are bringing a totally new aspect in the qualification approach. Contrary to *classical programming*, ML design techniques are data-driven, thus the data management process is essential to the demonstration of conformity. As already stated, building a ML algorithm of good quality, requires data of good quality (no erroneous data, no mislabelled data, ...) but this is not sufficient. Indeed the data representativeness plays also a significant role. Representativeness comes from statistics and is a quite challenging problem: it allows to check if the data is a correct snapshot of the phenomenon to be learnt.

The quality of the data can be measured using existing metrics: accuracy, consistency, relevance, timeliness, traceability, and fairness are some of them [8, 48]. **Accuracy** checks if the data is well measured and stored. **Consistency** verifies if the preprocessing of the data does not compromise their integrity. **Relevance** measures if there are sufficient data to learn the intended function and if the data contain the correct information regarding the needs. **Timeliness** concerns the availability of the data over time. **Traceability** verifies the reliability of the data source and if all activities for the transmission do not alter the data integrity. **Fairness** is about avoiding undesirable bias in the dataset.

Methodology. The *methodology* structures the assurance activities that are needed to support the qualification aspects, i.e. the demonstration to the Authorities that the item development is compliant with the standardized guidance. The red square in Figure 3 highlights that changes are required in the AMCs and/or the industrial standards. Therefore it will be necessary to find alternative means to comply with the regulation material, either by adapting the current certification approach or by building a new one.

Then it will be crucial to clearly define the development process of ML items ([3, 4, 6, 58]) to *(i)* ease their development and maintainability, and *(ii)* identify the validation and verification activities. Figure 4 describes this development process: requirements from the system/subsystem level are refined into ML item requirements to fit to the three main stages of the workflow: Data Management process, Design process and Implementation process.

Data Management Process First data are collected with respect to the problem to be solved. Then, the data should be cleaned and labelled[2], i.e. inaccurate data samples are removed and each remaining data sample is assigned with a true label (known as "ground truth"). Finally, data are preprocessed and split. Preprocessing encompasses all the tasks that transform the data into a more suitable format for the design phases (e.g. feature engineering or data normalization). Preprocessing effort may vary depending on the data collection phase (e.g. data comes from different sources) and the type of data (e.g. images, time series, tabular, sound, text). After the preprocessing step, the data is split into a training, a validation and a testing dataset.

Design Process The design process takes as input the three datasets and outputs a frozen model. The processes in this stage are more iterative than sequential. To tune the hyper-parameters of a model, you iterate between the training and the validation phase. After the tuning part, you

[2]as the scope is reduced to offline supervised learning, labelling the data is necessary.

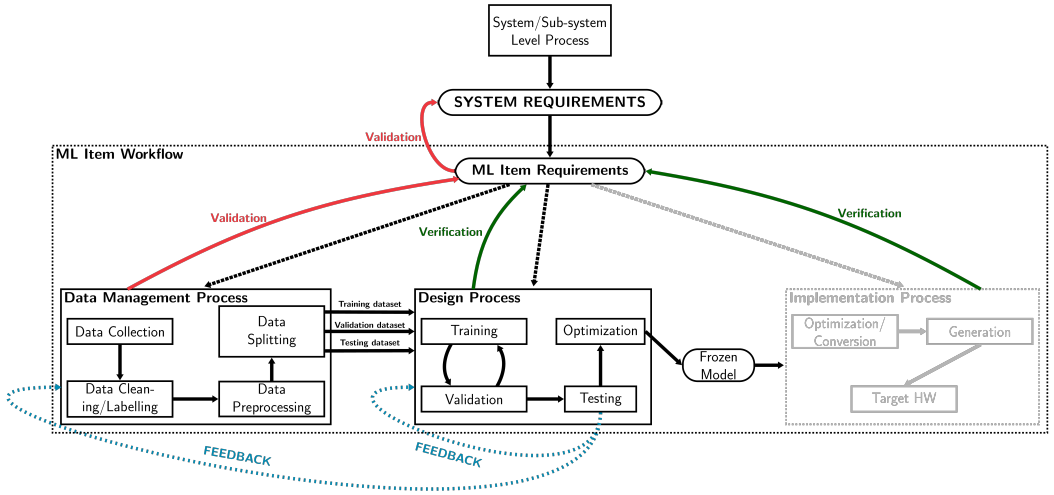


Fig. 4. Machine Learning development process: The three main stages of ML development process (Data Management Process, Design Process and Implementation Process). The stage in grey is out of the survey scope. The red arrows show the validation activities. The green arrows show the verification activities. The blue arrows show where we can loop back if the model underperforms.

are able to test your model. If the performances are lower than expected with respect to the ML item requirements, it is possible to loop back either to the data management process or the design process.

Implementation Process Within the implementation stage, the specificity of the hardware is taken into consideration: the frozen model is optimized and converted with respect to the host platform constraints and the operational requirements cascaded from the system level. Finally, a binary code is generated and loaded in the hardware target. Since we only consider the item design aspects, the implementation stage is out of the scope of this survey.

Since its creation in the 80's, the ED-12/DO-178 standard issues gather the *industrial best practices* that impose the necessary rigor of development to avoid the introduction of errors that may lead to a system failure, and thus increase the safety of the system. Historically, methodologies undeniably brought efficient results in terms of safety. Thus methodological considerations are key to build a correct certification approach. One lead to build relevant safety argumentation is the concept of safety case [53] or assurance case [40]. Indeed, Rushby [53] argues that the introduction of this kind of methodology (i.e. not prescriptive approaches) in the industries helps to significantly reduce the number of accidents and deaths. Assurance cases, which are a generalization of safety cases, are defined by MITRE [40] as follows: *a documented body of evidence that provides a convincing and valid argument that a specified set of critical claims regarding a system's properties are adequately justified for a given application in a given environment.* Hence, assurance cases could be used during the development process of a ML-based item to build, share and discuss sets of structured arguments to support the demonstration of conformity based on outcomes of specific assurance techniques.

4 ADVERSARIAL ROBUSTNESS

Concerning the fundamentals of the regulation requirements, i.e. mainly to demonstrate that the system safely performs the intended function, the key role of robustness is twofold: on one hand

[3]It is important to note that the techniques can be adapted with different ℓ_p-norm

Table 1. Summary of robustness method: The top of the table shows the papers that provide an attack and/or a defense and the bottom shows the papers that provide a verification method. The ℓ_p-norm column refers the norm originally considered in the papers to compute the distance between x and x_{adv}[3]. "-" means that the method do not rely on an ℓ_p-norm. The color code indicate whether the norm is used for attack or defense; black means it is used for both. The colunm "# parameters" reports the size of the Neural Network (in terms of numbers of parameters) used in the papers; it give an insight on the scalability of the associated method.

Attack/Defense				
Method Name	Papers	Attack	Defense	ℓ_p-norm
C&W	Carlini and Wagner [10]	✓		$\ell_0/\ell_2/\ell_\infty$
PARSEVAL TRAINING	Cissé et al. [12]		✓	ℓ_2
FGSM	Goodfellow et al. [21]	✓	✓	ℓ_∞
IFGSM	Kurakin et al. [35]	✓	✓	ℓ_∞
GRADIENT-BASED ATTACK+ILA ([25])	Li et al. [37]	✓*		ℓ_∞
PGD	Madry et al. [39]	✓	✓	ℓ_∞/ℓ_2
DEEPFOOL	Moosavi-Dezfooli et al. [43]	✓		ℓ_2
JSMA	Papernot et al. [45]	✓		ℓ_0
DKNN	Papernot and McDaniel [44]	✓*	✓	ℓ_∞/ℓ_2
L-BFGS	Szegedy et al. [59]	✓	✓	ℓ_2
LMT	Tsuzuku et al. [60]		✓	ℓ_2
GDA	Zantedeschi et al. [64]		✓	-
YOPO	Zhang et al. [66]		✓	ℓ_∞
Verification				
Method Name	Papers	Sound	Complete	# paramaters
AI^2	Gehr et al. [19]	✓		> 238,000
DLV	Huang et al. [27]	✓	✓	> 138,000,000
RELUPLEX	Katz et al. [28]	✓	✓	~ 13,000
MARABOU	Katz et al. [29]	✓	✓	~ 13,000
REFINEZONO	Singh et al. [57]	✓		> 1,000,000
RELUVAL	Wang et al. [62]	✓		

* The authors adapt existing attacks to target the weakness of their defense.

and according to ED-12C/DO-178C, it is *the extent to which software can continue to operate correctly despite abnormal inputs and conditions*. On the other hand and more specifically to ML application, EASA [16] states that ML system is robust when it *produces the same outputs for an input varying in a region of the state space*. Perturbations can be natural (e.g. sensor noise, bias...), variations due to failures (e.g. invalid data from degraded sensors) or maliciously inserted (e.g. pixels modified in images) to fool the model predictions. When perturbed examples fool the ML algorithm we talk about *adversarial examples*. It is commonly defined as noises on inputs that are imperceptible or that do not exceed a threshold. We state in Definition 4.1 two possible manners of generating adversarial examples, *i.e.* attacking an ML model.

Definition 4.1. Adversarial Example. Let (x, y) be a benign example and the true label, δ be a perturbation, ϵ be the maximum allowed perturbation, L be a loss function and f be the trained ML model, we have

$$\begin{aligned} &\min \|\delta\|_p \\ \text{such that}\quad &f(x+\delta) \neq y, \\ &x+\delta \in \mathit{Domain}(x) \end{aligned} \tag{3}$$

or

$$\begin{aligned} &\max L(f(x+\delta), y) \\ \text{such that}\quad &\|\delta\|_p \leq \epsilon, \\ &x+\delta \in \mathrm{Domain}(x) \end{aligned} \tag{4}$$

where $\|\cdot\|_p$ is an ℓ_p-norm and Domain(x) depict the allowed values for the example x.

As stated in Definition 4.1 the optimization problem leads to get an adversarial example with a different label than the original one; we refer to it as "untargeted attack". Few modifications of the optimization problem allow to have a "targeted attack", i.e. the possibility to choose the label of the adversarial examples. Let t be the targeted label we have,

$$\begin{aligned} &\min_{\delta} \|\delta\|_p \\ \text{such that}\quad &f(x+\delta) = t, \\ &x+\delta \in \mathit{Domain}(x) \end{aligned}$$

or

$$\begin{aligned} &\min_{\delta} L(f(x+\delta), t) \\ \text{such that}\quad &\|\delta\|_p \leq \epsilon, \\ &x+\delta \in \mathrm{Domain}(x) \end{aligned}$$

.

Moreover, the attacks could be done in either a white-box or black-box setting. White-box setting basically means that the *attacker* has full access to the model, its parameters and the training dataset while black-box setting means that the *attacker* can only query the model with data.

Finally, the means deployed to overcome adversarial attacks is known as adversarial defenses. One of the most efficient defense is the Adversarial Training (AT) [21, 35, 39]. It consists in augmenting your training dataset with adversarial examples. However, it is often observed that the adversarial training based on a particular ℓ_p-norm are less effective against attacks based on different ℓ_p-norm.

Nevertheless, the defenses only enhance the adversarial robustness of ML algorithms whereas for the demonstration of conformity, we will need proof that systems based on the ML algorithm behave safely and as expected. Actually, there exists methods in the literature that verify that an ML algorithm is adversarially robust. It could bring the guarantee needed to demonstrate the conformity to the regulation requirement. Thus, to handle the adversarial robustness issue, we review adversarial attacks/defenses (cf. Section 4.1) and verification methods (cf. Section 4.2). We report in table 1 adversarial attacks/defenses and verification methods. It is important to highlight that, for the attack/defense methods, we report the ℓ_p-norm used in the paper. However the methods could be adapted to other norms.

4.1 Adversarial attacks/defenses

Szegedy et al. [59] was one of the first to point out adversarial examples as a weakness of ML algorithm. The highlighting of this weakness gave rise to numerous researches around adversarial attacks and defenses [20–22, 31, 34, 35, 39, 44, 45, 59, 66]. Carlini et al. [9] provide advice and good practices about the evaluation of the adversarial robustness. Especially, they claim that one that develops a new defense mechanism must think about an "adaptive attack" to evaluate the efficiency of their defense mechanism. In other words, to evaluate the defense efficiency of your new defense mechanism, you have to test the worst attack against it.

4.1.1 Attack. Goodfellow et al. [21] develop an efficient method to find adversarial example called Fast Gradient Sign Method (FGSM). The attack consists in crafting the adversarial example x_{adv}, by adding a fraction, ϵ, of the loss gradient's sign with respect to the input to the original example x:

$$x_{adv} = x + \epsilon \cdot \mathrm{sign}(\nabla_x L(f(x), y)).$$

Besides, the authors show that adversarial examples are invariant to the learning and the architecture, i.e. different architecture trained on different subsets of the dataset misclassify the same adversarial example. Later, Kurakin et al. [35] propose an iterative version of FGSM which we denote

IFGSM where at each iteration, a perturbation is added to x by applying FGSM to finally obtain x_{adv} after the desired number of iterations:

$$x^0_{adv} = x \qquad x^i_{adv} = x^{i-1}_{adv} + \epsilon \cdot \text{sign}(\nabla_x L(f(x^{i-1}_{adv}), y)).$$

Since perturbations are added several times to x in IFGSM, the ϵ chosen is therefore smaller than in FGSM. The attack introduced by Madry et al. [39] based on Projected Gradient Descent (PGD) is similar to IFGSM except that PGD randomly initialize x before the optimization:

$$x^0_{adv} = x + noise \qquad x^i_{adv} = x^{i-1}_{adv} + \epsilon \cdot \text{sign}(\nabla_x L(f(x^{i-1}_{adv}), y)).$$

Carlini and Wagner [10] also develop a gradient based method but with a slightly different formulation of the optimization problem (cf. Definition 4.1). It is similar to Equation 3 but at the same time they minimize the margin of the model. Besides, the authors provide the formulation of their method for three distance metrics (ℓ_0, ℓ_2 and ℓ_∞). Moosavi-Dezfooli et al. [43] tailor their attack framework, called DEEPFOOL, for finding the minimum perturbation necessary with respect to the ℓ_2-norm[4] to fool the algorithm. They first simplify the optimization problem to a linear classifier, derive the optimal solution for it and finally adapt the optimal solution found for the linear classifier to neural network. They provide the algorithm of DEEPFOOL which describe their iterative approach to estimate the smallest perturbation to create an adversarial example (cf. Algorithm 2 of [43]). Their results show that the adversarial example crafted by DEEPFOOL is often closer to the original example than the ones crafted by FGSM and L-BFGS [59].

Papernot et al. [45] leverage the "forward derivative" of a network f, to design the Jacobian-based Saliency Map Attack (JSMA). To obtain the forward derivative, they compute, from the input layer to the output layer, the derivative of the network f, instead of the derivative of its loss function, with respect to the input x. It basically corresponds to the Jacobian of the function learned by the network f. Then, the authors derive a "adversarial saliency map" based on the Jacobian of the network which points out the input feature that should be modified in order to get a significant impact on the network's output. We refer the reader to their paper [45] to get more detail on the algorithm they provide to craft an adversarial example using forward derivative and adversarial saliency map.

Kurakin et al. [34] assess the adversarial robustness of models deployed in "real-world conditions", *i.e.* the only way to communicate with the systems is through its sensors. Indeed this is a legitimate question since an attacker will not necessarily have access to the ML model. In their experiments, the authors feed an image classification algorithm through a camera. However, they still use the model to generate adversarial examples. They demonstrate that models are still vulnerable against adversarial attack even in "real-world conditions".

In the avionic context, security mesure will be taken to restrict the access to the models and the datasets to undermine the white-box and black-box attacks. Nevertheless the threat still exists since Li et al. [37] propose an attack in a more restricted setting than black-box known as "no-box" setting [11] where the attacker only have very few number of examples that are not training examples. They succeed to efficiently fool the models trained on the imagenet dataset by developing an auto-encoder called "prototypical reconstruction" that manages to learn well with very few data. An auto-encoder consists of two parts: an encoder and a decoder. The encoder, encodes the input by learning a new representation of the original input and the decoder strives to reconstruct the input from the encoding as close as possible to the original input. The authors introduce a new loss for their auto-encoder that is well suited for gradient based attack. Then, their attack consists in using ILA [25], which improves the transferability of the attack, in addition to the gradient-based method.

[4]Note that the authors explain how to adapt their method to other ℓ_p-norm

4.1.2 Defense. As stated earlier, one of the most effective defense, called Adversarial Training (AT), consists in replacing your original training dataset by an "adversarial dataset" crafted with an attack. However, this method is time consuming, since it requires to generate adversarial example at each step of the learning phase. Nevertheless, Goodfellow et al. [21] overcome this issue with FGSM and propose an adversarial training where they replace a part of the training dataset with perturbed example. Following this principle, other works with stronger attacks propose adversarial training based on these attacks [35, 39]. Besides, Madry et al. [39] show that the adversarial training principle boils down to solve the following min-max optimization (or saddle-point problem):

$$\min_{\theta} \underset{(x,y)\sim\mathcal{D}}{\mathbb{E}} \left[\max_{\|\delta\|_p \leq \epsilon} L(f_\theta(x+\delta), y) \right], \tag{5}$$

where $\mathcal{D}$ is the unknown distribution of the dataset, f_θ is the model whose parameters is θ, L is the loss and ϵ is maximum allowed perturbations. Indeed, the equation 5 comprises two parts: maximization and minimization. On the one hand, the equation maximizes the loss regarding the noises $\delta \leq \epsilon$, *i.e.* it makes $x + \delta$ more likely to be an adversarial example. On the other hand, the equation minimizes the expectation of the loss regarding the model's parameters. To the best of our knowledge, AT based on PGD is one of the most efficient defense against attacks using the ℓ_∞-norm.

Though the methods presented above make AT feasible, it still increase the learning time of a model. Zhang et al. [66] aim at improving the computation cost of AT by reformulating it as a differential game and then derive the Pontryagin's Maximum Principle (PMP)[5]. The PMP reveals that AT are closely linked with the first layer of neural networks. Therefore they develop YOPO (You Only Propagate Once) which leverage this fact by limiting the number of forward and backward propagation without worsening the network's performance. Their experiments show that YOPO learn 4 to 5 time faster to achieve as good results as adversarial training based on PGD.

Papernot and McDaniel [44] find another mean to enhance the robustness of ML algorithms. The authors develop a method called Deep k-Nearest Neighbors (DKNN), i.e. a hybrid classifier that mixes Deep Neural Network (DNN) and kNN. Their motivation is to improve the confidence estimation, the model interpretability, and the robustness. The principle of DKNN is to find the nearest neighbors (from the training set) of an input x at each layer of the DNN and find the classes of each neighbor. This procedure allows the analysis of the evolution of classes in the neighborhood of x throughout the network. That is why, the authors introduce several metrics such as *nonconformity* and *credibility*. The nonconformity metric is used to measure the discrepancy between the labels of the neighborhood and the predicted label of an input x. A high nonconformity value means that the labels predicted for the neighborhood of x is different from the one predicted for x. The computation of the credibility measure is based on "calibration dataset" whose examples were not used for the training. Then, the credibility of an input x is the ratio of nonconformity measures of the examples from the calibration dataset that are greater than the input's nonconformity measure. DKNN increases the *confidence* in predictions thanks to the *credibility measure* which is used to assess and select model's predictions. Moreover, this algorithm becomes more interpretable because of the neighborhood it provides which gives an insight of its "internal work". Papernot and McDaniel [44] claim that DKNN would prevent adversarial examples by assigning to them a low credibiility measure. Their empirical results show particular encouraging results against C&W attack.

The drawback of the above techniques lies on their dependence to the ℓ_p-norm. Indeed, even if it brings robustness against attacks relying on the same metric distance (like PGD), the defense could become ineffective when the attacks depend on a different metric distance. Instead of searching for the perturbation that maximizes the loss function of a model (Equation 5), Zantedeschi et al. [64]

[5]PMP is used in optimal control theory to find the best solution regarding input and constraint.

propose to consider all the local perturbations (drawn from a gaussian distribution) around each examples:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathop{\mathbb{E}}_{\delta\sim\mathcal{N}(0,\sigma^2)} L(f_\theta(x+\delta), y),$$

where $\mathcal{D}$ is the unknown distribution of the dataset, f_θ is the model whose parameters is θ and L is the loss function of the model. They claim that their method, known as Gaussian Data Augmentation (GDA), instead of crafting adversarial example only in the direction of the gradient, explore much more directions around the examples. GDA outperforms AT considering adversarial accuracy measure (*i.e.* accuracy on a attacked dataset) on MNIST [36] and CIFAR-10 [32].

A hypothesis that could explain the adversarial phenomenon is the high expressiveness of the models, *i.e.* the the capacity of the model to learn complex behavior. Cissé et al. [12] and Tsuzuku et al. [60] explore this lead by constraining the Lipschitz constant of neural networks to enhance their robustness. Considering all possible couple of points of the input space of a function f, the Lipschitz constant is the smallest value that upperbound the absolute value of the slopes given by each pair of points. Intuitively, we would say that constraining the Lipschitz constant of a function f to a small value would smooth the function and therefore reduce its expressiveness. Cissé et al. [12] develop specific training called PARSEVAL TRAINING which consists in ensuring that the Lipschitz constant of all the network's layers is smaller than 1 by having Parseval tight weight's matrices. We refer interested readers to the paper [12] for the mathematical details. Tsuzuku et al. [60] propose another training method, called Lipschitz Margin Training (LMT), to constrain the Lipschitz constant of a network. They first link the margin of network to the Lipschitz constant. From that, they derive an algorithm relying on the relation between the margin and the Lipschitz constant that enhance the robustness of the network. Besides, LMT provide a "certified" lower bound on the smallest perturbation that can defeat the network. This latter property could be desirable for the demonstration of conformity to the regulation requirement of an embedded system based on ML. Indeed, theoretical results are great assets for the certification process. For example, Gourdeau et al. [22] leverage the **PAC theory** (Probably Approximately Correct) to provide theoretical proof of the feasibility of robust learning from the perspective of computational learning theory. The authors focus on the setting where the input space is the boolean hypercube $\mathcal{X} = \{0,1\}^n$ and show that robust learning is feasible or not for different classes of models.

Remark. Gilmer et al. [20] conduct experiments showing that improving adversarial robustness also improve corruption robustness, i.e. the robustness of the model against distributional shift. They want to show that both type of robustness could be the manifestation of the same phenomenon. Nevertheless, it is encouraging that improving adversarial robustness has a positive impact on corruption robustness.

4.2 Verification

The existing frameworks for verifying ML models show that is possible to verify many types of property as long as you are able to express it in the manner expected by the verification framework. However, the literature focuses on a specific property which is the robustness. For that reason we decided to distinguish *4.2.1 Properties Verification* and *4.2.2 Robustness Verification*. Basically, there are two important criteria for a verifier: *completeness* and *soundness*. *Completeness* means that the verifier can verify all the properties that hold whereas *soundness* means that the verifier cannot prove any wrong property.

4.2.1 Properties Verification. Some methods in the literature claim to do *property verification* [28, 29, 62]. As stated earlier, verifying that the desired properties hold for an ML algorithm

presupposes that we can express them as expected by the verifier, *i.e.* in the format that verifier could verify.

Katz et al. [28] develop a method that uses Simplex algorithm —a method to solve optimization problem of linear programming— and Satisfiability Modulo Theories solver (SMT solver) —a solver for formulas of first-order logic with respect to some background theories such as arithmetic, arrays, ...— which the authors adapt to work with Neural Network using specific activation function (ReLU: $y = max(0, x)$). Their method is called RELUPLEX and they run it on a case study: ACAS XU system. They use quantifier-free formulas to the formalization of properties which consists in constrained domains for inputs and outputs. Basically, they reduce the verification problem into a constraint satisfiability problem (CSP). They verified 10 properties with RELUPLEX and got a timeout for 2 of them, i.e. RELUPLEX was not able to end the verification within the given time. The longest verification took almost 5 days while the fastest took less than 8 minutes. MARABOU [29], which is the successor of RELUPLEX, is based on the same principles. The framework, now supports piecewise linear layer and activation and has a "divide and conquer mode" which improves the computation time of the verification.

4.2.2 Adversarial Robustness Verification. Robustness verification is an emerging field [12, 19, 22, 27, 57, 60]. Verifying robustness means ensuring that the algorithm outputs the same label y for an example x and its neighborhood whose the computation usually rely on metric distance (*e.g.* ℓ_p-norm) This property could be stated as follow:

$$\forall x' \text{ s.t. } \|x - x'\|_p \leq \epsilon, \quad f(x) = f(x'), \tag{6}$$

where $\|x - x'\|_p \leq \epsilon$ represent the neighborhood around x such that the distance between x and any neighbor x' does not exceed ϵ and f is the ML model. Indeed, the existence of adversarial examples could be seen as the negation of Equation 6. Thus, verifying that Equation 6 holds boils down to check the adversarial robustness of an algorithm.

Singh et al. [57] develop REFINEZONO, a verification framework mixing optimization techniques (MILP, LP/MILP relaxation[6]) and abstract interpretation —a method mainly use in static analysis that leverage overappromixation to analyse the behavior of computer programs—. Mixing those techniques allows better scalability for complete verifiers and improves the precision of incomplete ones. The principle of REFINEZONO is to compute the boundaries of the neurons using optimization techniques and abstract interpretation. As in [28], a property is expressed as domains for inputs and outputs. A robustness property expressed as *same label predicted whatever the point within a region around x* can be easily transformed into domains for input and output: the region around x is the domain for the input and the domain for the output is given to have the right class. For example, Figure 5 shows a robustness property where we verify that in the neighborhood of the point $x = (0.9, 0.7, 0.4)$ the model outputs only the desired class (the output layer predict always the same class).

Instead of trying to scale up to complete verifiers, such as MILP, with abstract interpretation [57], it is possible to make abstract interpretation as precise as possible [19]. The benefit of applying only abstract interpretation is to verify properties on large neural networks (e.g. CNN). Gehr et al. [19] develop AI^2 framework which implements abstract interpretation techniques to verify neural networks. Their framework also focus on the verification of the robustness property (Equation 6). In AI^2[19], the input is replaced by an abstract domain then it is propagated through the network thanks to abstract transformers until the output layer. Property verification is applied to the abstract domain obtained at the output layer. The method is sound but incomplete, meaning that for a

[6]Mixed Integer Linear Programming / Linear Programming

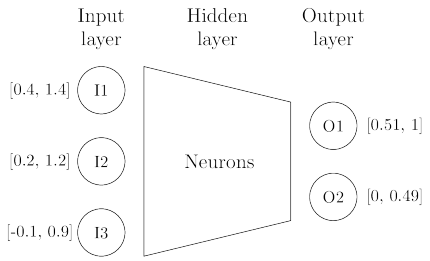


Fig. 5. Robustness property example - We consider a hypercube (of size 0.5) as a neighborhood around $x = (0.9, 0.7, 0.4)$ and the output must always be the first class, *i.e.* the lower bound of the output neuron O1 must be greater than the upper bound of the neuron O2.

property the result can either be true or inconclusive. Hence the choice of the abstract domain for the input is important because it influences the precision of the verifier.

Huang et al. [27] reduce the proof of robustness to a search of adversarial examples. They focus on the robustness of image classification which is a quite hard problem considering the possible input domain of an image and all perturbation it can have. The first assumption is that *images are discrete*. Therefore, Huang et al. [27] explore completely the region of an image x, searching for an adversarial example, by defining a set of *minimum* manipulation. To develop a verification procedure that ensures the safety of a point, the authors first formally define a *safe point*. Based on their assumption of "discrete images", the authors develop an algorithm of *safety verification* using SMT solvers; their tool is called Deep Learning Verification (DLV). Since the method of [27] is sound and complete, if the algorithm does not find any adversarial example, it means that there are none for the region and the manipulation set considered. Moreover, Huang et al. [27] show that their algorithm is scalable to big neural networks by doing experiments on neural networks with more than one hundred million of parameters.

Xu et al. [63] provide a framework, named LIRPA, that unifies the methods of linear relaxation whose goal is to find affine functions that upperbound and lowerbound the output of the network. For example, to prove a robustness property (as stated in Equation 6) with LIRPA, we must show that the lower bound of the true label is greater than alll upper bounds of all other classes; this would mean that in the worst case, the model will always output the correct class.

5 EXPLAINABILITY

As already stated earlier, explainability is a very new constraint for embedding ML-based systems. Implicitly contained in the traditionally programmed components, it has become a true challenge to demonstrate that the ML system's outcomes are trustworthy. Improving this level of confidence seems inescapable to meet the acceptance criteria of the ML application user (designer, authorities, investigators, pilot...). It has been clearly identified as a means of acceptance in the EASA AI roadmap [16].

We identify two type of explanations : local explanations and global explanation. Both subdomains suit different needs: *(i)* explanation at the prediction level (local) and *(ii)* explanation of the behavior of the model regarding inputs evolution (global). In each subdomain, we face two types of explanation techniques: *model agnostic* and *model specific*. Model agnostic means that whatever the ML algorithm, we can provide an explanation [13, 17, 23, 38, 51, 52, 67]. Model specific means that it will only work with a specific type of ML algorithm; most papers in the literature focus on DNN [24, 30, 31, 56, 65].

Table 2. Summary of explanation method. The column "method name" report the technique used when the proposed method has no name

	Method name	Papers	Model targeted
Local Explanation	MASKING MODEL	Dabkowski and Gal [13]	ALL
	MPSM	Fong and Vedaldi [17]	ALL
	LORE	Guidotti et al. [23]	ALL
	GVE	Hendricks et al. [24]	
	INFLUENCE FUNCTION	Koh and Liang [31]	
	LIME	Ribeiro et al. [51]	ALL
	ANCHORS	Ribeiro et al. [52]	ALL
	DECONVNET	Zeiler and Fergus [65]	
Global Explanation	SHAP	Lundberg and Lee [38]	ALL
	DEEPLIFT	Shrikumar et al. [56]	
	NPSEM	Zhao and Hastie [67]	ALL

where ALL = model agnostic and  = DNN specific.

5.1 Local Explanation

Computer Vision is a complex domain because of the nature of the data we have to deal with: images or videos. Since the development of DNN, it has become easier to reach good performance on image classification, object detection, ... However, we still do not fully understand why methods based on DNN work so well. Hence researchers develop new methods to explain, assess, and increase the confidence we can have in the decision taken by DNN for computer vision tasks.

5.1.1 Model agnostic. We will first speak of 3 systems: LIME (Local Interpretable Model-agnostic Explanations) [51], ANCHORS [52] and LORE (LOcal Ruled-based Explanation) [23]. Note that LORE is not applicable to images. These techniques use inputs and outputs and try to approximate locally the behavior of the ML algorithm. They are based on the same principle: generate a neighborhood around a sample x and provide an explanation for the sample x thanks to the neighborhood. If the reader wants to go further, Garreau and von Luxburg [18] provide theoretical explanations of LIME.

The generation of the neighborhood differs according to the system. LIME and ANCHORS use an interpretable representation and apply some perturbation on it. An interpretable representation could be a binary vector where the ones are the relevant information and a perturbation could be the modification on the vector (0 switches to 1) but this should be defined according to the problem that needs to be solved. Ribeiro et al. [51] define metrics to measure the distance between the neighbors and the original sample. ANCHORS and LIME are using the same method of neighborhood generation. LORE uses a genetic algorithm to generate a balanced neighborhood. The genetic algorithm is tuned in order to create the best possible neighbors around the desired sample.

Since the neighborhood is generated, it remains to provide the explanation. In LIME linear models are used while in ANCHORS decision trees are used. For the ANCHORS explanation, "anchors" are used, i.e. the minimum number of features that lead to the right prediction. Note that, Ribeiro et al. [52] define several bandit algorithms in order to find the best anchors. An explanation given by LIME correspond to the area of the image that help to take the decision. ANCHORS and LORE provide explanation in the IF-THEN form. Guidotti et al. [23] (LORE) extract explanations from the decision

- LORE

$r =$ *({credit_amount > 836, housing = own, other_debtors = none, credit_history = critical account} ⟶ decision = 0)*

$\Phi =$ *{ ({credit_amount ≤ 836, housing = own, other_debtors = none, credit_history = critical account} ⟶ decision = 1), ({credit_amount > 836, housing = own, other_debtors = none, credit_history = all paid back} ⟶ decision = 1) }*

Fig. 6. Explanations of LORE. (taken from Guidotti et al. [23]. We only use the part of figure 9 of Guidotti et al. [23] that concerns their method, LORE.)

tree while Ribeiro et al. [52] (ANCHORS) only check the presence or absence of anchors. Figure 6 shows an example of an explanation given by LORE: the r corresponds to the explanation of the decision and Φ corresponds to the minimum changes that should occur to flip the decision. The original decision (0) comes from an algorithm which was trained on the German dataset[7] to recognize good ("0") or bad ("1") creditor according to a set of attributes (age, sex, job, credit amount, duration, ...).

A popular method to explain decision on images is the *saliency map* [1, 13, 17]. Intuitively, a saliency map higlights the features, *i.e.* the pixels on images, that the model considers to take its decision. Fong and Vedaldi [17] propose an agnostic gradient-based method which considers explanation as meta-predictors. A meta-predictor is a rule that is used to explain the prediction of the model. One advantage of using meta-predictors as explanation is that you can measure the "faithfulness" of your explanation by computing its prediction error, *i.e.* it represent the number of time that the model and the rule disagree on the prediction. Then, they claim that a good explanatory rule (*i.e.* meta-predictor) to produce a saliency map rely on the local explanation principle. Indeed, they want to study the behavior of the model using the neighborhood of the original example. This neighborhood is given by perturbing the original example; that is why they introduce the notion of "meaningful pertubations". They argue that the explanation would be better, if the perturbations applied to build the neighborhood of the original example mimic *natural image effect*. Based on meaningful perturbations[8], the authors define an optimization problem which tries to find the smallest area of the original image that must be perturbed in order to reduce prediction probability of the true class of the original image; This method then outputs the saliency map which give the explanation of the model's prediction. In table 2, we named their method Meaningful Perturbation Saliency Map (MPSM).

On the other side, Dabkowski and Gal [13] develop a MASKING MODEL that learn how to generate interpretable and accurate saliency map for ML model. The authors craft a new objective function which ensures the quality of the saliency map: it ensures that the region is necessary to the good classification while its absence leads to low probability to pick the good class and at the same time it penalizes large and not smooth region. The MASKING MODEL comprises features filter (input) given by a trained network and upsampler layer which upscale the input into the original image dimension in order to provide a mask. Dabkowski and Gal [13] train the MASKING MODEL by directly minimizing its new objective function; the MASKING MODEL outputs saliency map with really accurate salient area (Figure 7).

5.1.2 DNN specific. Adebayo et al. [1] develop a methodology to test the usefulness of a saliency method for explanation. It is based on two tests: model parameter randomization test and data randomization test. A saliency method would fail the tests if it shows the same result for the

[7] https://archive.ics.uci.edu/ml/datasets/statlog+(german+credit+data)

[8] the authors used blur, noise as meaningful perturbation



(a) Input Image (b) Generated saliency map (c) Image multiplied by the mask (d) Image multiplied by inverted mask

Fig. 7. An example of explanations produced by Dabkowski and Gal [13]. The top row shows the explanation for the "Egyptian cat" while the bottom row shows the explanation for the "Beagle". Note that produced explanations can precisely both highlight and remove the selected object from the image.

randomized case and the trained case. A failure points out the independency of the saliency method regarding the architecture parameters or labeled data. With the methodology proposed by Adebayo et al. [1], we can test model-agnostic and model-specific methods.

Few years before the works previously presented, Zeiler and Fergus [65] introduced a method that allow to visualize the internal working of convolutional neural networks (CNN). Their motivation was to understand how and why CNN work well by studying the hidden layer of CNN. The principle is to sample the features learned by a neural network, back to the original image dimension. However, contrary to the method seen until now, the goal of the was not to provide a saliency map that explain the model's predictions but only to visualize the feature that a neural network learned for classifying. The system develop by Zeiler and Fergus [65] is known as multi-layered Deconvolutional Network (DECONVNET). One advantage of this technique is the possibility to start from any layer in order to see at this point the features learned by the neural network.

Another way of explaining an image is literally to provide written explanations: Hendricks et al. [24] propose a *visual explanation* for images[9]. Visual explanations are defined as *class discriminative* and *accurately described*, i.e. the textual explanation highlights elements of the image which are specific to the class. The authors use VGG, a CNN for image classification, and add to it an LSTM that generates visual explanations; the efficiency of this system relies on the loss function. Precisely, Hendricks et al. [24] define two losses for their network: one for the accurate description (relevance loss) that handles the probability of word occurrence in the sentence and the other for the class discrimination (discriminative loss) which is based on a reward function. The authors reach good results on an experiment that deals with bird classification. Figure 8, taken from the paper [24] shows the difference between visual explanation, image description, and class definition.

Koh and Liang [31] through the use of INFLUENCE FUNCTION increase in the explanability of DNN. Unlike the other methods whose the explanability was increased by providing evidences

[9]their method is denoted "GVE" (Generating Visual Explanation) in table 2.

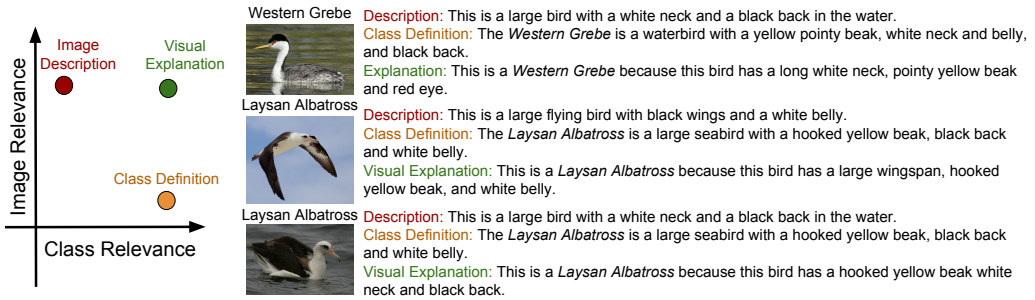


Fig. 8. Visual explanations are both image relevant and class relevant. In contrast, image descriptions are image relevant, but not necessarily class relevant, and class definitions are class relevant but not necessarily image relevant. (taken from the work of Hendricks et al. [24])

directly on the examples, Koh and Liang [31] to explain the prediction of an example, they provide the training datas that leads to that prediction. The principle of their method is to compute the impact on the predictions while modifying the training dataset. For explanability purpose, they study the effect of the removal of training data. The removed training data which strongly decrease the probability to get the right class of a given example are actually the data that leads to the prediction. Actually, the training data given as an explanation of a prediction provide insight on the behavior of the model. Indeed, by looking at the data that influence towards the prediction, it is possible to detect abnormal behavior of the model.

5.2 Global Explanation

As stated earlier, global explanation is the one that could explain the evolution of the model's outputs according to the trends of the inputs. A popular method is known as "feature importance". The principle of features importance is to find out which input features play a significant role in the decision of the ML algorithm [38, 56, 67].

5.2.1 Model agnostic. One way of exploit the feature importance principle is to use the Shapley value which comes from cooperative game theory. The idea is to find the fair share of the gain between players in a cooperative game regarding the involvement of each player in the coalition. Lundberg and Lee [38] develop SHAP (SHapley Additive exPlanations) that use the Shapley value in order to provide explanations for the predictions of ML algorithm. In this context, the features are the players and the prediction is the payout. Then, SHAP compute the contribution of each feature to the prediction. Since the computation of the Shapley value requires all the possible permutation of the players (features), the problem becomes intractable for inputs with numerous number of features. However, the authors provide an efficient way to compute an approximation of the Shapley value.

Zhao and Hastie [67] study the importance of the feature through a causal model, called Non-Parametric Structural Equation Model (NPSEM), and using Partial Dependence Plot to visualize the results. Thanks to this approach they go further than just compute the impact of features on the prediction and manage to catch the tendency of a prediction according to the evolution of features.

5.2.2 DNN specific. Shrikumar et al. [56] bring a new way of handling features importance with DEEPLIFT (Deep Learning Important FeaTures). The principle is to compare inputs to an input reference and outputs to an output reference. These references represent a kind of *neutral* behavior of the neural network. Since these references are considered as baseline, the authors backpropagate

the differences between the references and the actual sample through the network by comparing the activation values. Then, they define a contribution score system that derives the features importance from the differences. Hence, at the end of the backpropagation, we end up with the importance of each feature for a given prediction. Intuitively, we would say that the greater the difference is, the more it has an impact on the feature importance. The efficiency of the algorithm depends on the choice of references that relies on domain-specific knowledge.

6 CONCLUSION

In this survey we have investigated the compatibility between recent advances in machine learning and established certification procedures used in civil avionic systems. Machine learning proves to be an interesting approach for developing functionalities that are difficult or impossible to implement using classical programming, such as vision-based navigation (e.g. ATTOL project from Airbus [2]). This has the potential to offer meaningful improvements in the area of the system safety, and opens the way to advanced features in autonomous flight systems. However, the intrinsic characteristics of machine learning based software, such as the use of alternative requirement specification techniques or the difficult traceability of the results, seriously challenge established certification procedures.

As a first step towards certifiable embedded system based on ML, we focused on Adversarial Robustness and Explainability that will eventually support the proof of conformity to regulation requirements. With Robustness, we address the *innocuity* issue, inherent concept of ML algorithm inducing the necessary management of unintended behaviors of the system. With Explainability, we address the *trustworthiness* issue in ML algorithms which are still suffering from their black-box aspect. For both domains, our review of the state of the art shows promising methods that address those issues. It is interesting to notice that most of verification methods (cf. Section 4.2) are based on formal methods that have already been used for the demonstration of conformity since a while. Assuming that verification is the most time and cost consuming part of the development process, formal methods are promising means to handle the robustness aspects because since it provides mathematical evidences, it allows avoiding a huge amount of empirical verification. Our survey takes a closer look at some particular issues. The ultimate goal of certifying embedded systems based on ML, requires to address all the various issues identified in Figure 3 and thus demands further research in the other domains.

REFERENCES

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian J. Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *NeurIPS*.

[2] Airbus. 2020. Autonomous Taxi, Take-Off and Landing (ATTOL). https://www.airbus.com/newsroom/press-releases/en/2020/01/airbus-demonstrates-first-fully-automatic-visionbased-takeoff.html

[3] Saleema Amershi, Andrew Begel, Christian Bird, Robert DeLine, Harald C. Gall, Ece Kamar, Nachiappan Nagappan, Besmira Nushi, and Thomas Zimmermann. 2019. Software Engineering for Machine Learning: A Case Study. In *ICSE (SEIP)*, Helen Sharp and Mike Whalen (Eds.). IEEE / ACM, 291–300.

[4] Rob Ashmore, Radu Calinescu, and Colin Paterson. 2019. Assuring the Machine Learning Lifecycle: Desiderata, Methods, and Challenges. *CoRR* abs/1905.04223 (2019).

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR*.

[6] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, Chiu Yuen Koo, Lukasz Lew, Clemens Mewald, Akshay Naresh Modi, Neoklis Polyzotis, Sukriti Ramesh, Sudip Roy, Steven Euijong Whang, Martin Wicke, Jarek Wilkiewicz, Xin Zhang, and Martin Zinkevich. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *ACM SIGKDD*. ACM, 1387–1395.

[7] Battista Biggio and Fabio Roli. 2018. Wild Patterns: Ten Years After the Rise of Adversarial Machine Learning. *Pattern Recognit.* 84 (2018), 317–331.

[8] Li Cai and Yangyong Zhu. 2015. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Science Journal* 14 (2015).

[9] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian J. Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On Evaluating Adversarial Robustness. *CoRR* abs/1902.06705 (2019).

[10] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *IEEE SP*. IEEE Computer Society, 39–57.

[11] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. 2017. ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models. In *AISec@CCS 2017*. ACM, 15–26.

[12] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann N. Dauphin, and Nicolas Usunier. 2017. Parseval Networks: Improving Robustness to Adversarial Examples. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 854–863.

[13] Piotr Dabkowski and Yarin Gal. 2017. Real Time Image Saliency for Black Box Classifiers. In *NeurIPS*. 9525–9536.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. Association for Computational Linguistics, 4171–4186.

[15] EASA. 2018. *CS-25 Amendment 22*. https://www.easa.europa.eu/sites/default/files/dfu/CS-25%20Amendment%2022.pdf

[16] EASA. 2020. *EASA Artificial Intelligence Roadmap 1.0*. https://www.easa.europa.eu/sites/default/files/dfu/EASA-AI-Roadmap-v1.0.pdf

[17] Ruth C. Fong and Andrea Vedaldi. 2017. Interpretable Explanations of Black Boxes by Meaningful Perturbation. In *ICCV*. IEEE Computer Society, 3449–3457.

[18] Damien Garreau and Ulrike von Luxburg. 2020. Explaining the Explainer: A First Theoretical Analysis of LIME. In *AISTATS (Proceedings of Machine Learning Research, Vol. 108)*. PMLR, 1287–1296.

[19] Timon Gehr, Matthew Mirman, Dana Drachsler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin T. Vechev. 2018. AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. In *IEEE SP*. IEEE Computer Society, 3–18.

[20] Justin Gilmer, Nicolas Ford, Nicholas Carlini, and Ekin D. Cubuk. 2019. Adversarial Examples Are a Natural Consequence of Test Error in Noise. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2280–2289.

[21] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

[22] Pascale Gourdeau, Varun Kanade, Marta Kwiatkowska, and James Worrell. 2019. On the Hardness of Robust Classification. In *NeurIPS*. 7444–7453.

[23] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. 2018. Local Rule-Based Explanations of Black Box Decision Systems. *CoRR* abs/1806.09936 (2018).

[24] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. 2016. Generating Visual Explanations. In *ECCV (Lecture Notes in Computer Science, Vol. 9908)*. Springer, 3–19.

[25] Qian Huang, Isay Katsman, Zeqi Gu, Horace He, Serge J. Belongie, and Ser-Nam Lim. 2019. Enhancing Adversarial Example Transferability With an Intermediate Level Attack. In *ICCV*. IEEE, 4732–4741.

[26] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. 2020. A Survey of Safety and Trustworthiness of Deep Neural Networks: Verification, Testing, Adversarial Attack and Defence, and Interpretability. *Computer Science Review* 37 (2020), 100–270.

[27] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *CAV (Lecture Notes in Computer Science, Vol. 10426)*. Springer, 3–29.

[28] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. In *CAV (Lecture Notes in Computer Science, Vol. 10426)*. Springer, 97–117.

[29] Guy Katz, Derek A. Huang, Duligur Ibeling, Kyle Julian, Christopher Lazarus, Rachel Lim, Parth Shah, Shantanu Thakoor, Haoze Wu, Aleksandar Zeljic, David L. Dill, Mykel J. Kochenderfer, and Clark W. Barrett. 2019. The Marabou Framework for Verification and Analysis of Deep Neural Networks. In *CAV (Lecture Notes in Computer Science, Vol. 11561)*. Springer, 443–452.

[30] Alex Kendall and Yarin Gal. 2017. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?. In *NeurIPS*. 5574–5584.

[31] Pang Wei Koh and Percy Liang. 2017. Understanding Black-box Predictions via Influence Functions. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 1885–1894.

[32] A. Krizhevsky. 2009. Learning Multiple Layers of Features from Tiny Images.

[33] Jim Krodel. 2008. Technology Changes In Aeronautical Systems. In *Embedded Real Time Software and Systems (ERTS2008)*. https://hal.archives-ouvertes.fr/hal-02269834

[34] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *ICLR*. OpenReview.net.

[35] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial Machine Learning at Scale. In *ICLR*. OpenReview.net.

[36] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* 86, 11 (1998), 2278–2323.

[37] Qizhang Li, Yiwen Guo, and Hao Chen. 2020. Practical No-box Adversarial Attacks against DNNs. In *NeurIPS 2020*.

[38] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *NeurIPS*. 4765–4774.

[39] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *ICLR*. OpenReview.net.

[40] Robert Martin. 2017. Assured Software - A journey and discussion. "https://www.his-2019.co.uk/session/cwe-cve-its-history-and-future"

[41] David A. McAllester. 1999. Some PAC-Bayesian Theorems. *Mach. Learn.* 37, 3 (1999), 355–363.

[42] Matthew Mirman, Timon Gehr, and Martin T. Vechev. 2018. Differentiable Abstract Interpretation for Provably Robust Neural Networks. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 3575–3583.

[43] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *CVPR*. IEEE Computer Society, 2574–2582.

[44] Nicolas Papernot and Patrick D. McDaniel. 2018. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *CoRR* abs/1803.04765 (2018).

[45] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2016. The Limitations of Deep Learning in Adversarial Settings. In *IEEE EuroS&P*. IEEE, 372–387.

[46] Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. 2012. PAC-Bayes Bounds with Data Dependent Priors. *Journal of Machine Learning Research* 13, 112 (2012), 3507–3531.

[47] P Phillips, Amanda Hahn, Peter Fontana, David Broniatowski, and Mark Przybocki. 2020. Four Principles of Explainable Artificial Intelligence (Draft). https://doi.org/10.6028/NIST.IR.8312-draft

[48] Sylvaine Picard, Camille Chapdelaine, Cyril Cappi, Laurent Gardes, Eric Jenn, Baptiste Lefèvre, and Thomas Soumarmon. 2020. Ensuring Dataset Quality for Machine Learning Certification. In *ISSRE*. IEEE, 275–282.

[49] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *CVPR*. IEEE Computer Society, 779–788.

[50] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. 2020. Adversarial Attacks and Defenses in Deep Learning. *Engineering* 6, 3 (2020), 346 – 360.

[51] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *SIGKDD*. ACM, 1135–1144.

[52] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. 2018. Anchors: High-Precision Model-Agnostic Explanations. In *AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18)*. AAAI Press, 1527–1535.

[53] John Rushby. 2015. *The Interpretation and Evaluation of Assurance Cases.* Technical Report SRI-CSL-15-01. Computer Science Laboratory, SRI International, Menlo Park, CA. Available at url=http://www.csl.sri.com/users/rushby/papers/sri-csl-15-1-assurance-cases.pdf.

[54] Hadi Salman, Greg Yang, Huan Zhang, Cho-Jui Hsieh, and Pengchuan Zhang. 2019. A Convex Relaxation Barrier to Tight Robustness Verification of Neural Networks. In *NeurIPS*. 9832–9842.

[55] John Shawe-Taylor and Robert C. Williamson. 1997. A PAC Analysis of a Bayesian Estimator. In *COLT*, Yoav Freund and Robert E. Schapire (Eds.). ACM, 2–9.

[56] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. 2017. Learning Important Features Through Propagating Activation Differences. In *ICML (Proceedings of Machine Learning Research, Vol. 70)*. PMLR, 3145–3153.

[57] Gagandeep Singh, Timon Gehr, Markus Püschel, and Martin Vechev. 2019. Boosting Robustness Certification of Neural Networks. In *ICLR*. OpenReview.net.

[58] Stefan Studer, Thanh Binh Bui, Christian Drescher, Alexander Hanuschkin, Ludwig Winkler, Steven Peters, and Klaus-Robert Müller. 2020. Towards CRISP-ML(Q): A Machine Learning Process Model with Quality Assurance Methodology. *CoRR* abs/2003.05155 (2020).

[59] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

[60] Yusuke Tsuzuku, Issei Sato, and Masashi Sugiyama. 2018. Lipschitz-Margin Training: Scalable Certification of Perturbation Invariance for Deep Neural Networks. In *NeurIPS*. 6542–6551.

[61] L. G. Valiant. 1984. A Theory of the Learnable. *Commun. ACM* 27, 11 (Nov. 1984), 1134–1142.

[62] Shiqi Wang, Kexin Pei, Justin Whitehouse, Junfeng Yang, and Suman Jana. 2018. Formal Security Analysis of Neural Networks using Symbolic Intervals. In *USENIX Security*. USENIX Association, 1599–1614.

[63] Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic Perturbation Analysis for Scalable Certified Robustness and Beyond. In *NeurIPS*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin (Eds.).

[64] Valentina Zantedeschi, Maria-Irina Nicolae, and Ambrish Rawat. 2017. Efficient Defenses Against Adversarial Attacks. In *ACM Workshop on Artificial Intelligence and Security, AISec@CCS*.
[65] Matthew D. Zeiler and Rob Fergus. 2014. Visualizing and Understanding Convolutional Networks. In *ECCV (Lecture Notes in Computer Science, Vol. 8689)*. Springer, 818–833.
[66] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. 2019. You Only Propagate Once: Accelerating Adversarial Training via Maximal Principle. In *NeurIPS*. 227–238.
[67] Qingyuan Zhao and Trevor Hastie. 2019. Causal Interpretations of Black-Box Models. *JBES* 0, 0 (2019), 1–10.