TECHNICAL REPORT

ISO/IEC TR 24029-1

First edition
2021-03

Artificial Intelligence (AI) — Assessment of the robustness of neural networks —

Part 1: Overview

Reference number
ISO/IEC TR 24029-1:2021(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT), see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 42, *Artificial intelligence*.

A list of all parts in the ISO/IEC 24029 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

When designing an AI system, several properties are often considered desirable, such as robustness, resiliency, reliability, accuracy, safety, security, privacy. A definition of robustness is provided in 3.6. Robustness is a crucial property that poses new challenges in the context of AI systems. For example, in AI systems there are some risks specifically tied to the robustness of AI systems. Understanding these risks is essential for the adoption of AI in many contexts. This document aims at providing an overview of the approaches available to assess these risks, with a particular focus on neural networks, which are heavily used in industry, government and academia.

In many organizations, software validation is an essential part of putting software into production. The objective is to ensure various properties including safety and performance of the software used in all parts of the system. In some domains, the software validation and verification process is also an important part of system certification. For example, in the automotive or aeronautic fields, existing standards, such as ISO 26262 or Reference [2], require some specific actions to justify the design, the implementation and the testing of any piece of embedded software.

The techniques used in AI systems are also subject to validation. However, common techniques used in AI systems pose new challenges that require specific approaches in order to ensure adequate testing and validation.

AI technologies are designed to fulfil various tasks, including interpolation/regression, classification and other tasks.

While many methods exist for validating non-AI systems, they are not always directly applicable to AI systems, and neural networks in particular. Neural network systems represent a specific challenge as they are both hard to explain and sometimes have unexpected behaviour due to their non-linear nature. As a result, alternative approaches are needed.

Methods are categorized into three groups: statistical methods, formal methods and empirical methods. This document provides background on these methods to assess the robustness of neural networks.

It is noted that characterizing the robustness of neural networks is an open area of research, and there are limitations to both testing and validation approaches.

Artificial Intelligence (AI) — Assessment of the robustness of neural networks —

Part 1: Overview

1 Scope

This document provides background about existing methods to assess the robustness of neural networks.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
artificial intelligence
AI
<system>capability of an engineered system to acquire, process and apply knowledge and skills

3.2
field trial
trial of a new system in actual situations for which it is intended (potentially with a restricted user group)

Note 1 to entry: Situation encompasses environment and process of usage.

3.3
input data
data for which a deployed machine learning model calculates a predicted output or inference

Note 1 to entry: Input data is also referred to by machine learning practitioners as out-of-sample data, new data and production data.

3.4
neural network
neural net
NN
artificial neural network
ANN
network of primitive processing elements connected by weighted links with adjustable weights, in which each element produces a value by applying a non-linear function to its input values, and transmits it to other elements or presents it as an output value

Note 1 to entry: Whereas some neural networks are intended to simulate the functioning of neurons in the nervous system, most neural networks are used in artificial intelligence as realizations of the connectionist model.

Note 2 to entry: Examples of non-linear functions are a threshold function, a sigmoid function and a polynomial function.

[SOURCE: ISO/IEC 2382:2015, 2120625, modified — Abbreviated terms have been added under the terms and Notes 3 to 5 to entry have been removed.]

3.5
requirement
statement which translates or expresses a need and its associated constraints and conditions

[SOURCE: ISO/IEC/IEEE 15288:2015, 4.1.37]

3.6
robustness
ability of an AI system to maintain its level of performance under any circumstances

Note 1 to entry: This document mainly describes data input circumstances such as domain change but the definition is broader not to exclude hardware failure and other types of circumstances.

3.7
testing
activity in which a system or component is executed under specified conditions, the results are observed or recorded, and an evaluation is made of some aspect of the system or component

[SOURCE: ISO/IEC/IEEE 26513:2017, 3.42]

3.8
test data
subset of *input data* (3.3) samples used to assess the generalization error of a final machine learning (ML) model selected from a set of candidate ML models

[SOURCE: Reference [2]]

3.9
training dataset
set of samples used to fit a machine learning model

3.10
validation
confirmation, through the provision of objective evidence, that the *requirements* (3.5) for a specific intended use or application have been fulfilled

[SOURCE: ISO/IEC 25000:2014, 4.41, modified — Note 1 to entry has been removed.]

3.11
validation data
subset of *input data* (3.3) samples used to assess the prediction error of a candidate machine learning model

Note 1 to entry: Machine learning (ML) model *validation* (3.10) can be used for ML model selection.

[SOURCE: Reference [2]]

3.12
verification
confirmation, through the provision of objective evidence, that specified requirements have been fulfilled

[SOURCE: ISO/IEC 25000:2014, 4.43, modified — Note 1 to entry has been removed.]

4 Overview of the existing methods to assess the robustness of neural networks

4.1 General

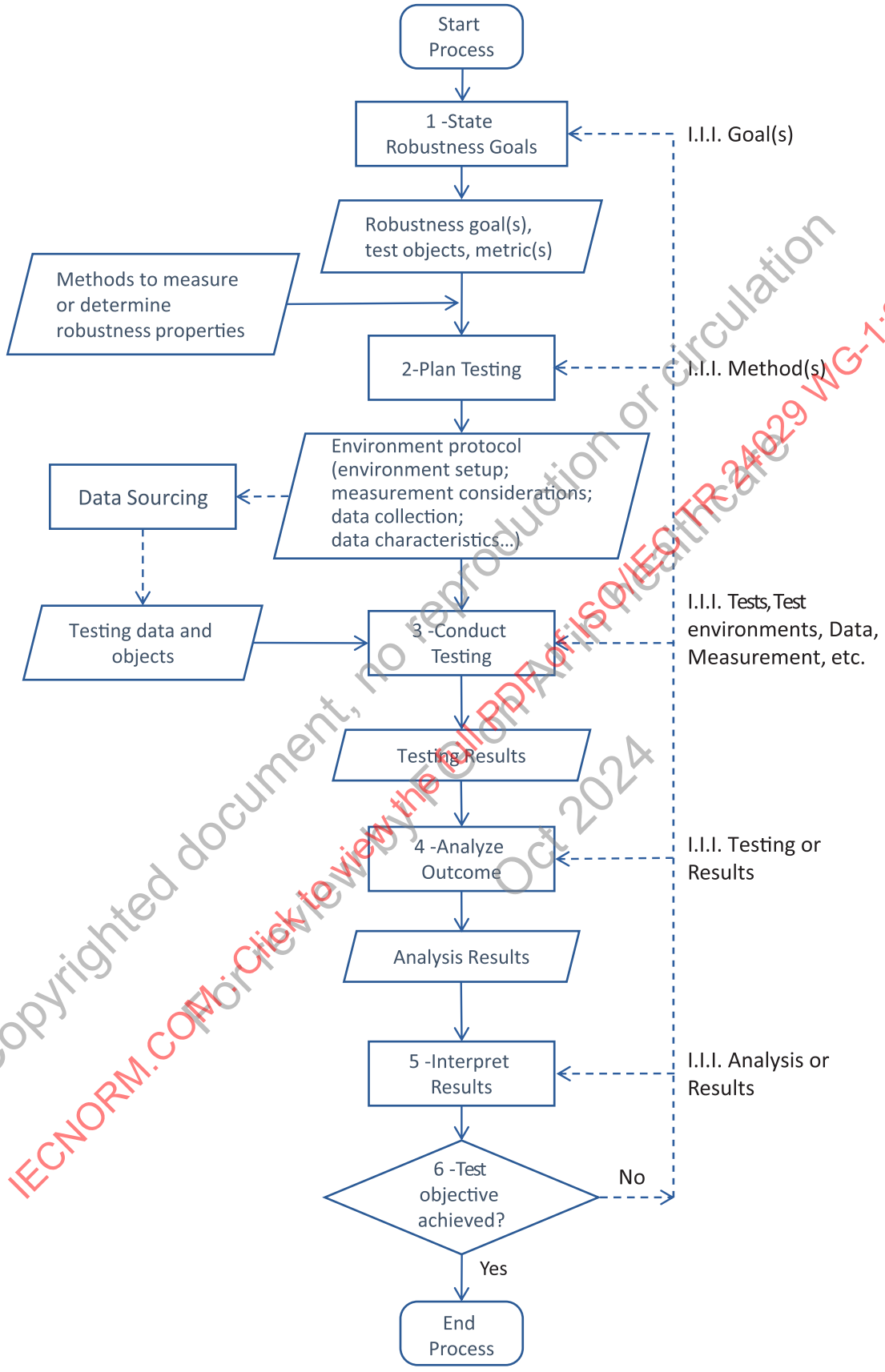
4.1.1 Robustness concept

Robustness goals aim at answering the question "To what degree is the system required to be robust?" or "What are the robustness properties of interest?". Robustness properties demonstrate the degree to which the system performs with atypical data as opposed to the data expected in typical operations.

4.1.2 Typical workflow to assess robustness

This subclause explains how the robustness of neural networks is assessed for different classes of AI applications such as classification, interpolation and other complex tasks.

There are different ways to assess the robustness of neural networks using objective information. A typical workflow for determining neural network (or other technique) robustness is as shown in Figure 1.



Key

I.I.I.	incomplete, incorrect or insufficient
	start/end
	step
	input/output
	decision

Figure 1 — Typical workflow to determine neural network robustness

Step 1: State robustness goals

The process begins with a statement of the robustness goals. During this initial step, the targets to be tested for robustness are identified. The metrics to quantify the objects that demonstrate the achievement of robustness are subsequently identified. This constitutes the set of decision criteria on robustness properties that can be subject to further approval by relevant stakeholders (see ISO/IEC/IEEE 16085:2021, 7.4.2).

Step 2: Plan testing

This step plans the tests that demonstrate robustness. The tests rely on different methods, for example: statistical, formal or empirical methods. In practice, a combination of methods is used. Statistical approaches usually rely on a mathematical testing process and are able to illustrate a certain level of confidence in the results. Formal methods rely on formal proofs to demonstrate a mathematical property over a domain. Empirical methods rely on experimentation, observation and expert judgement. In planning the testing, the environment setup needs to be identified, data collection planned, and data characteristics defined (that is, which data element ranges and data types will be used, which edge cases will be specified to test robustness, etc.). The output of Step 2 is a testing protocol that comprises a document stating the rationale, objectives, design and proposed analysis, methodology, monitoring, conduct and record-keeping of the tests (more details of the content of a testing protocol are available through the definition of the clinical investigation plan found in ISO 14155:2020, 3.9).

Step 3: Conduct testing

The testing is then conducted according to the defined testing protocol, and outcomes are collected. It is possible to perform the tests using a real-world experiment or a simulation, and potentially a combination of these approaches.

Step 4: Analyze outcome

After completion, tests outcomes are analysed using the metrics chosen in Step 1.

Step 5: Interpret results

The analysis results are then interpreted to inform the decision.

Step 6: Test objective achieved?

A decision on system robustness is then formulated given the criteria identified earlier and the resulting interpretation of the analysis results.

If the test objectives are not met, an analysis of the process is conducted and the process returns to the appropriate preceding step, in order to alleviate deficiencies, e.g. add robustness goals, modify or add metrics, add consideration of different aspects to measure, re-plan tests, etc.

AI systems that significantly rely on neural networks, particularly deep neural networks (DNN), bear built-in malfunctions. These malfunctions are showing up by a system behaviour that resembles

an occurrence of a conventional software. Typical situations have been demonstrated by feeding "adversarial examples" to object recognition systems, e.g. in Reference [5]. These built-in errors of DNNs are not simple to "fix". Research on this problem shows that there are measures to improve the robustness of DNNs with respect to adversarial examples, but this works to a certain degree only [6],[7]. However, if detected during a test procedure, the AI system is able to signal a problem when an associated input pattern is encountered.

Data sourcing:

Data sourcing is the process of selecting, producing and/or generating the testing data and objects that are needed for conducting the testing.

This sometimes includes consideration of legal or other regulatory requirements, as well as practical or technical issues.

The testing protocol contains the requirements and the criteria necessary for data sourcing. Data sourcing issues and methods are not covered in detail in this document.

Especially the following issues can have an impact on robustness:

— scale;

— diversity, representativeness, and range of outliers;

— choice of real or synthetic data;

— datasets used specifically for robustness testing;

— adversarial and other examples that explore hypothetic domain extremes;

— composition of training, testing, and validation datasets.

4.2 Classification of methods

Following the workflow defined above for determining robustness, the remainder of this document describes the methods and metrics applicable to the various testing types, i.e. statistical, formal and empirical methods.

Statistical approaches usually rely on a mathematical testing process on some datasets, and help ensure a certain level of confidence in the results. Formal methods rely on a sound formal proof in order to demonstrate a mathematical property over a domain. Formal methods in this document are not constrained to the traditional notion of syntactic proof methods and include correctness checking methods, such as model checking. Empirical methods rely on experimentation, observation and expert judgement.

While it is possible to characterize a system through either observation or proof, this document chooses to separate observation techniques into statistical and empirical methods. Statistical methods generate reproducible measures of robustness based on specified datasets. Empirical methods produce data that can be analysed with statistical methods but is not necessarily reproducible due to the inclusion of subjective assessment. Therefore, it is usually necessary that methods from both categories be performed jointly.

Thus, this document first considers statistical approaches which are the most common approaches used to assess robustness. They are characterized by a testing approach defined by a methodology using mathematical metrics. This document then examines approaches to attain a formal proof that are increasingly being used to assess robustness. Finally, this document presents empirical approaches that rely on subjective observations that complement the assessment of robustness when statistical and formal approaches are not sufficient or viable.

In practice, in the current state of the art, these methods are not used to directly assess robustness as a whole. Instead, they each target complementary aspects of robustness, providing several partial indicators whose conjunction enables robustness assessment.

For an evaluator, it is indeed possible to use these methods to answer different kinds of questions on the system they intend to validate. For example:

— statistical methods allow the evaluator to check if the systems properties reach a desired target threshold (e.g. how many defective units are produced?);

— formal methods allow the evaluator to check if the properties are provable on the domain of use (e.g. does the system always operate within the specified safety bounds?);

— empirical methods allow the evaluator to assess the degree to which the system's properties hold true in the scenario tested (e.g. is the observed behaviour satisfactory?).

The principle of applying such methods to robustness assessment is to evaluate to which extent these properties hold when circumstances change:

— when using statistical methods: how is the measured value of performance affected when changing the conditions?

— when using formal methods: do the new conditions still belong to the domain where the properties are provable?

— when using empirical methods: do the properties still hold true in other scenarios?

It is noted that characterizing the robustness of neural networks is an active area of research, and there are limitations to both testing and validation approaches. With testing approaches, the variation of possible inputs is unlikely to be large enough to provide any guarantees on system performance. With validation approaches, approximations are usually required to handle the high dimensionality of inputs and parameterizations of a neural network.

5 Statistical methods

5.1 General

One aspect of robustness is the effect of changing circumstances on quantitative performance, which statistical methods are particularly suited to measure. They enable this assessment by direct evaluation of the performance in various scenarios using comparative measures.

When using statistical methods, the four following main criteria are used in the computation of robustness:

1) **Appropriate testing data.** To evaluate the robustness of a model, a dataset that spans the distribution and the input conditions of interest for the target application is first established, either through acquisition of real measurement data or simulated data. Several sources for the data are possible, such as noisy data that was not accounted for during the initial training of the model, data from similar domain applications, data from a different but equivalent data source. While there is no general method to assess the relevance of a dataset and it often relies on human judgment, some techniques exist (e.g. based on intermediate representations of the data) to support this analysis with various indicators. The evaluation of the robustness of neural network models can vary with different testing datasets.

2) **Choose a setting of the model.** The evaluation can also assess the robustness using different settings of the trained model (for example the model precision, quantized weight, etc).

3) **A choice of metrics or metrics of performance.** Based on the context, the task at hand and the nature of the data, some metrics are not always appropriate, as they can lead to irrelevant or misleading results. An appropriate set of metrics (see 5.2) helps to avoid these situations.

4) **A method for a decision on robustness.** Given a selected metric, an appropriate statistical test is performed to reach a decision regarding whether the model is sufficiently robust for the chosen robustness goal(s) or not.

A robustness property assessed through statistical methods is defined by one or more thresholds over a set of metrics that need to hold on some testing data. The evaluation of robustness is case-specific, given that certain organizations or situations require different robustness goals and metrics to determine if a goal is met.

This clause follows the general workflow described in Figure 1 to assess the robustness of a neural network. In particular, it focuses on Steps 1, 2 and 3 of the workflow defined in 4.1.2, i.e. state robustness goal, plan testing and conduct testing.

Subclauses 5.2 and 5.3 present metrics and methods to assess the robustness of neural networks statistically, more detailed information on each is available in References [8],[9],[10] and[11].

5.2 Robustness metrics available using statistical methods

5.2.1 General

This subclause presents background information about statistical metrics that are available and typically used on the output of neural networks. It describes the robustness goals, using Step 1 of Figure 1. Robustness goals need t3o be well defined. For example, to say simply that "the trained neural network has to be robust to inputs dissimilar to those on which it has been trained" is not sufficiently well defined. It is possible for a neural network to have full compliance or no compliance with this goal depending on the input. For example, it is possible for a neural network to be fully robust to inputs that follow a different distribution from the initial training and testing sets but remains within the scope of the domain. On the other hand, it is likely to have a neural network that is not compliant at all if the inputs are in a completely different domain than those it was trained for. Therefore, the robustness goal needs to be stated sufficiently to enable meaningful determination of a neural network's robustness.

An example of a well-defined goal (structured in three parts) is as follows.

1) The trained neural network needs to be robust to inputs dissimilar to those on which it has been trained.

2) Inputs are assumed to be from the same domain and can include both physically realizable and hypothetical.

3) Metrics that can be used are included in 5.2.2.

Depending on the task addressed by the AI system (e.g. classification, interpolation/regression) different statistical metrics are possible. This subclause describes common statistical metrics available and the way to calculate them. The list is not exhaustive and some of these metrics are compatible with other tasks. It is possible to use them independently or in combination. Depending on the application, there are also numerous task-specific metrics (e.g. BLEU, TER or METEOR for machine translation, intersection over union for object detection in images, or mean average precision for ranked retrieval), but their description is out of the scope of this document.

5.2.2 Examples of performance measures for interpolation

5.2.2.1 Root mean square error or root mean square deviation

The root mean square error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are and RMSE is a measure of the spread of the residuals.

5.2.2.2 Max error

Max error is either an absolute or a relative metric calculating a value in the input data and the corresponding value in the prediction from the AI system. The absolute max error is the maximal signed difference between a value in the input data and the corresponding value in the prediction from the AI system. The relative max error is the percentage of the width of the variation domain on which the AI system operates.

5.2.2.3 Actual/predicted correlation

The actual/predicted correlation is the linear correlation (in the statistical sense) between the actual values and the predicted values for every value considered in a set.

5.2.3 Examples of performance measures for classification

5.2.3.1 General notions and associated basic metrics

A set of samples can have the following characteristics:

— total population: the total number of samples in the data;

— condition positive: the number of real positive cases in the data;

— condition negative: the number of real negative cases in the data;

— prediction positive: the number of samples classified as positive;

— prediction negative: the number of samples classified as negative;

— prevalence: the proportion of a particular class in the total number of samples.

Each instance in the set of samples is classified by the classification system in one of the following ways:

— true positive (hit): instance belongs to the class and is predicted as belonging to the class;

— true negative (correct rejection): instance does not belong to the class and is predicted as not belonging to the class;

— false positive (false alarm, Type I error): instance does not belong to the class and is predicted as belonging to the class;

— false negative (miss, Type II error): instance belongs to the class and is predicted as not belonging to the class.

Several metrics are built on top of these sample characteristics, as described in Table 1.

— True positive rate, sensitivity: the true positive rate (also known as sensitivity, recall or probability of detection) indicates the proportion of objects correctly classified as positive in the total number of positive objects.

— True negative rate, specificity: the true negative rate (also known as specificity or selectivity) indicates the proportion of objects correctly classified as negative in the total number of negative objects.

— False negative rate: the false negative rate (also known as miss rate) indicates the proportion of objects falsely classified as negative in the total number of positive objects.

— False positive rate: the false positive rate (also known as fall-out or probability of false alarm) indicates the proportion of objects falsely classified as positive that are negative. Thus, the probability of a false alarm is given.

— Accuracy: the accuracy indicates the proportion of all objects that are correctly classified.

— positive predictive value: the positive predictive value (also known as precision or relevance) indicates the proportion of results correctly classified as positive in the total of results classified as positive.

— Negative predictive value: the negative predictive value (also known as separation ability) indicates the proportion of results correctly classified as negative in the total number of results classified as negative.

— False discovery rate: the false discovery rate indicates the ratio of mistakenly rejected null hypotheses (false positives, false alarm, type I errors) to the total rejected null hypotheses (prediction positives).

— False omission rate: the false omission rate indicates the ratio of mistakenly rejected false negatives to the total number of predicted negatives.

— Positive likelihood relation: the positive likelihood relation indicates the ratio of the true positive rate to the false positive rate.

— Negative likelihood relation: the negative likelihood relation indicates the ratio of the false negative rate to the true negative rate.

— Diagnostic odds rate: indicates the ratio of the probability of true positives to the probability of false positives and is independent of prevalence.

— F1 score: the F1 score combines the true positive rate and the positive predictive value using the harmonic mean.

Table 1 — Sample characteristics and relevant basic metrics built upon them

<table>
<tr><td></td><td></td><td colspan="2">True condition</td><td colspan="2"></td></tr>
<tr><td></td><td>Total population</td><td>Condition positive
(CP)</td><td>Condition negative
(CN)</td><td>Prevalence
$\frac{N_{C+}}{P_{tot}}$</td><td>Accuracy
$\frac{N_{T+}+N_{T-}}{P_{tot}}$</td></tr>
<tr><td rowspan="2">Predicted condition</td><td>Prediction positive</td><td rowspan="2">True positive
(TP)</td><td>False positive
(FP)</td><td>Positive predictive value (V_{P+})
Precision, relevance
$\frac{N_{T+}}{N_{P+}}$</td><td>False discovery rate
$\frac{N_{F+}}{N_{P+}}$</td></tr>
<tr><td>Prediction negative</td><td>Type I error</td><td>False omission rate
$\frac{N_{F-}}{N_{P-}}$</td><td>Negative predictive value
Separation ability
$\frac{N_{T-}}{N_{P-}}$</td></tr>
<tr><td colspan="2"></td><td>False negative
(FN)
Type II error</td><td>True negative
(TN)</td><td></td><td></td></tr>
<tr><td colspan="2"></td><td>True positive rate (R_{T+})
Sensitivity, recall
Probability of detection
$\frac{N_{T+}}{N_{C+}}$</td><td>False positive rate (R_{F+})
Fall-out, probability false alarm
$\frac{N_{F+}}{N_{C-}}$</td><td>Positive likelihood ratio (R_{L+})
$\frac{R_{T+}}{R_{F+}}$</td><td rowspan="2">Diagnostic odds ratio
$\frac{R_{L+}}{R_{L-}}$</td><td rowspan="2">F_1 score
$\left(\frac{{R_{T+}}^{-1}+{V_{P+}}^{-1}}{2}\right)^{-1}$</td></tr>
<tr><td colspan="2"></td><td>False negative rate (R_{F-})
Miss rate
$\frac{N_{F-}}{N_{C+}}$</td><td>True negative rate (R_{T-})
Specificity, selectivity
$\frac{N_{T-}}{N_{C-}}$</td><td>Negative likelihood ratio (R_{L-})
$\frac{R_{F-}}{R_{T-}}$</td><td></td><td></td></tr>
</table>

where

N_{T+} is the number of true positives;

N_{T-} is the number of true negatives;

N_{F+} is the number of false positives;

N_{F-} is the number of false negatives;

N_{C+} is the number of conditions positive;

N_{C-} is the number of conditions negative;

P_{tot} is the total population;

N_{P+} is the number of predictions positive;

N_{P-} is the number of predictions negative;

R_{T+} is the true positive rate;

R_{T-} is the true negative rate;

R_{F+} is the false positive rate;

R_{F+} is the false negative rate;

R_{L+} is the likelihood positive ratio;

R_{L-} is the likelihood negative ratio;

V_{P+} is the positive predictive value.

Table 1 provides a synthetic view of the sample characteristics and metrics described in this subclause. All these sample characteristics and metrics apply primarily to binary classification but have also generalized definitions in the multiclass and multilabel cases.

5.2.3.2 Advanced metrics

5.2.3.2.1 Precision recall curve

Precision/recall pairs of metrics are computed at different output thresholds. Precision/recall pairs express trade-offs between precision and recall when these metrics are used to evaluate robustness.

5.2.3.2.2 Receiver operating characteristic (ROC)

The ROC curve is a plot of the true positive rate against the false positive rate at different settings of the hyperparameters (e.g. decision threshold).

ROC express trade-offs between true positive rates and false positive rates when these metrics are used to evaluate robustness. ROCs are used when one metric is associated with a significant cost or benefit in robustness evaluation, such as in the medical domain where false diagnoses can be especially problematic.

5.2.3.3 Lift

The lift metric is a measure comparing the relative performance of a prediction system against another control group (usually randomly selected).

5.2.3.4 Area under curve

Area under curve measures the integral of the ROC curve which represents the performance of a model for every threshold of classification. The ROC curve shows the true positive rate relative to the false positive rate.

5.2.3.5 Balanced accuracy

Balanced accuracy is the average recall obtained on each class as described in Reference [12].

5.2.3.6 Micro average and macro average

Measures like precision or recall computed over the whole dataset are sometimes misleading, in cases of unbalanced datasets. A possible strategy to alleviate this is to compute a macro-averaged measure, which is the average of the measure computed for each class separately, instead of the micro-average which is the standard computation without class separation[13].

5.2.3.7 Matthews correlation coefficient (MCC)

Matthews correlation coefficient is a measure on a set of classifications. Its range lies within [-1,1] in which +1 represents perfect prediction, -1 represents total inverse prediction and 0 represents average prediction. Crucially, this metric generalizes to instances where the classes are unbalanced in the data themselves (i.e. an MCC of 0 does not necessitate $1/N$ prediction accuracy, given N classes)[14],[15].

It is computed in Formula 1:

$$\frac{N_{T+} \times N_{T-} - N_{F+} \times N_{F-}}{\sqrt{(N_{T+} + N_{F+})(N_{T+} + N_{F-})(N_{T-} + N_{F+})(N_{T-} + N_{F-})}} \quad (1)$$

where

N_{T+} is the number of true positives;

N_{T-} is the number of false negatives;

N_{F+} is the number of false positives;

N_{F+} is the number of false negatives.

5.2.3.8 Confusion matrix and associated metrics

A confusion matrix allows a detailed analysis of the performance of a classifier and is helpful to circumvent or uncover the weaknesses of individual metrics as it achieves a more rigorous and well-rounded analysis of classifier performance. By contrast, using a single metric to express classifier performance is not informative enough to conduct this analysis, as it does not indicate which classes are best recognized or the type of errors committed by the classifier.

The confusion matrix C is a square matrix where entry $C_{r,c}$ at row r and column c are the number of instances belonging to the r^{th} class or category that are labelled by the classifier as having the c^{th} class.

Confusion matrices include counts of true positives, true negatives, false positives and false negatives: metrics such as accuracy, per-class recall, and per-class precision can be calculated from these. Further metrics can be derived from confusion matrix elements, such as entropy of the histogram represented by the matrix.

5.2.4 Other measures

5.2.4.1 Hinge loss

Hinge loss is an upper bound on the number of mistakes made by a classifier. In the general, multiclass case, the margin is computed by the Crammer-Singer method[16].

5.2.4.2 Cohen's kappa

Cohen's kappa is a measure of inter-annotator agreement [see Formula (2)].

$$\kappa = (p_o - p_e) / (1 - p_e) \quad (2)$$

where

p_o is the prior probability of agreement of the label on any sample in observed data;

p_e is the expected agreement when each of two annotators assign labels independently and according to their own measured prior distributions, given empirical data.

This measure is primarily used for evaluating data quality after (error-prone) human annotation, but it also has applications as a proxy evaluation method when labels are missing, by comparing two classifiers with each other.

5.3 Statistical methods to measure robustness of a neural network

5.3.1 General

When applying the metrics from 5.2 on testing data in order to assess robustness, several statistical techniques are available. This subclause describes some of the statistical methodologies available to perform steps 2 and 3 described in 4.1 to plan and conduct the testing. Performing a testing protocol is not unique to neural networks and considerations include the testing environment set-up, what and how to measure, and data sourcing and characteristics. The difference in neural network robustness test planning is a more "intense" consideration of the data sourcing (e.g. quality, granularity, train/test/validation datasets, etc.). While conducting the testing, planned data sourcing and availability of computational resources are important considerations due to the sometimes massive amounts of data and computational resources required by neural networks.

5.3.2 Contrastive measures

The statistical measures of performance are applied first on a reference dataset and then on one or several datasets representative of the targeted changes of circumstances. For each of those, if the performance drop from the reference test set is sufficiently low, then the system is deemed robust.

6 Formal methods

6.1 General

Another aspect of robustness is the degree to which changing circumstances affect the behaviour of the system, independently of its performance. Formal methods are especially appropriate to assess the stability of the system, i.e. the extent to which its outcome changes when input varies. Although a robust system can be unstable and a stable system can be not robust, stability is a strong indicator for robustness, as it makes the outcome more predictable.

Formal methods have been used to improve software reliability and allow stronger quality assurance on systems involving software. While it has mainly been used in the contexts of safety-critical applications, e.g. transportation systems, it is now being used more widely.

Formal methods allow a mathematical proof of a property all over a given domain, whereas statistical or empirical methodologies are based on extrapolation of the results from only the tested samples to the whole domain. Safety properties are usually a main focus, but the methods apply to a wide range of properties. Formal methods are usually complex to deploy, as they sometimes require specific mathematical modelling depending on the type of system to be analysed, and potentially a heavier instrumentation of the system.

AI software systems pose new specific challenges in terms of system validation, e.g. contrary to classical software, their behaviour is harder to explain and to prove. This is especially the case for neural networks because of the way their construction is done (through training instead of programming), and the inherent non-linearity of their behaviour. As a result, more adequate system properties are required, and new methodologies are considered to prove them efficiently.

This clause follows the general workflow described in Figure 1 to assess the robustness of a neural network. In particular, it focuses on Steps 1, 2 and 3 presented in 4.1. These steps consist of first choosing the robustness properties wanted, then preparing the data, then performing the tests. Depending on the kinds of use of the neural network, different properties are to be considered, different preparation steps are sometimes required, and several methods to conduct the testing are possible.

6.2 Robustness goal achievable using formal methods

6.2.1 General

This subclause describes the property of stability which is one of the properties used to assess robustness. It is formalized differently depending on the task fulfilled by the system.

1) For interpolation systems: interpolation stability calculates the uncertainty of a neural network, which is a way to detect when the neural network can have insufficient robustness.

2) For classification systems: maximum stable space calculates the size of the domain where the neural network has a stable classification performance.

6.2.2 Interpolation stability

Neural networks are, in some cases, used to replace complex mathematical computations that are costly to perform. For example, for a complex system of differential equations requiring an iterative method to be resolved (such as a Newton Raphson approach), the neural network is used as an "oracle" to infer directly some satisfying conditions of the systems. For this approach, a training dataset is generated using a mathematical model of the system, which can be expensive to compute. The question of the relevance and the coverage of the input space bears deep implication on the behaviour of the neural network. While it is obvious that the training dataset is not exhaustive (does not cover the entire input/output space), the problem remains the same for the testing dataset used to validate it. Also, as they are usually used to model functions operating in more than two dimensions, neural networks pose issues both in terms of behaviour and visualization.

For interpolation systems, the main advantage of neural networks is their ability to efficiently model complex linear and non-linear behaviour. However, by nature, some non-linear behaviours occur at some point, making it possible for the neural network to have some unexpected behaviour on parts of the domain in which it is to be used. One consequence is that their behaviour is erratic on some parts of the domain and not on others. As it is difficult to cover every part of the domain in order to find these erratic behaviours, they are likely to be missed and this can reduce robustness. This unintended behaviour constitutes an uncertainty on the quality of the interpolation capability of the neural network.

6.2.3 Maximum stable space for perturbation resistance

To mitigate the risks posed by data perturbations, which are also referred to as adversarial examples, it is possible to prove the robustness of the classifier to a certain point. The maximum stable space property is used for this purpose. The property states that there is no adversarial example around a specific input. The notion of distance is essential to define the set of inputs that are "around" a specific point. Annex A describes several types of data perturbations as well as associated distance metrics that are used for the assessment of neural networks.

This subclause describes three general approaches that are able to demonstrate a maximal stable space property, which is used to measure the robustness of a neural network performing classification. The property can be proven through the following techniques:

1) as a Boolean problem addressed using a solver;

2) as a numerical problem addressed using an optimization algorithm seeking a maximum; or

3) as a mathematical property approximated using abstract interpretation.

Each technique has been customized for the specific nature of neural networks.

6.3 Conduct the testing using formal methods

6.3.1 Using uncertainty analysis to prove interpolation stability

Uncertainty analysis is a technique commonly used to control the behaviour of mathematical functions. The goal is to identify upon which inputs the function is having sudden and significant variations. For neural networks, it is indeed a way to find the issues described in 6.2.2. In particular, it is possible to compare the measured behaviour with the knowledge of the actual behaviour. The goal is to measure the capacity of the neural network to model within an acceptable range of deviation the phenomenon it is intended to model. It especially helps measuring the variation of response of the network in order to check that no unstable behaviour have been introduced.

Foundational work has been done to formalize the stability of a neural network. For example, in Reference [17], a method is described to calculate the propagation of uncertainty in a network, thereby allowing it to detect the part of the domain where the response of the neural network is abnormal and non-robust behaviour is to be expected. In Reference [18], a method is described indicating the effect or importance of the input variables on the output, independently of the nature (quantitative or discrete) of the variables. In Reference [19], several sources of uncertainty are described, including uncertainty in the input, the sensitivity of the network itself and the influence of random choices for training and testing datasets. In this way, it is possible to detect the conditions under which the network is not robust and to detect the cause of uncertainty in the network response.

6.3.2 Using solver to prove a maximum stable space property

Neural networks are known to be large, non-linear, non-convex and beyond the reach of general-purpose tools such as linear programming solvers or existing satisfiability modulo theories (SMT). However, some advances have been made to use solver technologies to prove properties over neural networks. For example, Reference [20] presents an approach to efficiently prove properties over some classes of neural networks [using ReLu activation functions which is defined as $\mathrm{ReLu}(x) = \max(0, x)$] by using a variation of a Simplex algorithm. In Reference [21], an SMT solver is used to prove the absence or the existence of adversarial example, including the ability to exhibit it. In Reference [22], a combination of satisfiability solving and linear programming on a linear approximation of the overall network behaviour are considered. All of these works illustrate that it is possible to adapt generic solver technologies to prove properties on neural networks. Although these works aim to prove the robustness of classifiers, it is also possible to adapt these methods to handle other neural networks tasks.

6.3.3 Using optimization techniques to prove a maximum stable space property

Common optimization techniques are also able to verify a neural network, whereby any satisfiability problem is converted to an optimization problem. It then becomes possible to apply conventional optimization techniques, such as the Branch and Bound algorithm, to solve it.

A property is usually expressed as a Boolean formula on linear inequalities. For example, the output of a network always needs to be greater than some part of the input space. To prove it using an optimization technique, the property intended is expressed as additional layers at the end of the network. This way, once a solution of the optimization problem is found, the solution of the satisfiability problem is solved by checking the sign of the solution. In Reference [23], the construction of such an optimization problem from a satisfiability problem on a neural network is described, combining a classical gradient descent to find a local minimum, as well as a branch and bound optimizer to determine the global optimum.

One other interesting example of optimizing techniques to prove neural networks robustness concerns the use of constraint programming[23]. The neural network is approximated by first modelling it as a linear program (using network composed of piece-wise linear functions), then approximating the feasible states using only convex sets, and finally applying iteratively a constraint solver to prove the robustness property.

6.3.4 Using abstract interpretation to prove a maximum stable space property

Abstract interpretation is a form of formal method that relies on a theory which constructs controlled approximations (see Annex B for more information). It is often used to prove complex program properties[25]. Abstract interpretation has taken a larger role in the software verification and validation community, especially in the context of safety-critical software, such as embedded software for planes[26], cars[27], and spacecraft[28].

Recent work[29],[30],[31],[32] constructs new abstract domains specially tailored for the behaviour of neural networks. Indeed, the non-linear nature of neural networks tends to render some of the existing abstract domains ineffective, especially the ones that use the affine dynamics of a system to define the abstract domain. This is the case, for example, with a new zonotopic domain described in Reference [30] that captures the specific dynamics of ReLu activation functions commonly used in image processing neural networks.

To prove the stability of decision on some part of the input space, one needs to first express the part of the input space to be analysed using an abstract domain. Then, it needs to define an abstract semantic able to perform a symbolic computing of the neural network over the abstract domain. The output is then an abstract value representing an over-approximation of all the possible classification outputs on this part of the input space. The output is a vector whose elements are the confidence of classification for each class. Each element is itself an abstract value (for example an interval). Once the output is computed, two cases are possible: either one of the elements is greater than any other, or there is no class that always takes over the decision.

7 Empirical methods

7.1 General

The next aspect of robustness covered in this document is the subjective assessment of robustness. This assessment is based on a functional evaluation at the system level, for which empirical methods are the most suitable, as they directly collect human perceptions on this matter.

7.2 Field trials

While there are several aspects that need to be studied for the establishment of trust in AI systems, the number of feasible approaches for analysing a system's behaviour and performance are limited. AI systems typically consist of software to a large extent. Therefore, standards for software testing are needed, such as the ISO/IEC/IEEE 29119 series, for example.

The primary goals of software testing are stated in ISO/IEC/IEEE 29119-3:2013:

> "Provide information about the quality of the test item and any residual risk in relation to how much the test item has been tested; to find defects in the test item prior to its release for use; and to mitigate the risks to the stakeholders of poor product quality."

The workflow to assess neural network robustness described in Figure 1 contains three steps that are crucial for every field trial:

1) setting up the testing plan (plan testing);

2) realization of the data acquisition (data sourcing);

3) carrying out the trial in a real operation environment (conduct testing).

In contrast to other testing methods, in field testing the neural network is integrated into a system that operates in a realistic environment for the respective application. Therefore, data acquisition and sourcing are integral to experimental design and execution.

Defects and poor product quality are concerns when testing AI systems too. However, the failure of an AI system in a functional test is not necessarily related to a "software bug" or an erroneous design. AI systems showing occasional malfunctions are sometimes accepted because they are still regarded as useful for their intended purpose, in particular where there are no feasible alternatives. AI systems reveal their degree of performance mainly during field trials or deployment, as is the case with systems like virtual assistants, for example. This applies to many AI systems that operate in interaction with or dependency of natural environments and humans.

How to deal with the uncertainty of a product's efficacy and the risks of its deployment are subjects of many regulations in the medical domain. For instance, in Europe medical devices, including those using AI, need to comply with ISO 14155. They need to undergo "clinical investigations", a procedure that resembles "clinical trials"[34],[35],[36].

For non-medical devices using AI, field trials have for long been a recognized means of comparing and assessing the robustness of solutions. Some prominent examples are:

— facial recognition trials[37],[38],[39];

— tests of decision support systems for agricultural applications[40];

— practice for testing driverless cars[41];

— tests of speech and voice recognition systems[42],[43];

— networked robot at a train station[44].

Field trials for AI systems greatly vary with respect to methodology, number of users or use samples involved, status of the responsible organization/persons and documentation of the results.

7.3 A posteriori testing

In some cases, it is possible to formally validate the robustness of an intelligent system. When this is not possible, as is very often the case for neural networks[45], validation by testing empirically the robustness of the system is then implemented, and input-output evaluation are very popular in this context. In this kind of evaluation there are a priori testing and a posteriori testing methods. While a priori testing knows in advance the expected output and statistical measures are therefore applicable, a posteriori testing does not know it in advance. In that case, it is sometimes possible to design automated measures to still perform statistical measures by indirect means. Otherwise, the only available method is empirical, relying on the judgment of human subjects.

In a posteriori testing, Steps 4 and 5 of the process in Figure 1 are slightly altered. Step 4 is likely to be more complicated because the correct answer is unknown. Interpreting the results in Step 5 is a matter of consensus and not based on an unambiguous truth.

In general, to validate the robustness of a system, data or test environments representing a wide variety of test scenarios, for normal operating conditions and critical cases, are identified (Step 2 of the process). These inputs are submitted to the system to be evaluated and the system outputs (called hypotheses) are compared to references, i.e. to a ground truth (Step 3). These inputs are designed to challenge the system to check its robustness, for example by using adversarial examples. These references are usually provided by human annotators performing the same task as that performed by the evaluated system, or by physical measurements.

In the case of a priori testing, references are provided by human annotators, and they usually all agree among themselves on the "right answer" to be provided (high inter-annotator agreement rate). The ground truth is therefore unambiguous. On the contrary, with a posteriori testing, the references produced by the annotators varies from one to another. The field truth is therefore ambiguous. In general, this is the case when the task has several correct answers[46].

As it is not possible to determine a priori all the possible correct answers, a posteriori evaluations are carried out. That is, it is by looking at the outputs of the systems that human annotators or automated measures can tell whether they are "good" or "bad".

Machine translation is a classic example of a task for which a posteriori evaluation is a useful complement to a priori testing. There are usually several ways to translate the same sentence from one language to another. While statistical methods are often applied in this case by establishing an arbitrary set of correct or acceptable answers to compare the output with Reference [47], this is not a fully reliable measure of performance and subjective a posteriori testing is often more precise. Similarly, during a navigation task, it is possible to accept several trajectories to move from one place to another. Depending on the ability to define an objective criterion for successful trajectories, a posteriori testing can be applied with either statistical or empirical means.

It is also possible to use a posteriori evaluation to validate a new robustness metric (a new method or formula to measure). Indeed, when the quality of task is subjective the metrics needs to assign quality scores that correlate with human judgment of quality. Human judgment is the benchmark for assessing automatic metrics[48].

However, the concepts of a posteriori evaluation and post-deployment evaluation are overlapping in some cases, particularly when testing with end users. For example, in the case of the evaluation of the quality of a human–machine interaction, the evaluation is done a posteriori because it is not possible to know the how the interaction will generalize across the population before widespread interaction takes place. To carry such evaluation, it is possible to vary the user profile and having a user pool representative of the system's actual operating conditions and obtain an empirical analysis of the robustness of this interactive intelligent system.

7.4 Benchmarking of neural networks

Benchmarking a neural network-based system can help in establishing some degree of the robustness of the system. Often, the initial trust in an AI solution based on neural networks is established by a benchmark test. For example, in pattern recognition and similar applications of AI methods, benchmarking has for long been the gold standard for establishing trust in a certain method[49]. However, benchmarking can introduce elements of subjectivity, such as in the tagging or annotation of test datasets by expert practitioners.

Benchmarking measures the performance of a system on carefully designed datasets that are publicly available in most cases. Often, they are used for testing different systems competitively. Prominent examples of benchmarking are the face recognition vendor tests (FRVT) conducted by the US Department of Commerce[50]. Other examples are the "grand challenges in biomedical image analysis"[51].

In contrast to 7.2, benchmarking does not necessarily require an operational system in a real application scenario. For benchmarking, datasets are created that pose a serious challenge for state-of-the-art classification or regression methods. The benchmark datasets need to be complemented by a set of benchmarking rules that describe and standardize ways of setting up testing, documenting these setup, measuring results and documenting these results[52].

Benchmarking plays a strong role in pattern recognition research and has contributed decisively to the advancement of the field. However, benchmarking is usually not sufficient for a decision on a certain robustness goal. Benchmarking results should be interpreted with care[53].

Annex A
(informative)

Data perturbation

A.1 General

Formally, a perturbation is defined as a homomorphism (i.e. a function from one domain to the same one) of the domain of the inputs of the system. The domain of the inputs containing all RGB images of a certain width and height is an example of such domain. This annex describes data perturbations in the context of assessing the robustness of neural networks.

For example, automated classification systems are a prominent use of neural networks in the industry. Such classification systems are used for facial recognition, object tracking, sound recognition, etc. A usual way to construct a classification system based on neural networks is to perform a supervised training using a labelled dataset.

The work in Reference [54] has shown that even state-of-the-art neural networks for classification are strongly susceptible to data perturbations or adversarial examples. Adversarial examples include images or sound samples that are slightly modified from an existing one, resulting in a different classification result compared to the original. Adversarial examples occur naturally in the environment or because of sensor properties, and there are also many techniques to engineer such examples, but there is currently no tractable way to discover them all.

From the view point of AI software engineers, the existence of adversarial examples presents a risk to the robustness of the system, as the system has an erratic behaviour in some cases. Engineers know that adversarial examples exist. However, it is not easy to identify all possible adversarial examples in advance.

Using an adversarial example to cause an unintended behaviour of a neural network constitutes an attack. There are two main attack paradigms that are found in the literature:

— white-box attack, where the adversary has full knowledge of the neural network, training dataset and training algorithm;

— black-box attack, where the adversary lacks knowledge of the neural network, training dataset or training algorithm.

While this annex describes various types of perturbations to different types of data, it is not intended to be exhaustive. It is also important to note that hardware is able to cause unintended behaviour by altering the data through their numerical representation and, thus, causing perturbations. There are also many strategies and techniques proposed in the literature to defend and protect a system from these types of attacks.

In A.2 and A.3, examples of data perturbations are described for images and sounds. In each case, both unintentional natural perturbations and intentional attacks coexist in a wide range of applications.

A.2 Example image perturbations

A.2.1 General

Several types of image perturbations exist that can represent the possible degradation that the environment is able to inflict to the image processed by the AI system. An image is (usually) a 2-dimensional array of pixel, each one being represented by one or more numerical values (for example

one for a black and image, 3 for a RGB image). Without any loss of generality, we will consider in the following an image to be an array of width, W, and height, H, of pixels, with each pixel $p_{i,j}$ being between the value 0 and 255. Therefore, an image is a point in an $L \times W$ space. A perturbation of an image is a function that transform one image into another using a specific function.

When two images are in the same space, there are a variety of metrics available to calculate the distance between them, including mean-squared error, Levenstein distance[55], structural similarity index[56], etc. Each perturbation present is also applicable also to colour images by considering the perturbation on each channel. As there are many perturbations possible, there are also many metrics that define which ones are closer to the input. Attacks can be designed to mimic actual deterioration of the image acquisition process, for example noise, vibration, dazzling or occlusion of the camera.

In A.2.2 to A.2.7, some example image perturbations and some of the existing metrics derived from them are described.

A.2.2 Homogeneous noising

A homogenous noising is a transformation that puts a signed bounded random perturbation on every pixel of the image. The homogeneous noising is defined by a value K corresponding to the maximum of noise that it is possible to apply to each pixel.

Formally, each pixel is transformed using the following function:

$p_{i,j} = v \rightarrow p'_{i,j} = v \pm k$ with $k \leq K$ and $0 \leq v \pm k \leq 255$

In the case of a homogeneous noising, the metric used is directly correlated to the value K. An image I_1 at a distance of 1 of an image I_2 if:

— for each pixel $p^1_{i,j}$ of I_1 and its corresponding pixel $p^2_{i,j}$ of I_2, $\left|p^1_{i,j} - p^2_{i,j}\right| \leq k$; and

— at least one pixel exists where $\left|p^1_{i,j} - p^2_{i,j}\right| = k$.

A.2.3 Brightening

Brightening (or its inverse darkening) is one of the most basic image transformation available. It is used to create a change in the lighting of an image. Interestingly, Reference [57] demonstrates that lighting conditions is able to impact the performance of an image classifier. For that reason, it is crucial to demonstrate that a classifier is robust to such a perturbation.

Formally, each pixel is transformed using, for example, the following function:

$p_{i,j} = v \rightarrow p'_{i,j} = v + k$ with k constant and $-255 \leq k \leq 255$

Another way to adjust the brightness parameter for RGB image is to switch to HSL (hue/saturation/luminance) representation and adjust the luminance factor.

In the case of brightening, the metrics associated is either the luminance factor of the HSL representation or the constant used to do the perturbation for grayscale or RGB representation.

A.2.4 Vibration and rotation

When cameras are mounted on top of a vehicle some image degradation occurs due to a vibration or an oscillation of the device. These perturbations cause some inversion of pixels from one frame to another.

When considering the robustness against such rotation or vibration perturbations, it is important to consider the set of transformed images by such perturbations. For rotation, the metric used is the maximum angle of rotation that is being applied onto the image. A model of a rotated image[30] is achieved by first computing the real-valued position (i', j') that is to be mapped to the position of the centre of the pixel. Then, the intensity of the rotated pixel it deduced by performing a linear interpolation

taking into account surrounding pixels, such that the contribution of each pixel is proportional to the distance to (i', j'), cutting off contributions at distance 1.

It is possible to model the vibration of an image by using an adequate blurring kernel[58] that represents the degree of vibration applied on the image. This model allows the removal of vibration shake on an image by deblurring the image after discovering the appropriate blur kernel. Also, for vibration and flickering, it is possible to use a metric constructed[59] by using a Cox process. A Cox process serves as a model to aggregate spatial point patterns where those patterns follow a stochastic process.

A.2.5 Atmospheric turbulence

Optical degradation occurs sometimes due to the effect of turbulent flow of air, changes in temperature, density of air particles, humidity and carbon dioxide level. Considering all of these factors, the refraction of light changes when crossing through several layers of air with different properties. The result is a geometric distortion than degrades significantly the ability to process an image. This is especially the case when the image taken goes through a large number of atmospheric layers, for instance in the case of satellite imaging or high-altitude drone imaging. For example, in References [60] and [61], a framework to model atmospheric turbulence is described. A related problem is to improve the reconstruction of images to remove atmospheric turbulence[62],[63],[64]. It is important for these techniques to use metrics that model the intensity of the perturbation.

A.2.6 Blurring

In computer vision, blurring is used in order to smooth edges on an image. The most common blurring effect is done using Gaussian blur, but it is possible to define several other blurring effects using the following principle. A Gaussian blur typically relies on a convolution of the image with a kernel of Gaussian values. The size of the kernel is chosen in order to span a more or less blurring effect. The values of the kernel are defined by the standard deviation of the kernel, or are defined manually to apply non-Gaussian effect (for example a Box blur).

Formally a blurring is defined by a kernel K of size $n \times m$, usually $n = m = 3$. Each perturbed image is constructed by applying the kernel on each pixel of the original image and store the result of the convolution as the value of the new pixel. In Reference [65], the authors present some general blurring effects (motion, defocus and Gaussian) as well as a way to classify them automatically.

In the case of a Gaussian blur, the associated metric is the size of the kernel and the standard deviation of the kernel. Other blur effects introduce different metrics depending on their own parametrization. Each metric allows to set the intensity of the blur applied to the image.

A.2.7 Blooming

Blooming is identified as the formation of a bright spot that spreads around a local overexposure. The reason of the blooming artifact is that the single light sensitive elements of a CCD- or CMOS-sensor (pixel) is sometimes only able to absorb a limited amount of light. If the amount of charge of a single pixel is exceeded by a punctual overexposure, its excess charge is released to adjacent pixels. The charges flow primarily to the pixels that are coupled to each other for charge transport when the sensor is read out. As these pixels are most often coupled in the vertical direction, the artifact occurs often as a sharply defined strip that appears completely overexposed[66],[67].

Formally blooming is defined by a kernel K of size $n \times m$ around a local spot where $0 \leq n \leq L$ and $0 \leq m \leq W$ and depending on the direction of coupled pixels $m \gg n$ or $n \ll m$. Each pixel in K is transformed using the following function:

$$p_{i,j} = v \rightarrow p'_{i,j} = v = 255$$

Sensors with special anti-blooming structures are able to circumvent this artifact but have the disadvantage that due to the additionally required space for these structures the pixel size and with that also the sensitivity of the sensor decreases[68].

A.2.8 Smear

Smear is identified as the formation of bright lines in the image that are vertical for slow CCD cameras or have an angle to the vertical for CCD cameras with a higher speed. The reason for the smear artifact is the charge transport after the exposure that lead to the exposure of transporting pixels in case of strong incidence of light[69],[70]. In contrast to the mostly sharply defined Blooming strip, the smear effect strip always reaches the edge of the image and does not appear completely overexposed.

Formally vertical smear is defined by a strip S of size $L \times m$ with uniform intensity along each column starting at an overexposed spot. Each pixel in S is transformed using the following function:

$$p_{i,j} = v \rightarrow p'_{i,j} = v + b \text{ with } 0 \leq v + b \leq 255$$

Circumventing smear before artifacts that occur before a new exposure is done by resetting charge. But smear after artifacts that occur after an overexposure are only to be circumvented by the use of a mechanical or electro-optical shutter[70].

A.3 Example sound perturbations

A.3.1 Principle

Sound perturbations can impact a wide range of audio processing systems. A number of systems rely on voice input as the primary interface to control a device. For instance, virtual assistants with Automatic Speech Recognition (ASR) capabilities in the home and on mobile devices are able to control lighting, home appliances, home security systems, initiate purchasing and help to make phone calls or send messages. However, such systems can be sensitive to perturbations of the input sound data, potentially resulting in a failure to recognize the command, an unintended action, or even a malicious action if the perturbation is related to an attack.

There are two main types of sound perturbations: a) those that modify the acoustic signal in a human-audible range, and b) those relying on ultrasound. The following overview papers discuss a number of related techniques[71],[72] among these types of sound perturbations.

A.3.2 Sound attacks in human-audible frequency range

References [73] and [74] are among the first papers to consider automated attacks on the input speech signal. The perturbed sounds are intelligible as a specific command to an ASR system, but not understandable to a human, and in some cases, not even perceptible by a human. These methods essentially modify the acoustic features that many ASR systems rely on to recognize speech (e.g. Mel-frequency cepstral coefficients).

Reference [75] describes a sound perturbation based on genetic algorithm, and has been applied on a lightweight/simple ASR system. Reference [76] describes an improved version of this work based on a state-of-the-art DeepSpeech system. The SirenAttack described in Reference [77] is a broadly applicable attack based on particle swarm optimization.

Reference [78] developed a targeted speech adversarial example that makes slight perturbations to the original speech to fool the Mozilla DeepSpeech ASR system. The perturbations are determined based on a mathematical optimization technique and result in an ASR output that is recognized as some other pre-defined text. This is a white-box attack, i.e. this technique requires information about the system being attacked.

Reference [79] develops effectively imperceptible audio adversarial examples (verified through a human study) by leveraging the psychoacoustic principle of auditory masking, while retaining 100 % targeted success rate on arbitrary full-sentence targets. This work also makes progress towards physical-world over-the-air audio adversarial examples by constructing perturbations which remain effective even after applying realistic simulated environmental distortions. Reference [80] proposes a similar idea using psychoacoustic hiding to attack an ASR system. Both of these methods are white-box attacks.

Reference [81] demonstrates the existence of universal adversarial audio perturbations that cause mistranscription of audio signals by ASR systems. An algorithm is proposed to find a single quasi-imperceptible perturbation, which when added to any arbitrary speech signal, can most likely fool the target ASR model. Their method is applied to the Mozilla DeepSpeech ASR system. Additionally, it is shown that such perturbations generalize to a significant extent across models that are not available during training, by performing a transferability test on a WaveNetç based ASR system.

The approach in Reference [82] shows that is possible to do voice commands stealthily by embedding into songs which, when played, is able to effectively control the target system through ASR without being noticed.

A.3.3 Ultrasound based attacks

Ultrasound attacks mainly rely on the nonlinearity of the recording device, causing inaudible sound to be recorded. This was first accomplished in Reference [83], in which modulated ultrasound transmissions were transformed through microphone and amplifier non-linearities into valid commands which were executed by a variety of commercial ASR systems. Reference [84] also noted that non-linearities in loudspeakers make it harder for an adversary to increase the attack distance and used multiple loudspeakers in the form of an ultrasonic loudspeaker array to attack ASR systems at longer distances.

Reference [85] used inaudible ultrasound transmissions to record audible sounds, proposing a high-bandwidth covert channel in their BackDoor work. The work in Reference [86] improves upon the BackDoor method by not requiring any software at the microphone so the perturbation is able to be used on many deployed microphones or assistants.

Additionally, Reference [87] considers attacks on sound classification system, although only ultrasound-based ones so far.