भारतीय मानक

Indian Standard

IS/ISO/IEC TR 5469 : 2024

आर्टिफिशियल इंटेलिजेंस — कार्यात्मक सुरक्षा और एआई सिस्टम

Artificial Intelligence — Functional Safety and AI Systems

ICS 35.020

भारतीय मानक ब्यूरो
BUREAU OF INDIAN STANDARDS
मानक भवन, 9 बहादुर शाह ज़फर मार्ग, नई दिल्ली - 110002
MANAK BHAVAN, 9 BAHADUR SHAH ZAFAR MARG
NEW DELHI - 110002
www.bis.gov.in www.standardsbis.in

June 2024 **Price Group 16**

Artificial Intelligence Sectional Committee, LITD 30

NATIONAL FOREWORD

This Indian Standard which is identical to 'ISO/IEC TR 5469 : 2024 Artificial intelligence — Functional safety and AI systems' issued by the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) was adopted by the Bureau of Indian Standards (BIS) on the recommendations of the Artificial Intelligence Sectional Committee and approval of the Electronics and Information Technology Division Council.

The text of ISO/IEC standard has been approved as suitable for publication as an Indian Standard without deviations. Certain conventions and terminologies are, however, not identical to those used in Indian Standards. Attention is particularly drawn to the following:

a) Wherever the words 'International Standard' appear referring to this standard, they should be read as 'Indian Standard'; and

b) Comma (,) has been used as a decimal marker while in Indian Standards, the current practice is to use a point (.) as the decimal marker.

In this adopted standard, reference appears to certain International Standards for which Indian Standards also exist. The corresponding Indian Standards, which are to be substituted in their respective places, are listed below along with their degree of equivalence for the editions indicated. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies:

International Standards	*Corresponding Indian Standard*	*Degree of Equivalence*
ISO/IEC 22989 : 2022 Information technology — Artificial intelligence — Artificial intelligence concepts and terminology	IS/ISO/IEC 22989 : 2022 Information technology — Artificial intelligence — Artificial intelligence concepts and terminology	Identical

Contents

Page

Introduction

The use of artificial intelligence (AI) technology in industry has increased significantly in recent years and AI has been demonstrated to deliver benefits in certain applications. However, there is limited information on specification, design, and verification of functionally safe AI systems or on how to apply AI technology for functions that have safety-related effects. For functions realized with AI technology, such as machine learning (ML), it is difficult to explain why they behave in a particular manner and to guarantee their performance. Therefore, whenever AI technology is used in general and especially when it is used to realize safety-related systems, special considerations are likely to arise.

The availability of powerful computational and data storage technologies makes the prospect of large-scale deployment of ML possible. For more and more applications, adopting machine learning (as an AI technology) is enabling the rapid and successful development of functions that detect trends and patterns in data. This makes it possible to induce a function's behaviour from observation and to quickly extract the key parameters that determine its behaviour. Machine learning is also used to identify anomalous behaviour or to converge on an optimal solution within a specific environment. Successful ML applications are found in analysis of, for example, financial data, social networking applications and language recognition, image recognition (particularly face recognition), healthcare management and prognostics, digital assistants, manufacturing robotics, machine health monitoring and automated vehicles.

In addition to ML, other AI technologies are also gaining importance in engineering applications. Applied statistics, probability theory and estimation theory have, for example, enabled significant progress in the field of robotics and perception. As a result, AI technology and AI systems are starting to realize applications that affect safety.

Models play a central role in the implementation of AI technology. The properties of these models are used to demonstrate the compatibility of AI technology and AI systems with functional safety requirements. For instance, where there is an underlying known and understood scientific relationship between the key parameters that determine a function's behaviour, there is likely to be a strong correlation between the observed input data and the output data. This leads to a transparent and sufficiently complete model as the basis for AI technology. In this case, compatibility of the model with functional safety requirements is demonstrated. However, AI technology is often used in cases where physical phenomena are so complex or at such a small scale, or unobservable without influencing the experimental data, that consequently there is no scientific model of the underlying behaviour. In this case, the model of the AI technology is possibly neither transparent nor complete and the compatibility of the model with functional safety requirements is hard to demonstrate.

Machine learning is used to create models and thus to extend the understanding of the world. However, machine-learnt models are only as good as the information used to derive the model. If the training data does not cover important cases, then the derived models are incorrect. As more known instances are observed they are used to reinforce a model, but this biases the relative importance of observations, steering the function away from less frequent, but still real, behaviours. Continuous observation and reinforcement moves the model towards an optimum or it overemphasizes common data and overlook extreme, but critical, conditions.

In the case of continuous improvement of the model through the use of AI technology, the verification and validation activities in order to demonstrate its safety integrity are undermined as the function behaviour progressively moves away from the rigorously tested, ideally deterministic and repeatable behaviour.

The purpose of this document is to enable the developer of safety-related systems to appropriately apply AI technologies as part of safety functions by fostering awareness of the properties, functional safety risk factors, available functional safety methods and potential constraints of AI technologies. This document also provides information on the challenges and solution concepts related to the functional safety of AI systems.

Clause 5 provides an overview of functional safety and its relationship with AI technology and AI systems.

Clause 6 describes different classes of AI technology to show potential compliance with existing functional safety International Standards when AI technology forms part of a safety function. Clause 6 further introduces different usage levels of AI technology depending on their final impact on the system. Finally, Clause 6 also provides a qualitative overview of the relative levels of functional safety risk associated with different combinations of AI technology class and usage level.

Clause 7 describes, based on ISO/IEC 22989, a three-stage realization principle for usage of AI technology in safety-related systems, where compliance with existing functional safety International Standards cannot be shown directly.

Clause 8 discusses properties and related functional safety risk factors of AI systems and presents challenges that such use raises, as well as properties that are considered when attempting to treat or mitigate them.

Clauses 9, 10 and 11 show possible solutions to these challenges from the field of verification and validation, control and mitigation measures, processes, and methodologies.

The annexes provide examples of application of this document and additional details. Annex A addresses how IEC 61508-3 is applied to AI technology elements, and Annex B provides examples to how to apply three-stage realization principles and define various properties. Annex C describes more detailed processes related to 9.3. Annex D shows the mapping between safety life cycle in IEC 61508-3 and AI system life cycle in ISO/IEC 5338.

Indian Standard

ARTIFICIAL INTELLIGENCE — FUNCTIONAL SAFETY AND AI SYSTEMS

1 Scope

This document describes the properties, related risk factors, available methods and processes relating to:

- use of AI inside a safety related function to realize the functionality;
- use of non-AI safety related functions to ensure safety for an AI controlled equipment;
- use of AI systems to design and develop safety related functions.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 22989:2022, *Information technology — Artificial intelligence — Artificial intelligence concepts and terminology*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 22989:2022 and the following apply.

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

- ISO Online browsing platform: available at https://www.iso.org/obp
- IEC Electropedia: available at https://www.electropedia.org/

3.1
safety
freedom from *risk* (3.3) which is not tolerable

[SOURCE: IEC 61508-4:2010, 3.1.11]

3.2
functional safety
part of the overall *safety* (3.1) relating to the EUC (Equipment Under Control) and the EUC control system that depends on the correct functioning of the E/E/PE (Electrical/Electronic/Programmable Electronic) safety-related systems and other risk reduction measures

[SOURCE: IEC 61508-4:2010, 3.1.12]

3.3
risk
functional safety risk
<functional safety> combination of the probability of occurrence of *harm* (3.5) and the severity of that *harm* (3.5)

Note 1 to entry: For more discussion on this concept, see Annex A of IEC 61508-5.

[SOURCE: IEC 61508-4:2010, 3.1.6, modified — Added < functional safety > domain]

3.4
risk
organizational risk
<organizational> effect of uncertainty on objectives

Note 1 to entry: An effect is a deviation from the expected. It can be positive, negative or both and can address, create or result in opportunities and threats.

Note 2 to entry: Objectives can have different aspects and categories and can be applied at different levels.

Note 3 to entry: Risk is usually expressed in terms of risk sources, potential events, their consequences and their likelihood.

Note 4 to entry: This is the core definition of risk. As risks are specifically focused on *harm* (3.5) a discipline specific definition of *risk* (3.3) is used in this document in addition to the core risk definition.

[SOURCE: ISO 31000:2018, 3.1, modified — Added <organizational> domain and Note 4 to entry]

3.5
harm
physical injury or damage to the health of people, or damage to property or the environment

[SOURCE: IEC 61508-4:2010, 3.1.1]

3.6
hazard
potential source of *harm* (3.5)

[SOURCE: IEC 61508-4:2010, 3.1.2]

3.7
hazardous event
event that may result in *harm* (3.5)

[SOURCE: IEC 61508-4:2010, 3.1.4]

3.8
system
arrangement of parts or elements that together exhibit a stated behaviour or meaning that the individual constituents do not

[SOURCE: ISO/IEC/IEEE 15288:2023, 3.46, modified — Removed the three Notes to entry]

3.9
systematic failure
failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors

[SOURCE: IEC 61508-4:2010, 3.6.6]

3.10
safety-related system
designated system that both

- implements the required safety functions necessary to achieve or maintain a safe state for the EUC; and
- is intended to achieve, on its own or with other E/E/PE safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions

[SOURCE: IEC 61508-4:2010, 3.4.1]

3.11
safety function
function to be implemented by an E/E/PE safety-related system or other risk reduction measures, that is intended to achieve or maintain a safe state for the EUC, in respect of a specific *hazardous event* (3.7)

[SOURCE: IEC 61508-4:2010, 3.5.1]

3.12
equipment under control
EUC
equipment, machinery, apparatus or plant used for manufacturing, process, transportation, medical or other activities

Note 1 to entry: The EUC control system is separate and distinct from the EUC.

[SOURCE: IEC 61508-4:2010, 3.2.1]

3.13
programmable electronic
PE
based on computer technology which can be comprised of hardware, software and of input and/or output units

Note 1 to entry: This term covers microelectronic devices based on one or more central processing units (CPUs) together with associated memories, etc.

EXAMPLE The following are all programmable electronic devices:

- microprocessors;
- micro-controllers;
- programmable controllers;
- application specific integrated circuits (ASICs);
- programmable logic controllers (PLCs);
- other computer-based devices (e.g. smart sensors, transmitters, actuators).

[SOURCE: IEC 61508-4:2010, 3.2.12]

3.14
electrical/electronic/programmable electronic
E/E/PE
based on electrical (E) and/or electronic (E) and/or programmable electronic (PE) technology

Note 1 to entry: The term is intended to cover any and all devices or systems operating on electrical principles.

EXAMPLE Electrical/electronic/programmable electronic devices include:

- electro-mechanical devices (electrical);
- solid-state non-programmable electronic devices (electronic);
- electronic devices based on computer technology (programmable electronic).

[SOURCE: IEC 61508-4:2010, 3.2.13]

3.15
AI technology
technology used to implement an *AI model* (3.16)

3.16
AI model
physical, mathematical or otherwise logical representation of a system, entity, phenomenon, process or data

[SOURCE: ISO/IEC 22989:2022, 3.1.23, with the addition of AI]

3.17
test oracle
source of information for determining whether a test has passed or failed

[SOURCE: ISO/IEC/IEEE 29119-1:2022, 3.115]

4 Abbreviated terms

ALARP	as low as reasonably practicable
ANN	artificial neural network
CNN	convolutional neural network
CPU	central processing unit
CUDA	compute unified device architecture
DL	deep learning
DNN	deep neural network
GPU	graphics processing unit
EDDM	early drift detection method
E/E	electrical and/or electronic
E/E/PE	electrical/electronic/programmable electronic
EUC	equipment under control
FMEA	failure modes and effects analysis
GAMAB	globalement au moins aussi bon
HARA	hazard analysis and risk assessment
HAZOP	hazard and operability analysis
JPEG	joint photographic experts group
KPI	key performance indicator
MEM	minimum endogenous mortality
SVM	support vector machines

5 Overview of functional safety

5.1 General

The discipline of functional safety is focused on risks related to injury and damage to the health of people, or damage to the environment and, in some cases, mitigation against damage to product or equipment. The definition of risk differs based on the domain tags as shown in Clause 3. Both definitions are valid concepts for the use of AI. All references to risk in this document from this point on are related to the definition from the functional safety domain.

According to IEC 61508-1, control of risk is an iterative process of risk assessment and risk reduction. Risk assessment identifies sources of harm and evaluates the related risks for the intended use and the reasonably foreseeable misuse of the product or system. Risk reduction reduces risks until they become tolerable. Tolerable risk is a level of risk that is accepted in a given context based on the current state of the art.

The IEC 61508 series recognizes the following three-step (prioritised) approach as being good practice for risk reduction:

- Step 1: inherently functionally safe design;
- Step 2: guards and protective devices;
- Step 3: information for end users.

Risk reduction via the provision of functional safety is associated with Step 2.

This document focuses on the aspects of safety functions performed by a safety related system by making use of AI technology, either within the safety related system or during design and development of the safety related system (Step 2).

This document makes no provision of methodology for AI technology used for Steps 1 and 3.

5.2 Functional safety

IEC 61508-4[19] defines functional safety as that "part of the overall safety relating to the EUC (Equipment Under Control) and the EUC control system that depends on the correct functioning of the E/E/PE (Electrical/Electronic/Programmable Electronic) safety-related systems and other risk reduction measures." The E/E/PE safety-related system is delivering a "safety function", which is defined in IEC 61508-4 as a "function to be implemented by an E/E/PE safety-related system or other risk reduction measures, that is intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event." In other words, the safety functions control the risk associated with a hazard that leads to harm to people or the environment. The safety functions also reduce the risk of having serious economic implications.

As the term implies, functional safety - as defined in IEC 61508-4 - aims to achieve and maintain functionally safe system states of an EUC through the provision of safety functions. Based on the inclusion of "other risk reduction measures" in the definition of functional safety and safety functions, non-technical functions are explicitly included. The EUC is not limited to individual devices but it includes also systems.

Following these definitions, functional safety as a discipline is thus concerned with the proper engineering of these technical and non-technical safety functions for risk reduction or risk level containment of a particular equipment under control, from the component level up to the system level, including considering human factors, and under operational or environmental stress.

Functional safety focuses on safety functions for risk reduction and the properties of these functions required for risk reduction. While the functionality of a safety function is strongly use-case dependent, the properties required for risk reduction and the related measures are the main focus of functional safety standardization.

Prior to the advent of programmable systems, when safety functions were limited to implementation in hardware, the focus of functional safety was to reduce the consequences and the likelihood of random hardware failures. With software being increasingly used to implement safety functions, the focus shifted towards systematic failures introduced during design and development.

NOTE ISO 21448:2022[7] includes requirements on safety of the intended functionality including aspects such as performance limitation. Annex D describes implications for machine learning.

There is a robust body of knowledge on how to avoid systematic failures in non-AI systems and in software development.[138] This document considers the use of AI technology in the context of safety functions. Functions containing AI technology, especially machine learning, typically follow a different development paradigm to that of non-AI systems. They are less specification-driven and more driven by observation of the data defining the system behaviour. For this reason, the catalogue of available measures for dealing with systematic failures is extended with respect to the specificities of AI technologies: Annex A provides an example of that extension. AI-specific risk reduction measures also differ from non-AI systems from a functional perspective. Functional safety puts a focus on systematic capabilities (IEC 61508-4:2010, 3.5.9) in addition to random hardware and systematic failures throughout the life cycle.

The relevance of AI technologies when used to realize a safety function is their potential to address new methods for risk reduction. This document examines the use of such technologies for this purpose, while maintaining existing risk reduction concepts, by introducing risk and classification considerations.

In general, achieving an acceptable risk level for increasingly complex and automated systems is likely to depend on advanced safety concepts. This includes the adequate implementation of both technical and non-technical risk reduction measures to achieve and maintain safe system states. Assuring the validity of such advanced safety concepts is a great challenge in functional safety. It also leads to an increase in the number of functional safety requirements. For all technical risk reduction measures, the distinction is made that hardware random faults and systematic faults are considered, which is done in basic International Standards like the IEC 61508 series or derived International Standards. However, for safety functions including AI technology, it is inevitable that there is additional focus on the assurance that systematic capabilities of systems that implement these functions are sufficient for the intended use environment.

6 Use of AI technology in E/E/PE safety-related systems

6.1 Problem description

The use of AI technology and AI systems is currently not treated in mature functional safety International Standards (indeed, in some International Standards their use is explicitly forbidden). International Standards that include AI-related contents include ISO 21448[7].

6.2 AI technology in E/E/PE safety-related systems

E/E/PE safety-related systems have a set of properties to ensure that they provide the intended safety mitigation measures in a dependable way. These properties are ideally generic and application independent. However, the data and the specifications vary based on application and technology domain. The process in which properties are selected is described in Figure 3 for each of the three stages of the three-stage realization principle. The properties are selected on a case-by-case basis, as relevant to a particular application or technology domain, their data and specifications. Properties are based on existing International Standards, including the IEC 61508 series[16]-[19], the ISO 26262 series[12]-[15], IEC 62061[21] and the ISO 13849 series.[5],[8] Clause 8 provides a list of typical properties.

Satisfying the selected properties is likely to place particular functional safety requirements on the realization, installation, validation, operation and maintenance of such systems. For example, IEC 61508-3[18] defines such requirements for the software part of E/E/PE systems. However, several AI technologies use different development approaches (e.g. learning-based) compared to the non-AI software engineering life cycles targeted by IEC 61508-3.

To address the difference between traditional development processes and the approach typical of AI technologies, this clause provides a general classification scheme for the applicability of AI technology in E/E/PE safety-related systems, based on various contexts of the application of AI technology.

An example of a classification scheme is, summarized in Table 1 and the related flowchart represented in Figure 1. The scheme is intended to provide insight on how an AI technology is addressed in the context of functional safety for a specific application.

The classification scheme (see Table 1) is organized along two axes:

- AI Application and Usage Level. This axis considers the application of the AI technology and includes, among other things, the way in which it is used. It is classified from A to D, with two intermediate levels for A and B.

  NOTE 1 The factors identified in Clause 8 are of high relevance in the context of the classification. These factors are described further in Clause 8 and include the level of automation and control (see 8.2), the degree of decision transparency and explainability (see 8.3), the complexity of the environment and vague specifications (see 8.4), security (see 8.5), system hardware issues see (8.6) and the maturity of the technology (see 8.7).

  An example of a classification of Usage Level is as follows:

- Usage Level A1 is assigned when the AI technology is used in a safety-relevant E/E/PE system and where automated decision-making of the system function using AI technology is possible;

- Usage Level A2 is assigned when the AI technology is used in a safety-relevant E/E/PE system and where no automated decision-making of the system function using AI technology is possible (e.g. AI technology is used for diagnostic functionality within the E/E/PE system);

  NOTE 2 The evaluation can change depending on the role of the diagnostic function, such as whether the diagnostic is critical to maintaining the functional safety of the system or is merely a minor contributor to functional safety amongst many others.

- Usage Level B1 is assigned when the AI technology is used only during the development of the safety-relevant E/E/PE system (e.g. an offline support tool) and where automated decision-making of the function developed using AI technology is possible;

- Usage Level B2 is assigned when the AI technology is used only during the development of the safety-relevant E/E/PE system (e.g. an offline support tool) and where no automated decision-making of the function is possible:

- Usage Level C is assigned when the AI technology is not part of a functional safety function in the E/E/PE system, but can have an indirect impact on the function:

  NOTE 3 The Usage Level C includes AI techniques clearly providing additional risk reduction and whose failure is not critical to the level of acceptable risk.

  EXAMPLE An AI technique that increases or decreases the demand rate placed on a safety system.

- Usage Level D is assigned if the AI technology is not part of a safety function in the E/E/PE system and has no impact on the safety function due to sufficient segregation and behaviour control.

  NOTE 4 An example is separation through a “sandbox” or ”hypervisor” in such a way that it does not affect the safety functionality.

- AI Technology Class. This axis considers the level of fulfilment of AI technology in satisfying the identified set of properties, in which:

  - Class I is assigned if AI technology is developed and reviewed using existing functional safety International Standards, for example, if the properties and the set of methods and techniques leading to achievement of the properties are identified using existing functional safety International Standards;

  - Class II is assigned if AI technology cannot be fully developed and reviewed using existing functional safety International Standards, but it is still possible (as shown in Figure 4) to identify additional complementary requirements, methods and techniques for development, design, verification and validation of the desired safety properties to achieve the necessary risk reduction;

  - Class III is assigned if AI technology cannot be developed and reviewed using existing functional safety International Standards and it is also not possible to satisfy the set of identified properties with related methods and techniques.

Components containing AI technology are composed of various technology elements (see 7.1). Each element belongs to a different AI technology class. For example, the lower levels of abstraction of a neural network are implemented using C++ libraries, which by themselves are systematically reviewed (e.g. according to the requirements in IEC 61508-3[18], see the example in Annex A). As such, they are classified as Class I, though the higher level of abstractions (e.g. deep learning models) are more likely classified as Class II or Class III.

For AI Technology Class I components, application of existing functional safety International Standards is possible and desired in general. For AI Technology Class II components, use of existing functional safety International Standards leads to partial achievement of the properties required for functional safety. Then the residual part is addressed using a set of complementary methods and techniques such as additional verification and validation (as detailed in Clause 9). Effectiveness of complementary methods and techniques are different between applications and Usage Levels. Components in AI Technology Class III, at the time of publication of this document, lack known methods and techniques to identify a set of properties to achieve sufficiently reduction of risk.

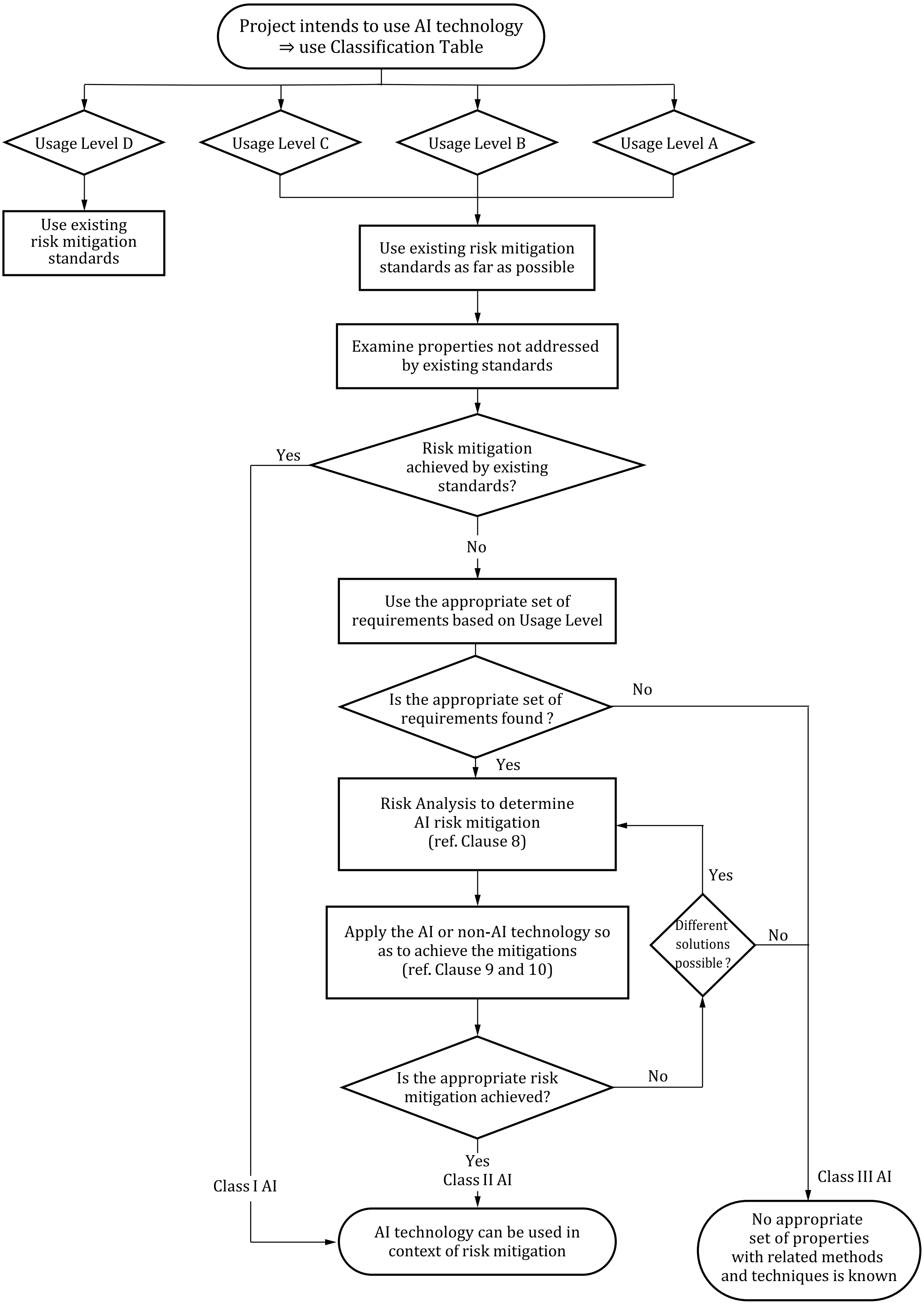
Table 1 — Example of AI classification table

AI Technology Class = > AI application and usage level	**AI technology Class I**	**AI technology Class II**	**AI technology Class III**
Usage Level A1 [a]	Application of risk reduction concepts of existing functional safety International Standards possible	Appropriate set of requirements [c]	At the time of publication of this document no appropriate set of properties with related methods and techniques is known to achieve sufficiently reduction of risk
Usage Level A2 [a]		Appropriate set of requirements [c]	
Usage Level B1 [a]		Appropriate set of requirements [c]	
Usage Level B2 [a]		Appropriate set of requirements [c]	
Usage Level C [a]		Appropriate set of requirements [c]	
Usage Level D [b]	Application of risk reduction concepts of existing functional safety International Standards		

[a] Static (offline) (during development) teaching or learning only.

[b] Dynamic (online) teaching or learning possible.

[c] The appropriate set of requirements for each usage level is established by application of risk reduction concepts of existing functional safety International Standards and additional considerations based on the literature review performed in Clauses 8, 9, 10 and 11. Examples are provided in Annex B.



Project intends to use AI technology
⇒ use Classification Table
Usage Level D
Usage Level C
Usage Level B
Usage Level A
Use existing risk mitigation standards
Use existing risk mitigation standards as far as possible
Examine properties not addressed by existing standards
Risk mitigation achieved by existing standards?
Yes
No
Use the appropriate set of requirements based on Usage Level
Is the appropriate set of requirements found ?
No
Yes
Risk Analysis to determine AI risk mitigation (ref. Clause 8)
Yes
Different solutions possible ?
No
Apply the AI or non-AI technology so as to achieve the mitigations (ref. Clause 9 and 10)
Is the appropriate risk mitigation achieved?
No
Yes
Class II AI
Class I AI
Class III AI
AI technology can be used in context of risk mitigation
No appropriate set of properties with related methods and techniques is known

Figure 1 — Flowchart for determining classification of AI technology

7 AI technology elements and the three-stage realization principle

7.1 Technology elements for AI model creation and execution

The creation and the execution of an AI model involves different AI technology elements. Table 2 provides a high-level description of the AI landscape and the typical technology elements involved, based on the functional layers of an AI ecosystem as described in ISO/IEC 22989:2022, Figure 6. Table 3 is an example of those technology elements for the specific case of machine learning.

Table 2 — AI technology elements (from ISO/IEC 22989:2022)

AI technology element
AI services
Machine learning — Model development and use — Tools — Data for machine learning
Engineering — Knowledge based on domain experience — Tools
Cloud and edge computing and big data and data sources
Resource pool-compute, storage, network
Resource management-resource provisioning

Table 3 — Example technology elements involved in model creation and execution for ML

Technology element (machine learning example)	Examples (not exhaustive)[b]
Application graph	Graph eXchange Format (GXF) graph in YAML Ain't Markup Language (YAML), recently qualified teacher (rqt) graph in Robot Operating System (ROS^{TM}/ROS2^{TM})
Machine learning framework[a]	TensorFlow®, PyTorch®, Keras, mxnet, Microsoft Cognitive Toolkit (CNTK), Caffe, Theano
Machine learning model language	Open Neural Network Exchange (ONNX®), Neural Network Exchange Format (NNEF^{TM}) PolyML
Machine learning graph compiler	TensorRT^{TM}, Glow, Multi-Level Intermediate Representation (MLIR), nGraph, Tensor Virtual Machine (TVM), PlaidML, Accelerated Linear Algebra (XLA)
Machine code compiler	Gcc, nvcc, clang/llvm, SYCL, dpc++, OpenCL^{TM}, openVX^{TM}

NOTE This table does not distinguish between elements used for training and those used for inference.

[a] A machine learning framework is an end-to-end machine learning platform including tools, libraries and community resources.

[b] The trade name(s) or trademark(s) in this table are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of these products.

Table 3 *(continued)*

Technology element (machine learning example)	Examples (not exhaustive)[b]
Libraries	CUDA® C++, CUTLASS, NumPy, SciPy, Pandas, Matplotlib, CUDA Deep Neural Network (cuDNN™), SYCL DNN, oneAPI Deep Neural Network (OneDNN), Math Kernel Library (MKL)
Executable machine code	executable machine code compiled for the following architectures: aarch64, PTX, RISC-V®, AMD64, x86/64, PowerPC™
Computational hardware	GPU, CPU, FPGA, ASIC, accelerators

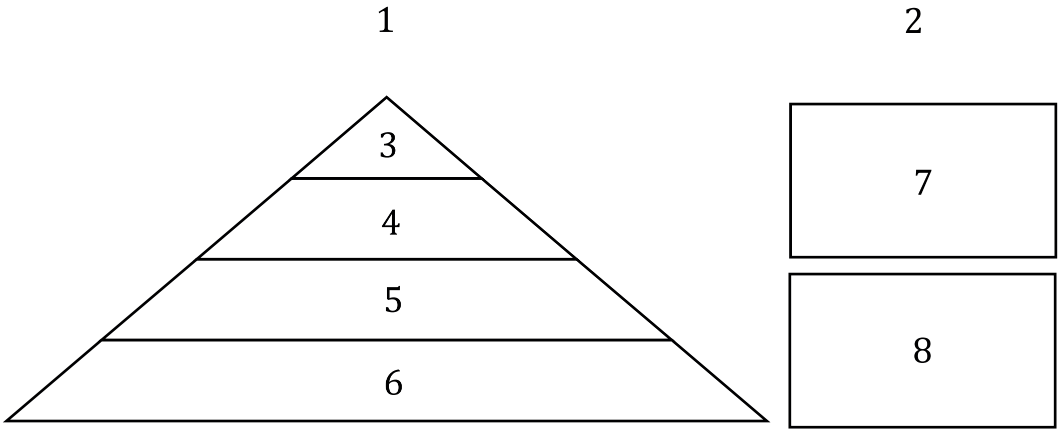
NOTE This table does not distinguish between elements used for training and those used for inference.

[a] A machine learning framework is an end-to-end machine learning platform including tools, libraries and community resources.

[b] The trade name(s) or trademark(s) in this table are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of these products.

Some technology elements are addressed with existing concepts of functional safety (e.g. the software translating the model to an executable code). However, all technology elements involved in the model creation and execution are part of the safety considerations, including those that are handled with existing concepts of functional safety and those for which new concepts are defined. Annex A includes an example of how existing concepts of functional safety are applied to AI technology via assessment of the applicability of IEC 61508-3[18]. Annex B includes an example of how specific properties, such as the ones described in Clause 8, are applied to AI technology for which existing concepts of functional safety cannot be applied.

As shown in Figure 2, elements containing AI technology are used at different levels of a system or application: for the higher-level elements (e.g. application graph or ML model, and related tools) specific properties, such as the ones described in Clause 8, are applicable; the lower-level elements (e.g. executable machine code, and related tools) are handled with non-AI properties as described in this clause, such as the properties defined in the IEC 61508 series[16]-[19], the ISO 26262 series[12]-[15] and other International Standards.



Key

1 code/models
2 tools/libraries
3 application graph
4 ML model
5 executable machine code
6 computational hardware
7 ML framework, code/graph/ML compilers
8 libraries

Figure 2 — Hierarchy of technology elements (ML example)

7.2 The three-stage realization principle of an AI system

Summarizing ISO/IEC 22989:2022, Figure 5, an AI system is represented (see Figure 3) by a realization principle comprising three main stages:

- data acquisition;
- knowledge induction from data and human knowledge;
- processing and generation of outputs.

NOTE 1 With respect to ISO/IEC 22989:2022 Figure 5, the first stage is mapped to the inputs task, the second stage is mapped to learning task and the third stage to the processing task.

NOTE 2 In this context, human knowledge is derived from a range of different expertise, both in the relevant domain as well as in AI systems.

NOTE 3 The described realization principle is general. Specific more detailed examples for AI system are the Monitor-Analyse-Plan-Execute (MAPE) or Sense-Understand-Decide-Act (SUDA).

NOTE 4 The intent of the three-stage realization principle is not to describe a life cycle (that is described in Clause 11 and includes all stages from concept development and maturation through to development of requirements) but mainly to show that AI includes another point of view (the data).

NOTE 5 Figure 3 does not show feedback loops that apply for AI systems that are tightly bound into decision loops or that change the real-world situation.

NOTE 6 Knowledge induction includes training, while processing includes inference.

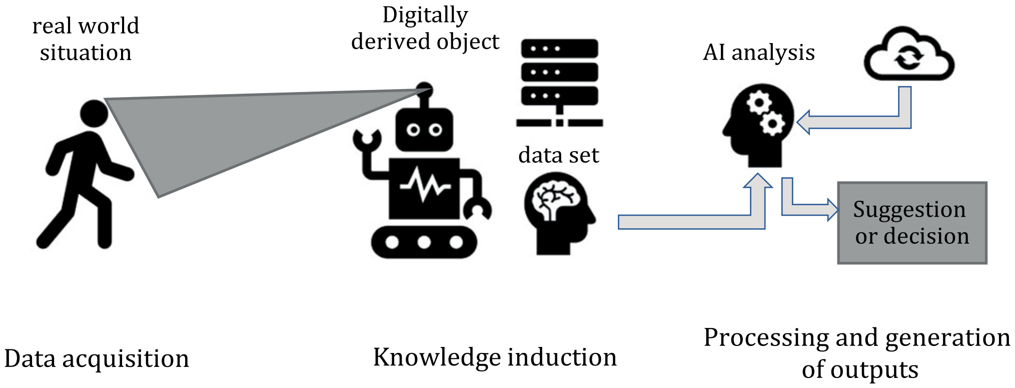


Figure 3 — Three-stage realization principle

7.3 Deriving acceptance criteria for the three-stage of the realization principle

Typically (see for example in ISO/IEC 22989:2022), a three-stage realization principle (see Figure 4) is used to derive acceptance criteria:

- Desirable properties are defined for each of the three stages.
- The properties are related to topics and eventually to detailed methods and techniques addressing those topics.
- Acceptance criteria are identified from the set of the detailed methods and techniques. As shown in Figure 4, the criteria for accepting selected methods and techniques and possibly metrics or thresholds like limits for estimated uncertainties are embedded in an overall acceptance argument. The overall acceptance argument demonstrates that the selected acceptance criteria harmonize with technology-independent risk acceptance criteria such as ALARP, MEM, GAMAB, or PRB. The overall argument is, for example, represented in an assurance case.

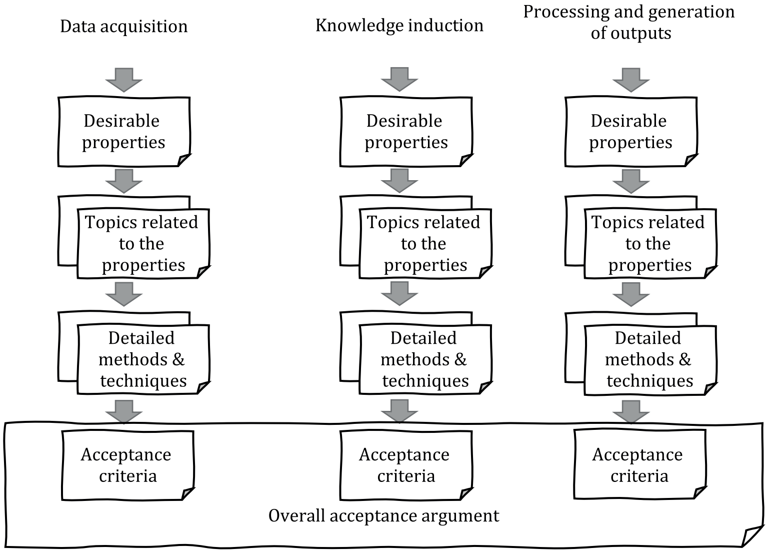

Data acquisition
Knowledge induction
Processing and generation of outputs
Desirable properties
Topics related to the properties
Detailed methods & techniques
Acceptance criteria
Overall acceptance argument

Figure 4 — Processes in each stage

NOTE 1 The properties are defined on a case-by-case basis or derived from properties listed in existing International Standards, based on the level of the elements containing AI technology. Refer to Clause 8 for the list of considered properties.

NOTE 2 The "Desirable properties", "Topics related to the properties" and "Detailed methods & techniques" are specific to each stage or common, depending on the specific application.

NOTE 3 In this context, the acceptance criteria are intended as something that is identified and confirmed during development.

8 Properties and related risk factors of AI systems

8.1 Overview

8.1.1 General

Clause 7 describes how the definition of desirable properties is the first step of the three-stage realization principle. The properties are related to topics and eventually to detailed methods and techniques addressing those topics. Acceptance criteria are then identified from the set of the detailed methods and techniques.

This clause provides a literature review on the properties that characterize systems using AI technology and their related risk factors. Such properties and risk factors include degree of automation and control (see 8.2), degree of decision transparency and explainability (see 8.3), environmental complexity and vagueness of their defining specifications (see 8.4), resilience to adversarial inputs (see 8.5), system hardware considerations (see 8.6) and technological maturity (see 8.7).

Details of the properties and risk factors of systems using AI technology, and their related topics and challenges, are discussed in this clause.

8.1.2 Algorithms and models

On a technological level, the capability of AI is often achieved by the combination of an AI model and parameters of the model. The parameters, which are generated by an AI algorithm during training, typically represents information that achieves the application's purpose, (e.g. knowledge about how various inputs are distinguished and recognized), while the model computes information from parameters and inputs, (e.g. to make a prediction). This means the functional safety of applications using AI technology depends on both.

Example types of AI models include linear functions, logical calculi, dynamic Bayesian networks and artificial neural networks (ANN). The parameters of the models are either handcrafted by an engineer, or are synthesized from data by machine learning algorithms that themselves use a systematic analysis process. The models are usually implemented as an executable representation, such as machine code (in the case of software), or special hardware, such as field programmable gate arrays (FPGAs).

Usually, models itself without parameters contain only a limited amount of knowledge or implications about the application's goals. This is quite similar to the role of foundational software libraries or programming environments (e.g. compilers) in non-AI software. As described by the IEC 61508 series[16]-[19], achieving functional safety includes demonstrating the correct functioning of the model. In this way, the integrity of model implementations in AI technology is often handled with existing principles of functional safety as specified in the IEC 61508 series[16]-[19], similar to that of non-AI software components. The same holds for the logic involved in the translation of the models and the parameters.

By contrast, model parameters often contain knowledge related to the objective of the systems involving functional safety. There are several different ways of constructing parameters and different approaches are used for assessing the completion of risk reduction measures to ensure functional safety.

For example, when model parameters are created manually by engineers, the models likely reflect the engineers' knowledge about the application, which is assessed during the management processes used within functional safety life cycles. In these cases, the life cycle of existing functional safety International Standards is followed (AI technology Class I as described in 6.2). It is often feasible to create parameters manually for simple models such as simple linear functions or logical calculi.

In some cases, parameters derived from data by machine learning algorithms are analysed and verified after their creation. Alternatively, parameters derived by machine learning algorithms are analysed, the underlying parameters extracted and used to extend general engineering knowledge, that, in turn, are used to develop further AI technologies. With the application of validated engineering knowledge, the life cycle of existing functional safety International Standards are applied (e.g. treating these models as AI technology Class I as described in 6.2).

In other cases, model parameters derived from data by machine learning algorithms are too complex to be understood, analysed and verified. This is often the case for complex types of models, such as neural networks, because representations of models in these types do not necessarily reflect human understanding or reasoning. The use of different approaches for assessing the risk reduction for functional safety is appropriate in these cases, which constitutes a major challenge for the use of AI technology in implementation of functional safety systems.

8.2 Level of automation and control

The level of automation (sometimes called "levels of autonomy" in the literature) describes the extent to which an AI system functions independently of human supervision and control. It thus determines not only how much information about the behaviour of the system is available to the operator, but also defines the control and intervention options of the human.

Dimensions of this topic include how high the level of automation is for the respective application, as well as the extent to which the user's control options are restricted. Systems using AI technology with a high-level of automation can exhibit unexpected behaviour that can be difficult to detect and control. Highly automated systems can thus pose risks in terms of their performance characteristics such as reliability and safety.

Several aspects are relevant in considering whether functional safety is achieved, such as the responsiveness of the AI system and the presence or absence of a ”supervisor” (as discussed in 10.2.3). In this context, a ”supervisor” serves to validate or approve automated decisions of the system. Such a ”supervisor” is achieved by technical control functions, though in some situations such a supervisory function is not feasible, e.g. highly complex decisions or ML systems that have learned new behaviours. For example, a second safety instrumented system for critical controls (Usage Level C or D as described in 6.2) is added and assigned to a safety function for redundant components, as in functional safety International Standards like IEC 61508-1[16]. See 10.2.3 for further details.

Another way for adding a ”supervisor” is to use a human whose task it is to intervene in critical situations or to acknowledge system decisions. One known way this is addressed is by adding a system (at Usage Level C or D as described in 6.2) to aid the human supervisor by detecting possible outcomes of the decision. An example is a simulation system that gives “what if” information for different decisions and checks for outcomes. However, even if humans are in the loop and oversee the actions of a system, sometimes this does not reduce such risks to a suitable level and can introduce additional risks.

EXAMPLE A human driver is not aware of a “black ice” road condition and takes control of a vehicle because they don’t understand why the autonomous system is driving so carefully.

Furthermore, the adaptability of the AI system is a consideration, and in particular whether a system changes its own behaviour over time, as is the case in some machine learning systems. These systems adapt to changing environmental conditions (e.g. via feedback loops or an evaluation function) and even acquire entirely new functions over time. A disadvantage of such learning systems, however, is that they can deviate from the initial specification and are difficult to validate. Therefore, such systems are considered very carefully before being used in systems of higher usage levels A-C as described in 6.2.

Table 4 (derived from ISO/IEC 22989:2022, Table 1) introduces several terminology and concepts related to the level of automation.

Table 4 — Relationship among autonomy, heteronomy and automation (derived from ISO/IEC 22989:2022, Table 1)

		Level of automation	**Comments**
Automated system	Autonomous	Autonomy	The system is capable of modifying its operating domain or its goals without external intervention, control or oversight.
	Heteronomous	Full automation	The system is capable of performing its entire mission without external intervention.
		High automation	The system performs parts of its mission without external intervention.
		Conditional automation	Sustained and specific performance by a system, with an external agent being ready to take over when necessary.
		Partial automation	Some sub-functions of the system are fully automated while the system remains under the control of an external agent.
		Assistance	The system assists an operator.
		No automation	The operator fully controls the system.

NOTE The dividing into levels applies to the control automation functions in any implementation of an automated AI system and taking into account the functions of the components of this system (for example, onboard equipment, floor equipment and control room equipment in railway systems).

8.3 Degree of transparency and explainability

Often, aspects of transparency and explainability are summarized under the term “transparency”. Literature described in the following clearly distinguish these terms.

ISO/IEC 22989:2022, 5.15.6 defines explainability as the property of an AI system to express important factors influencing the results of the AI system in a way that is understandable to humans, whereas transparency is defined as the property of a system that appropriate information about the internal processes of an AI system is made available to relevant stakeholders-see also ISO/IEC TR 24028[11].

Information about the model underlying the decision-making process is likely to be relevant. Systems with a low degree of transparency can pose risks in terms of their fairness, security and accountability. Furthermore, such systems can complicate the assessment of their quality. On the other hand, a high degree of transparency can lead to confusion due to information overload, or can conflict with privacy, security, confidentiality requirements and intellectual properties. According to Reference [11], a desirable level of explainability is often achieved without a high-level of transparency. Finding an appropriate level of transparency provides developers with opportunities for error identification and correction, as well as ensures that a user can trust the AI system.

In non-AI software, the intention and knowledge of the engineer is generally encoded into the system using a logical process, making it possible to trace through the code to determine how and why a certain decision has been made by the software. This is done by backtracking through and debugging the software or by reverse engineering the software. By contrast, decisions made by AI models, especially by models of high complexity, or models derived from machine learning algorithms, are more difficult to understand for humans. The way knowledge is encoded in the structure of the model and the way decisions are made, are often different from how humans reason about their own decision-making processes[130],[131].

A high-level of explainability protects against unpredictable behaviour of the system but is sometimes accompanied by lower overall performance in terms of the quality of decisions, due to the limitation of current explainability technology (which limits the amount of information contained in the model to create explanations of reasonable size). Here, a trade-off is often made between explainability and the performance of a system. In addition, the relevance of the information about an AI system's decision-making process is likely an important factor. It is possible that a system provides clear and coherent information about its decision-making process, but that this information is inaccurate or incomplete.

Explainable AI is also used to assist with post-incident analysis when the input data, which are sometimes transient, are recorded and reproduced.

Consequently, transparency and explainability are included in the general evaluation of the AI system. Considerations include:

- whether sufficient information about the system is available;
- whether it is understandable or at least delivers comprehensible information (possibly indirectly) to the intended recipient (the intended recipient of an explanation varies depending on the context. It is, for example, the system developer, first responders of the system in use, or bystanders in some cases);
- whether it delivers correct, complete and reproducible results consistently;
- whether the deliverable is adequate enough to the required confidence.

Several evaluation concepts and strategies exist to judge the transparency and explainability of an AI system, such as those reported in Reference [2] and in References [39] and [40]. Additionally, empirical assessments of the decision-making process of complex models are carried out, for example, by inspecting a convolutional neural network through visualization of components of its internal layers.[41] The goal is to make the network's decision process more explainable by determining how input features affect the model output. Reviewing the output of a convolutional neural network by having its internal state inspected by a human expert is an approach that is extended in related work such as References [42], [43] and [44]. Even when access to internal model states is completely unavailable, approaches such as randomized input sampling for explanation (RISE)[45] still provide insights for certain network types.

Even systems traditionally believed to be reasonably explainable with regards to inspection, (e.g. decision trees), quickly reach a complexity that defies understanding when deployed in real-world applications. In situations where an interpretable result is desired, tools such as optimal classification trees[46] or born-again tree ensembles[47] are applied to reduce complexity and allow for human expert review. See Reference [48] for a further discussion on the relation between AI model types and their interpretability.

Generally speaking, even when fully ”explainable AI” is not immediately achievable (such as a Class II AI technology), a methodical and formally documented evaluation of model interpretability is employed in regards to risk, subject to careful consideration of the consequences on functional safety risk that arise from inappropriate decisions. This aids in comparability and model selection and provides insights during an ”after the event” failure analysis.

8.4 Issues related to environments

8.4.1 Complexity of the environment and vague specifications

AI systems are often used under environmental conditions whose complexity is difficult for humans to fully analyse and describe. AI technology automatically generates rules, or apply judgement, without reliance on human-generated representations of analytic, detailed and complex environmental conditions. Further, the development life cycle for AI systems has the potential to begin with vague specifications or vague goals. Such specifications are given in the form of either functional or non-functional requirements.

Vagueness of specifications leads to difficulty during assurance of functional safety-related properties. The complexity of the environment only worsens the situation. Even the definition of a tolerable level of functional safety is likely to be undermined by a vague specification, because the definition of ”safety function” depends on the given specification.

Functional safety applications involve some minimal explanation for the functional completeness (as defined in ISO/IEC 25010[148]). Functional completeness is addressed by seeking an extremely detailed specification or by seeking to cover the whole complex environment (e.g. by training data), or by the combination of the two. See 9.3 for additional information.

Another feature of AI systems complexity is that, although their models are often deterministic, their output seem to be probabilistic. For example, given a very complex environment that is represented by a large state space and is subject to constant change and expansion, it is difficult to ensure that a model which generalizes the behaviour well under finite state conditions reacts appropriately to every possible state of the environment.

The effect of operating in complex, not completely defined environments, results in a new type of uncertainty beyond the scope of current functional safety assessments.

The extent to which the adequateness of models for the intended application is considered. Additionally, uncertainty of the model caused by possible incorrect predictions and misclassifications is considered in terms of behaviour planning and functionality.

To address the stochastic concept to address the operational environment, expanding stochastic assumptions generally used in the functional safety discipline for the random failures of the hardware and “proven in use” software is a relatively novel approach that is relevant to AI technologies, see 10.2.5.

For a more comprehensive list of emerging issues, see References [92] and [93].

8.4.2 Issues related to environmental changes

8.4.2.1 Data drift

Data drift is a phenomenon that distribution of runtime input data departs from those used in training phase which causes degradation of performance including safety. Data drift is often tied to an incomplete representation of the input domain during training. This is due, for example, to failure to account for seasonal changes in input data, unforeseen input by operators, or the addition of new sensors that become available as input features.

Components containing AI technology are inspected for sources of data drift in the context of a risk analysis and adequate measures are planned where appropriate.

Some examples of data drift are attributed to failure to apply best practices in model engineering. Common examples include picking inappropriate training data, data whose distribution does not reflect the actual distribution encountered in the application context or omitting important examples in the training data. These problem instances are fixed by improved input data modelling and retraining.

Data drift is caused by external factors, such as seasonal change or a change in process that induces data drift. Examples include replacement of a sensor with a new variant featuring a different bias voltage, or the sensor encountering different lighting conditions in training and previously unseen data. In some cases, the model deals with data drift while already deployed, where retraining is not feasible. In these cases, the model is constructed to estimate correction factors based on features of the input data or allow for supervised correction.

Model design is expected to provide safe outputs even in the presence of previously unknown inputs. Following proper model engineering practices, such as establishing a sufficiently diverse training data set, does not reduce the importance of careful analysis as to whether the resulting model is generalized to production data. In addition, in the event of the model providing unsafe output, a careful analysis is done of the causes of the unsafe output and specification of the way in which the system recovers from dangerous states.

For example, Reference [32] illustrates the most common sources of data drift and proposes model improvements, such as simpler or computationally more efficient models, even when data drift occurs as simple covariate shift without an apparent effect on classification output. These performance considerations translate to the development and application of modern, deep neural networks[33].

8.4.2.2 Concept drift

Concept drift refers to a change in relationship between input variables and model output and is accompanied by a change in the distribution of the input data. For example, the output of a model is used to gauge the acceptable minimal distance of an operator at runtime based on distance measurements obtained by a time-of-flight sensor (input data). If the accepted safety margins change due to external factors (e.g. increased machine speed not accounted for in the model), concept drift occurs despite both process and inputs having stayed the same.

Systems ideally incorporate forms of drift detection, distinguish drift from noise present in the system and adapt to changes over time. Potential approaches include models like early drift detection method (EDDM) [34], detecting drift using support vector machines (SVM)[35] or observing the inference error during training to allow for drift detection and potential adaptation.[36] Furthermore, work to quantify drift in machine learning systems is available in Reference [37]. It is noted that drift detection implies some form of runtime monitoring and model updates that introduce system design and safety considerations at a software or system level (e.g. knowing when it is functionally safe to perform an update, detecting failed updates).

Concept drift is often handled by selecting subsets of the available training data or by assigning weights to individual training instances and then re-training the model. For reference, Gama et al. provide a comprehensive survey of methods that allow a system to deal with drift phenomena[38].

Some examples of possible mitigation technologies for drift problems are summarized in Reference [94], Chapter 7.8.

8.4.3 Issues related to learning from environment

8.4.3.1 Reward hacking algorithms

Reward hacking algorithms refers to methods where AI technology finds a way to “game” its reward function and thus find a more ”optimal” solution to the posed problem. This solution, while being more optimal in the mathematical sense, is dangerous if it violates assumptions and constraints present in the intended real-world scenario. For example, an AI system that detects persons based on a camera sweep decides that it achieves very high rewards if it constantly detects persons and thus follows them around with its sensors, potentially missing critical events in other affected areas. This is countered by employing adversarial reward functions, such as through an independent system that verifies the reward claims made by the primary function using AI technology and subsequently learn and adapt to counter the primary system. Another option is to pre-train a decoupled reward function based solely on the desired outcome and with no direct relationship to the primary function.

8.4.3.2 Safe exploration

Safe exploration is a problem that a priori safety requirements are enforced during data collection and training, which also limits the domain of training data input to learn about “unsafe”. The safe exploration problem is of particular concern when an agent has the capability to explore or manipulate its environment. This does not only pose a problem when referring to, for example, service robots, unmanned air systems or other physical entities, but also applies to software agents using reinforcement learning to explore their operating space. In these contexts, exploration is typically rewarded, as this provides the system with new opportunities to learn. While it is obvious that a self-learning system follow appropriate functional safety protocols when exploring, a system that controls process parameters and employs a random exploration function while not being properly disconnected from the dangerous process poses equal or greater risks.

8.5 Resilience to adversarial and intentional malicious inputs

8.5.1 Overview

Assessing the trustworthiness of an AI system includes determining the integrity of functional safety behaviour against adversarial attack and intentional malicious inputs.

In general, two types of inputs intentionally crafted for possible misbehaviour are distinguished in the field of AI; the first are those inputs that destroy integrity of software execution (such as buffer overflow or integer overflow), and the second are those that cause AI models to compute incorrect output without causing malfunctions at the software level. For the first class of problems, traditional information technology (IT) security requirements are considered, see ISO/IEC 27001,[149] ISO/IEC 18045,[150] ISA/IEC 62443[151] and ISO/IEC TR 19791[95]. These International Standards provide processes for the audit and certification of horizontal IT security requirements that are also applicable to AI systems and are not discussed further in this document. However, for the second class of problems, following best practices and observing existing International Standards for non-AI systems are not sufficient. Subclause 8.5.3 includes a discussion of the second class of problem with adversarial examples of natural origin affecting the mode of action.

NOTE 1 This document is limited to the achievement of functional safety even in the presence of an AI-specific security threat. It does not address how malevolent action arising from a cyber security threat is controlled.

NOTE 2 Properties to ensure freedom from intentional malevolent inputs are contradictory to those that ensure functional safety properties. For further information on AI-specific security threats, see Reference [94], Chapter 9.

NOTE 3 Properties that ensure resilience to adversarial attacks are contradictory to those that ensure functional safety properties. This is addressed as part of a higher level of system suitability considerations.

8.5.2 General mitigations

Following known proper functional safety precautions, supervision functions are applied to take over the system in the event that a functional safety problem is detected, ensuring no harm is done by the AI system.

For systems that need high-levels of functional safety, both random failures and systematic errors warrant careful consideration. Overall, failures and errors are addressed according to best practices, (e.g. through hardening, robustness, testing and verification). Additionally, specific countermeasures in the field of machine learning further mitigate risks for the additional types of failure and errors specific to AI technology.

8.5.3 AI model attacks: adversarial machine learning

Models of AI systems, especially those with higher complexities (such as neural networks), can exhibit specific weaknesses not found in other types of systems. Additional scrutiny is done when deployed in a functional safety context. Examples of model-specific problems include adversarial machine learning and others.

Adversarial machine learning is a type of attack on an AI system that has garnered particular interest recently. As known as “adversarial attacks”, it is often possible to trick an AI model into outputting vastly different results by adding miniscule perturbations to the inputs, which are carefully crafted via the means of an optimization process. In case of image inputs, these perturbations are generally imperceptible to humans and can also be equally well hidden in numeric inputs. While these perturbations are typically non-random, hardware failures or system noise already present in the input are possible cause of a non-negligible shift in model output, see Reference [49]. Interestingly, adversarial examples generally translate well across different model architectures and intrinsic model components[50],[51]. That, along with the number of well-known model architectures and pre-trained models available in so called “model zoos”, makes the practical deployment of adversarial examples seem very likely and hence a significant threat to systems using AI technology[128],[129].

Even a system seemingly resilient against modification of its inputs, (e.g. a system employing a local, non-cloud AI model directly connected to sensors), is not exempt from a kind of adversarial attacks. The feasibility of physical attacks on models, even those considered black boxes with no access to internal model details being available, has already been shown in 2017 in Reference [52]. More recently, Reference [53] has shown that it is possible to introduce adversarial examples into the forward inference process of a model, creating the aforementioned perturbations using physical stickers applied to objects and causing the resulting inference to diverge significantly from the optimal solution.

When the input to an AI model is susceptible to adversarial attacks, the net effect of such attacks affecting the functional safety is evaluated before deciding whether and how much countermeasure is considered appropriate or sufficient. The possibility of adversarial attacks in the real system varies greatly depending on how the AI model is deployed. For example, its possibility heavily depends on systems surrounding the AI technology, including input sensing (e.g. camera) and pre-processing. Furthermore, analysis of possible attackers and victims is done; if the only possible victims coincide with possible attackers, it is appropriate in some cases to omit protection.

One possible countermeasure for these problems is called adversarial training [132]. In essence, adversarial training tries to train an AI system with adversarial examples in an attempt to have the model encode knowledge about the expected output of such an attack. A next natural avenue of action is to attempt to remove the artificially introduced perturbations. Examples of this approach include:

- High-level Representation Guided Denoiser introduced by Reference [56];
- MagNet, which aims to detect adversarial examples and revert them to benign data using a reformer network[57];
- Defense-GAN, employing a generative adversarial network[58].

It is worth mentioning that scenarios exist where both MagNet and Defense-GAN failed, see Reference [59].

Furthermore, noting that the model types typically affected by adversarial attacks are in general robust against noise, several authors propose randomization schemes to modify the input and increase robustness against malicious, targeted noise. Approaches include random resizing and padding[60], Random Self-Ensembles[61] and various input transformations such as JPEG compression or modifications of image bit depth[62]. While these methods are effective, recent results show that these transformations are not sufficient measures under all circumstances. In turn, if input transformations are used as a layer of defence against adversarial examples, the efficiency of said protective measures are evaluated against examples generated using the expectation over transformation (EOT) algorithm presented in Reference [63].

Goodfellow et al. argue that the use of models employing nonlinear components makes them less susceptible to adversarial attack at the cost of increased computational resources[54]. The problem of examining and augmenting the optimization methods used during training is addressed in Reference [50]. Model ensembles are often applied in order to create a more robust overall model through diversification. However, there are also results in the literature that show that diversification does not always sufficiently harden the system against adversarial attack, see Reference [55].

In addition to the attacks modifying the input to the running systems, it is also possible to put perturbation during the learning process by injecting malicious data during the training phase, which is called “model poisoning”[127]. Considerations for protecting the learning processes and data collection steps are known issues here.

8.6 AI hardware issues

AI technology does not make decisions by itself; it relies on algorithms, software implementing the algorithms and hardware executing the algorithms. Faults in the hardware can violate the correct execution of an algorithm by violating its control flow (causing memory-based errors, interfering with data inputs such as sensor signals) damaging the outputs directly, and generally cause erroneous results. This clause describes some hardware aspects when using AI technology that can affect functional safety. In short, reliable hardware is as important in AI systems as in non-AI systems. Like hardware used to execute non-AI software, the hardware used to execute AI technology can also suffer from random hardware failure. A list of relevant fault models can be found in International Standards such as IEC 61508-2[17] and ISO 26262-11[14].

8.7 Maturity of the technology

Technological maturity describes how mature and error-free a particular technology is in a particular application context. Less mature and new technologies used in the development of an AI system introduce risks that are unknown or difficult to assess. For mature technologies, a greater variety of experience is usually available, making risks easier to identify, assess and address. However, mature technologies come with a danger of decreasing awareness of their potential effect on risk over time, so that the positive effects depend on continuous risk monitoring (e.g. based on collected field data), as well as appropriate awareness training and maintenance.

9 Verification and validation techniques

9.1 Overview

This clause describes the difference between verification and validation techniques in AI systems as compared with non-AI systems, as well as some considerations for solving or mitigating problems arising from these differences applicable to functional safety. This clause addresses four significant aspects of such differences, although potential differences are not limited to those described in this clause (see Reference [136] for additional examples). ISO/IEC TR 29119-11:2020,[152] Clauses 7 to 9, are also worthy of consideration.

This clause focuses particularly on data-driven models e.g. those created by machine learning. Subclause 8.1.2 describes this class of models as the main challenge for ensuring the functional safety of an AI system. This is because the functional safety of other types of AI technology sometimes is achieved by applying the principles of existing functional safety International Standards, as discussed in Clause 7. This clause pertains to Usage Levels from A1 to C of Class-II AI systems (see Table 1).

When aiming for functionally safe systems containing AI technology created from data, it is taken into account that the AI technology is not constructed by rules as in non-AI developed systems. This means in particular:

— What is not in the data cannot be learned.

— What is in the data is likely learned, but not always perfectly.

Furthermore, just having data in most use cases is not sufficient. Labels are used when applying supervised learning techniques. Incorrect labels are one of primary causes for errors during the learning process. A thoroughly defined data engineering process addresses these aspects.

If the model is derived from a data set, the content of this clause is also useful for the training and validation data sets.

NOTE The terms “validation” and “verification” refers to different concepts among different technology areas or domains. In the context of machine learning technology, “validation” means a process step to check convergence of the developing model to terminate the AI training process, which is quite different from the concept of verification and validation used in the functional safety community. Model convergence is a precondition for testing, but it does not guarantee the quality of the final product. For example, the “reward hacking” problem arises from a model that is subjectively designed to maximize the given reward function. In this document, the terms verification and validation are almost exclusively used in the context of the functional safety concept.

9.2 Problems related to verification and validation

9.2.1 Non-existence of an a priori specification

During the training phase of machine learning models, the selection of training data (together with the definition of the loss function, if applicable) is replacing the definition of a formalized specification of operational behaviour. This leads to problems with the traceability of individual aspects of behaviour as there are no individual specification statements. Instead, the information which replaces the discrete specification statements is implicitly contained in the collection of training data[135]. Although it is a benefit of machine learning that it derives or acquires knowledge from poorly structured data, the lack of a predefined specification causes a significant problem for verification and validation, as well as for the evaluation of the uncertainty. See Reference [137] for wider discussion.

Another source of risk is the presence of bias or incompleteness in the data used to train the model. Techniques are typically deployed to check for both these sources of risk.

9.2.2 Non-separability of particular system behaviour

During development of software for non-AI functional safety-related applications, each risk described in Clause 8 that considered "tolerable" that has been identified during a Hazard and Risk Analysis (HARA) is mapped to one or more mitigations. The implementation of the mitigations and their role in maintaining functional safety is explained. Usually, mitigations are designed not to interfere with each other, so that the effectiveness of each mitigation is verified, validated and evaluated separately.

On the other hand, many AI technologies are considered as a "black box", as their internal behaviour and the basis of their decision-making processes are difficult for a human to understand. This means that if the training data set contains some data that are intended to work as a mitigation for a particular risk, its influence on the trained model is not certain, nor tested separately for each risk. Furthermore, if some additional training data are added for an additional mitigation, the data affect existing measures for mitigation of other risks. This makes verification and validation of machine learning models more difficult.

9.2.3 Limitation of test coverage

Testing AI technology is difficult when compared to the process of testing non-AI software. In general, two types of tests for software are often designed and performed: one focuses on structure of the problem description and the other focuses on the structure of the implemented software. In non-AI software, these two kinds of focused structure have some degree of correspondence, which enables efficient testing. Most data-driven AI technology lacks this property, unfortunately, which makes many existing techniques for non-AI software (including those described in existing functional safety standards) not efficiently applicable or even not effective at all. This difference is given careful attention when designing tests for any AI technology (especially those based on machine learning).

9.2.4 Non-predictable nature

As noted in 8.4.1, AI system outputs are often said to be non-predictable or probabilistic in nature, although the algorithm itself is deterministic. Mitigation is approached through systematic application of the verification and validation process, with careful considerations for the nature of the AI system. Again, ”explainable AI” is a future solution, but process-supported solutions are more often available.

Further, the apparent non-predictable or probabilistic nature of AI technology, as well as other causes, such as discussed in 9.2.3, decreases the effectiveness or applicability of non-AI testing techniques, especially white box based testing technologies. See ISO/IEC TR 29119-11:2020, Clause 9 for alternative solutions for white box-based testing applicable for AI systems.

9.2.5 Drifts and long-term risk mitigations

Drifts (see 8.4.2 and 8.4.3) are other causes of uncertainties in the long-term system installation. Even if the systematic and comprehensive analysis of its behaviour in the operational environment has been performed, AI models can still suffer from the concept and data drifts, because a training data set contains an intrinsic bias for training-time environments. To overcome such drift, several methods for re-training and updating the model exist for most real-world applications of AI systems.

In addition, updating the software is a significant undertaking, especially in applications involving functional safety. Related assessments and considerations for update procedures are carried out from the earliest stages of the system design.

9.3 Possible solutions

9.3.1 General

9.3.1.1 Directions for risk mitigation

Generally, there are at least two types of approach to realize reliable verification and validation of implementations generated from data-driven models.

One type of approach, generally the more difficult, is to analyse the generated model to extract human-understandable knowledge of the model's expected behaviour. Theoretically, if the behaviour becomes completely human-explainable, it enables AI technology and systems to be treated as Class-I AI. This type of approach is further discussed in 9.4.

The other type of approach is to evaluate achieved levels of safety risk mitigations indirectly by analysing how the AI system is constructed during the development process. Although testing of machine learning-based AI is not always complete, additional analysis and assurance on the development processes and its inputs mitigate the risks of unwanted behaviour systematically. The rest of 9.3 is mainly focused on this type of approach.

9.3.1.2 AI metrics and safety verification and validation

Typically, metrics such as accuracy are used during training of machine learning algorithms. These metrics are part of managing the progress of AI training. Although better accuracy suggests better quality for functional safety-related applications, it is not generally enough for ensuring required functional safety properties. This Clause describes mitigation typically applied in parallel or in sequence to the evaluation using AI technology metrics, especially in data design, data preparation and testing phases.

9.3.2 Relationship between data distributions and HARA

Data-driven processes are based on the relationship between risks identified in the HARA and the data distribution. For a given use case, the question becomes whether an AI system is given sufficient training and test data to develop a particular behaviour and if there is some conceptual correspondence between the outcome of the specific HARA activity and the design of the data set to be used. This approach is considered similar to that of non-AI safety-related software, for which a set of risk mitigations has been identified.

Typically, the operational domain of the system is defined and bounded as precisely as possible whether the initial specification is predefined, or derived from example data instances. The boundary is defined either as an input data space or as a profile of real-world usage. Metrics are established for checking that data set and verification and validation activities correspond to the defined domain at an early stage of development.

Logical analysis of the input data distribution is performed in addition to collection and learning from the given data set. Such an analysis relates to the outcome of the HARA activity, so that data distribution points in the data set are identified as corresponding to each identified risk—see, for example, the ISO/IEC 5259 series[1),[125]] which highlights that data quality is key for AI technologies.

In addition, even if the input data set is well-designed, there is no guarantee that the training process will encode the intended behaviour into the output model corresponding to each identified risk from the data distribution observations. Both systematic errors and random errors can occur during training, which can cause functional safety goal violations. While detecting such failures to the best degree possible is one of the

[1)] Under preparation.

intentions of testing activities in the verification and validation phase, a means of mitigating training errors is also considered during the training phase.

9.3.3 Data preparation and model-level validation and verification

As described in ISO/IEC TR 29119-11:2020, 8.1 and ISO/IEC/IEEE 29119-4, the design target for training and test data sets is determined in relation to HARA results. This data requirement is further divided into four criteria:

a) Whether all functional safety relevant scenarios identified during HARA have corresponding data included in the given data sets.

b) For each identified risk in the HARA in a), whether the test data cover all reasonable variations of situations which cause such a risk.

c) For each risk-causing situation in b), whether the test data are both sufficiently large and sufficiently diverse to result in complete coverage of the possible states of the system after training.

d) For each risk-causing situation in b), whether the results of the selected test case data are stable under the variations which are expected for inputs which according to human analysis would be classified as belonging to the same group, scenario or use case.

Each test activity is expected to give answers for each of the four criteria. The following considerations are one possible set of known answers for the criteria, applicable to any AI technology for which test data are attributed with clear, correct and expected answers ("test oracles"). In these examples, bias in the data is also considered, see Reference [10].

For a):

— Specification of the sets of data attributes corresponding to each identified risk in the HARA.

For b):

— For each identified set of data attributes for an identified risk, checking for the existence of the test data within test data set.

— For the subset of test data extracted for each identified risk, checking for the distribution of other attributes and assess whether the data are unintentionally biased toward specific situations; for this purpose, existing technology for test designs for non-AI software (e.g. combinatorial testing) are used. See ISO/IEC TR 29119-11:2020, 8.1 and ISO/IEC/IEEE 29119-4 for further details.

— Considering collection of additional test data if it is suspected there is unintended bias in the data set. In some cases, synthesis of test data from simulations is a solution, if sufficient diversity, representativity and coverage are not obtained from real data. See ISO/IEC TR 29119-11:2020, 8.4 for some examples. One known option for the developer also is to remove real data to rebalance the data set.

For c):

— The mitigations identified in bullet b) are used for diversity with existing attribute values.

— Assessing the data collection and preparation processes so that any unwanted bias is not likely included in the test data set; see ISO/IEC TR 24027[10].

— The amount of test data is determined from inputs including (but not limited to) the intended probability of risk mitigation (derived from the HARA) and the amount of data needed for training (derived from monitoring the accuracy indicator for the subset of training data). In addition, complexity of the operational domain is considered to mitigate data distribution shifts occurring by many uncontrolled factors (e.g. time, weather, location).

For d):

— Ensuring that over-fitting to the training data is detected within the development process. One known way of achieving this is to ensure the independence between training data and test data, which is enforced through development process management and assessment, tool-based approaches, or even using a level of independence in the teams or organizations carrying out the testing (see IEC 61508-1[16], Clause 8 or BS EN 50128:2011 [23], Clause 5). Another known method to detect overfitting is cross-validation, in which models are trained on several different subsets of the training data, and the performance is evaluated on the held-out data. Several methods are available, see Reference [82].

- Ensuring that trained models have sufficient robustness in terms of the given problem, using the following approaches:
  - generating multiple models of different sizes, using smaller models so long as other objectives are met (large models lead to excess sensitivity);
  - applying a technology that improves robustness, (e.g. regularisation or randomized training);
  - numerically and directly evaluating the robustness, (e.g. using safe radius[84] — this is an emerging discipline).
- Searching for possible data that affect stability: (e.g. metamorphic testing,[85] data augmentation,[86] generative adversarial networks,[87] adversarial training,[88] adversarial example generation[89] or adversarial example detection[90]).
- Ensuring that the training data set and test data set are free of malicious modifications or alterations; this entails reviewing the credibility of data source or data collection processes.
- Considering use of explainability technology for analysing behaviour of output model (see 9.6).

There are several references available for proposing some concrete technologies and techniques representing these criteria. Annex C gives some examples for applicable procedures and techniques.

The costs for implementing these mitigations vary considerably on the depth of investigations, on used levels of combination in b), and chosen technology. Verification and validation are planned according to the required level of functional safety and other application criteria.

9.3.4 Choice of AI metrics

The performance and KPIs of a system containing AI technology is thoroughly evaluated. In machine learning often single metrics are used. The following is typically considered for metrics:

NOTE Metrics are not typically the only measure to assess the safety of a system containing AI but only one aspect.

- The significance and trustworthiness of a metric: this is connected to the amount of data available for training, validation and testing—the amount of data has a bearing on how much trust is placed in a metric with a defined confidence level (e.g. 95 %) based on the number of executed test cases.
- Metrics reduce information: such a reduction of information obscures safety issues. Various metrics are used to identify missing information, e.g. safety related misclassifications to assess the targeted performance or KPIs.
- Field monitoring: collection of data on the performance of an AI system during the operational phase assesses whether the performance and KPIs are still being met during the operational phase – intervening actions are considered if the assumptions on which the safety of the system is based is no longer being achieved.

ISO/IEC TR 24029-1[153] separates the robustness assessment into three core categories: statistical, formal and empirical-based tests.

9.3.5 System-level testing

In complex systems using AI technology as a component, system-level testing is a complement to verification and validation at the detailed level. Some of the criteria described in 9.3.3 for example criteria b), are also applicable for system-level testing. System-level testing are either data-based or scenario-based (e.g. running a test vehicle in test fields with simulated risks). System-level testing are carried out in simulations, as a digital twin, or in the real-world application. Real-world testing is expensive and not always possible (due in part to risks to safety) but it is useful for validating KPIs and unveiling unidentified hazardous unknowns to mitigate against incomplete HARA. Simulation is useful for exploring large numbers of scenarios in both software-in-the-loop and hardware-in-the-loop settings. Good verification and validation results depends on the quality and realism of simulators. See 9.4.2 and 9.4.3 for more descriptions.

9.3.6 Mitigating techniques for data-size limitation

Preparing sufficiently large test oracles to test all outcomes is infeasible within development life cycles. Back-to-back testing, as described by ISO/IEC TR 29119-11:2020, 8.2, is used to annotate test oracles with the expected answers. The extent of independence between the different versions of the system to be tested is assessed carefully. Back-to-back testing with AI technology generated from the same source of training data likely fails to address criteria a) and b).

Another solution, where a large test oracle is used to address the full range of operation, is to use simulation as a test data generator.

For some AI systems it is difficult for engineers to construct a reliable test oracle (e.g. AI systems constructed with ”AI-versus-AI” competitions, including reinforcement learning and Generative Adversarial Network). The general conditions for testing in these cases are similar; however, additional criteria for reliability of tests applies. For example, well-tested alternative implementations are used to undertake back-to-back testing. Alternatively, a design change is implemented to separate any risks from influence from the model-driven AI technology, effectively converting to Usage Level C as described in 6.2.

9.3.7 Notes and additional resources

See ISO/IEC TR 29119-11:2020, Clause 9 for alternative solutions for white box-based testing applicable for AI systems.

9.4 Virtual and physical testing

9.4.1 General

Functional safety approaches for AI technology tend to focus on elements of the AI system that are shown to ensure functional safety attributes, for example functional safety or rule monitors that overrides the primary control system to inhibit unsafe action. An effective and objective way to demonstrate a system’s performance is via virtual testing or simulation, where a curated set of well-chosen stress-test scenarios are exercised during the qualification and certification activities. Individual components are tested, as well as multiple components at a system level. Such approaches use constrained random selection of scenario parameter values, scenario testing based on parameter distribution or importance sampling when constructing the scenarios to be tested (see ISO 21448:2022[7], Clause C.5).

Physical tests are also considered to correlate simulation results, validate KPIs and uncover unknown unknowns. Physical tests are far more limited than simulation in their ability to probe the domain space due to cost and time limitations but do test some aspects that are difficult to emulate in a simulation, for example, the effect of hardware delays on feedback loops and cascade effects. Structured tests take place in which tests are set up for known scenarios, such as on a test track for automated vehicle applications.

9.4.2 Considerations on virtual testing

The use of simulation for testing has long been an integral part of functional safety. Established methods such as timing simulation and fault injection have direct extension to AI systems, and their use is encouraged. For the complex, high dimensional models featured in many AI solutions (such as neural networks for perception or decision-making tasks), simulation offers many additional benefits:

— For certain applications, simulation provides more complete test coverage than real-world testing. Examples include scenarios where real-world testing is dangerous or prohibitively expensive to conduct at large scale over the possible input space. For models with high dimensional inputs, simulation is used to iterate over the input space and produce correlated results in ways infeasible by traditional testing.

— Simulation greatly speeds up development time, allowing greater access to functional safety products and updates. Newly discovered hazards are incorporated into the functional safety solution with much improved turnaround time. For highly complex environments, this reduced latency in the development and update cycle is critical.

— Simulation provides multiple entry points for fault injection. Faults are introduced at the system, component or subcomponent level, and they are introduced in combinations that are inaccessible by real-world testing.

— Simulation provides accurate ground truth, which negates the potential of systematic errors induced by real-world measurements and setup.

— Simulation environments are well-controlled and track all metadata associated with a particular test. This prevents any systematic bias introduced in a real test, or loss of relevant metadata.

The following items are worthy of consideration when introducing simulation or in general virtual testing as part of the verification and validation process of AI technology in functional safety systems.

— Fidelity of simulation: consider the underlying models, toolchain, simplifications and assumptions. A risk assessment of the simulation environment addresses the implications of inaccuracies, imprecision or incompleteness of the simulation environment. Evidence is used to support the claims of the simulation output, such as a simulation to real-world correlation. For example, a simulator used to justify a functional safety component used for perception includes arguments about the realistic rendering of the scene, metrics to correlate the two, indistinguishability by human observers, etc. See 9.4.3 for more discussion.

— Type of simulation: no one virtual testing tool is used to test all aspects of an AI system. This is why multiple tools sometimes are used to develop confidence in the functional safety of the full AI system. A virtual testing toolchain includes the following types: MIL (model in the loop), SIL (software in the loop), HIL (hardware in the loop).

— Test-coverage approach: approaches include random test sampling, constrained test sampling based on certain justification of the input space, distribution-based test sampling based on a user profile, criticality or importance test sampling based on functional safety analysis, stress-based sampling based on edge cases or expected conditions that stresses the system, etc. (see ISO 21448:2022[7], Clause C.5). For multi-dimensional inputs, this also addresses what combination of factors are tested.

— Test coverage size: i.e. the amount of simulation which is sufficient to justify the functional safety argument.

Before virtual testing tools is used to validate or approve an AI system, the toolchain itself is verified and validated. Confidence in a virtual testing toolchain is achieved by assessing four key attributes:

— Fit for purpose: the extent to which the tools are suitable for the AI system assessment. Fitness for purpose addresses a clear description of the test objective and a definition of all boundary conditions of the AI system. It involves analysis of the operating environment and derivation of the requirements for the individual simulation models. The complexity and level of detail for each model varies depending on the relevance, significance and range of each factor. For example, if the operating environment excludes night operation, then the sensor models will not be validated against low-light conditions.

— Capability: the extent to which virtual tests reveal faults and the associated risks of dormant faults. Test capability involves defining assumptions, limitations and fidelity levels of the toolchain, ways to assess the fidelity (KPIs), and reasonable tolerance for the KPIs. It supports justification that the tolerance for simulation to real-world correlation is acceptable for the test objective. Note that the chosen fidelity level for the models and the assumptions made play a major role in defining the limitations of the toolchain.

— Correctness (verification): the extent to which tools' data and algorithms are sound and robust. Verification looks into the implementation of the conceptual or mathematical models building up the toolchain. It provides assurance that the toolchain does not exhibit unrealistic behaviour for a set of inputs that are not tested during the validation phase. The procedure is grounded on a multi-step approach that includes code verification, calculation verification and sensitivity analysis.

- Accuracy (validation): the extent to which the virtual tests reproduce the target data. This includes generating data that are used to demonstrate the accuracy of the virtual testing tools with respect to the real-world. Toolchain validation consists of 4 main steps. The exact methodology depends on the structure and purpose of the toolchain. The validation consists of one or more of the following:
  - Validate Subsystem models e.g. environment model (infrastructure, weather conditions, user interaction), sensor models (Radar, Camera, Light Detection and Ranging (Lidar)), chassis model (actuation, powertrain);
  - Validate chassis system (chassis model together with the environment model);
  - Validate sensor system (sensor model together with the environment model);
  - Validate integrated system (sensor model plus environment model with influences from chassis model).

Applying the same scenarios across all tool levels (MIL, SIL and HIL) allows effective validation of the system without requiring an impossible number of physical tests to be carried out.

The usage of virtual testing tools depends on the virtual validation and verification strategies implemented during their development. Therefore, the simulation design and the toolchain are not typically standardized but rather explained and reviewed during the certification process.

Therefore, the overall assessment of a virtual testing toolchain requires a unified method to investigate these properties and gain confidence in the data generated by the tools. Simulation models and the simulation tools used in the overall toolchain are investigated in terms of their impact in case of a safety error in the final product. The approach for criticality analysis is described in IEC 61508-3 or ISO 26262-8, which requires qualification for some of the tools used in the development process.

9.4.3 Considerations on physical testing

Physical testing has a complementary role to simulation testing. Testing the system in a real-world environment, or final operating environment, provides the highest fidelity of real use validation. For real-world testing, some additional considerations include:

- Use of structured tests, setting up known scenarios, or use case tests. Examples include test track cases for autonomous vehicle applications, or defined scenes for sensor perception tasks. Such tests are well specified and provide controlled measurements that are tracked and compared over time. Structured tests are derived from many different inputs, such as safety, technology and product level analysis. A comprehensive test plan requires good understanding of the final application.
- Combination of real-world testing with simulation. Physical tests are far more limited than simulation in their ability to probe the input domain space due to cost and time limitations, but provide the highest real use fidelity and automatically capture random phenomena that are not simulated. In contrast to structured tests, which typically test “known knowns”, both real and simulated testing uncovers different types of “known unknowns”.
- Correspondence of real-world testing to simulation. Real-world tests are used to validate the models used in simulated tests.
- Continuous testing and feedback. Real-world testing also uncovers “unknown”. Reported incidents (and possibly data collected from incidents) provide information to continuously advance the qualification simulation scenario suite.
- Domains for testing: the boundaries of operation. Domains for testing (for both simulation and real-world testing) parallels the defined operation for real operations. That domain includes limits of use, environmental limits, location and temporal limits, and responsibilities between the system and users, and if appropriate, other systems. In addition, tests are evaluated with metrics to show coverage of the design domain (this applies to both simulation and real-world testing).

- Statistical significance. Test procedures and results are derived from sound statistical principles. For example, a final on-site validation test of a safety stopping function is carried out multiple times to demonstrate that relevant parameters fall within a predefined limit based on statistical analysis. In contrast, verification tests of a perception function for human detection are carried out on a large test database, with size and coverage determined from target failure rates and confidence intervals.

9.4.4 Evaluation of vulnerability to hardware random failures

It has been shown that the vulnerability of deep neural networks to soft errors is low (see References [25], [83]). Evaluation of the fraction of failures leading to safe behaviour (as opposed to unsafe behaviour) is useful for certain types of networks. Possible methods include fault injection (e.g. individual weights in a neural network) as a proxy for faults in underlying hardware. For example, it is possible to analyse classification models to determine with confidence the only vulnerable parts of the AI technology with respect to soft errors (see References [26], [78]).

9.5 Monitoring and incident feedback

Once an AI system is approved and in operation, its own incident statistics are used to provide ongoing evidence of safety performance. Reported incidents are used to feedback information which are used to continuously enhance the scenario suite used during the testing activities. However, for the core functions where inductive or deductive absolute proof is not possible, acceptable failure rate targets are derived from system failure rate goals, together with suitable justifications to substantiate the functional safety. If this is demonstrated empirically, the test methodology and results are recorded.

Operational design domain and real-world usage profiles are used to define and bound the problem scope, creating metrics for coverage of testing (both simulated and real).

Statistical significance considerations derive test data set size and test coverage from target failure rates and confidence intervals, following acceptable confidence levels.

Recording of field data, if conditions allow, is considered viable for motoring safety performance of the system and for appropriate incident response.

9.6 A note on explainable AI

A type of evolving AI technology, known as “Explainable AI”, aims to provide important factors influencing an AI-based decision in a way that humans understand (see ISO/IEC 22989:2022). Sufficiently explainable AI technology, if successfully realized, enables developers to understand the AI decision-making algorithms and paves the way for assurance of functional safety of machine-learnt algorithms in a similar way to current functional safety International Standards. Alternatively, some knowledge is sometimes extracted by a human observer of the AI model and then re-implemented as traditional software.

Although it is currently impractical to enforce sufficient explanation of decision-making for every class-II AI system, there are some currently achievable approaches to interpretability or explainability of the model structure, which possibly helps in the verification and audit processes. For example, heat maps of the internal nodes contributing to specific decisions are useful for understanding the causes of decisions[96]. Such techniques, sometimes called “grey box” approaches, are useful for the understanding of the behaviour of an AI system, especially when it differs in its decision-making from the implementors’ intentions. In such cases, those techniques consider whether the meaning of the extracted explanation is consistent or inconsistent with functional safety requirements. For example, understanding of the decision-making process extracted from the mid layers of DNN is not alone sufficient for justifying safe behaviour, since unexplained processes in other layers invalidate such explanations.

Refer to 8.3 and ISO/IEC TR 24028:2020, 9.3[11] for further information on AI explainability.

Additional information can be found in References [158] and [159].

10 Control and mitigation measures

10.1 Overview

Having a good and robust architecture of an AI system capable to tolerate a failure without loss of safety properties is preferable than improving AI quality only. The architectural design principles for safe systems are not changed by machine learning, though they impose new challenges in defining and guaranteeing their reliability properties and failure behaviours.

This clause considers the methods for enhancing ML models as components of AI systems and discusses how subsystems around them are used to improve non-functional properties such as reliability, availability, and quality. Subclause 10.2 describes AI subsystem architectural considerations including mitigation and control methods. Subclause 10.3 proposes methods to increase reliability of components. The failure mechanisms from Clause 8 have highlighted differentiating challenges for ML components, while Clause 9 addresses the through-life process of verifying and validating these components. Measures introduced in this clause are directed by knowledge of these failure modes and are introduced as part of a robust ML process described in Clause 11.

10.2 AI subsystem architectural considerations

10.2.1 Overview

Safety assessment at a system level determines the appropriate subsystem reliability of a function incorporating ML components. The subsystem architecture incorporates complementary technologies to meet these demands. Based on the reviewed literature, consideration of the following features drives different possible solutions:

a) Safe (suboptimal) back-up function to the ML component is designed with ”non-AI” techniques. This back-up decision system is, for example, an alternative controller or a failsafe null action (e.g. power-off). The back-up action allows the use of detection methods to switch the output when unsafe conditions are detected.

b) A safe subset of the action space is determined (a priori or online) using a supervisor function with constraints or limits.

c) ML redundancy with output voters or aggregators is considered.

NOTE These architecture solutions co-exist or are alternative, i.e. one of them is sufficient.

The inclusion of AI technology introduces specific challenges to each of these architectural options. Subclause 10.2.2 describes how detection mechanisms for abnormal input, output or internal state (e.g. neuron activation strength) are used to identify situations of possible failure. Subclause 10.2.3 describes how to use supervision functions using elements of control theory to minimally bound AI operation. Subclause 10.2.4 shows different ways of establishing redundancy with AI technologies. Finally, subclause 10.2.5 discusses AI system design with statistical evaluation.

10.2.2 Detection mechanisms for switching

The architecture in Figure 5 is often denoted as passive (diverse) redundancy in fault-tolerant systems literature. For example, a supervisory monitor detects when an AI technology is producing potentially unsafe actions, either due to internal or external faults. Following detection, an action is taken to maintain the system in a safe state. The monitor is developed using either non-AI technology or using AI technology. In the latter case, considerations of the level of independence between the monitor and the primary system are used to justify the approach.

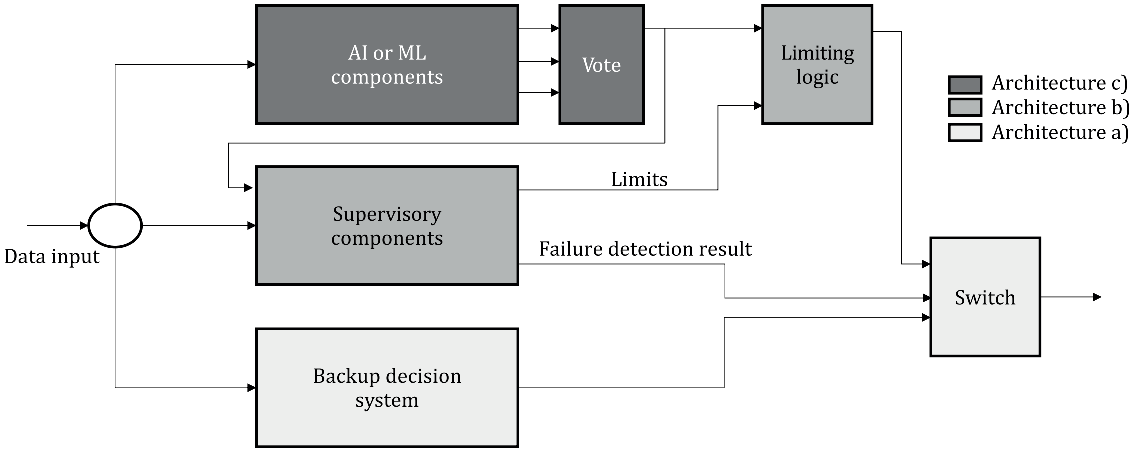
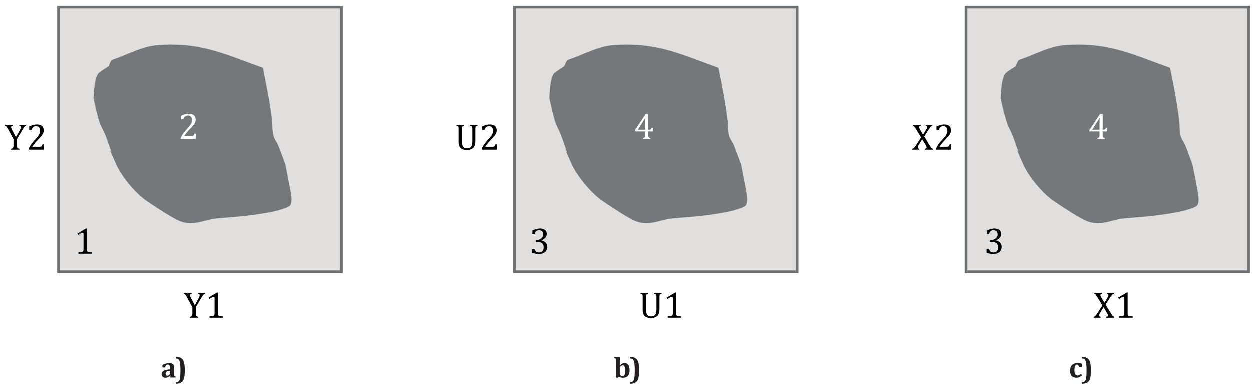


Figure 5 — Architectural patterns for systems using AI technology components

Acceptable behaviour of AI technology is evaluated through the context of its training data distribution [see Figure 6, a)]. It is typically not possible to verify the model behaviour in case of "unseen" out-of-distribution (OOD) input data, this includes the result of data drift. An acceptable model output relies on untested generalization properties of the model and could be erroneous, thus motivating the detection of such input data. The distribution depends on both the parameters in a single sample and their evolution in time (i.e. dynamics). Simple boundaries on acceptable inputs are unlikely to detect gaps in the training data, particularly for high dimensional systems. In the absence of training data improvements, anomaly detection methods (see Reference [97]), in some cases, are selected based upon the data properties (dimensionality, linearity of parameter correlations, dynamics, seasonality drift, etc.) (see References [98] and [99]). However, adversarial methods have shown the extreme sensitivity of deep networks to small, seemingly random perturbations (e.g. misclassification of images by adding noise, making reliable input parsing challenging). Adversarial methods include use of run time checks and checks on incoming data to determine whether there are adversarial attacks or not (see References [100] and [101]).

In contrast to input monitoring [see Figure 6, a)], output monitoring detects undesirable behaviour resulting from OOD or in-distribution input data. Monitoring of the output against a known boundary [see Figure 6, b)] or alternative model, is well known in fault detection literature (e.g. statistical or model-based residual detection). Boundaries are adaptive to a sequence of multi-variate system inputs and thus prohibit the system from leaving a safe state region [see Figure 6, c)]. For dynamic systems, the detection decision also ensures that a safe state is reachable without constraint violation (i.e. system inertia or instability do not prevent dangerous states being entered under back-up controller decisions). ML is used to create a monitor where the output distributions defy conventional modelling techniques. One example is to train a (simpler) secondary network (e.g. a student–teacher architecture) on the output generated by the ML model and use this with labelled data to predict confidence in the output[97].



Key

X1	input (Y1, Y2, ...)
X2	output (U1, U2, ...)
Y1	input Y1
Y2	input Y2
U1	output U1
U2	output U2
1	unseen
2	tested
3	unsafe
4	safe

Figure 6 — Evaluation of acceptable behaviour of AI technology

Meta-information from the model are used, such as the internal neuron activations or by designing uncertainty measures into the network. Uncertainty measures in ML are explicitly derived (Bayesian neural network) or approximated (dropout, ensembles, softmax output layer). Confidence of the output (strength of activation) is useful but requires careful calibration to a probability and is subject to risks. These risks include overconfidence (classifiers often fail silently by providing incorrect but confident outputs), not only at extremes or beyond bounds of training space but also to small perturbation as in the case of adversarial examples.

Four points are considered for the development of monitors:

— the type of AI technology faults that is detected;

— the ways in which AI technology faults are revealed at runtime;

— the performance benchmarks of different runtime monitors;

— the types of intervention that are used to circumvent a fault after detection and also circumvent the potential hazards invoked.

Examples related to machine learning are provided in Reference [75].

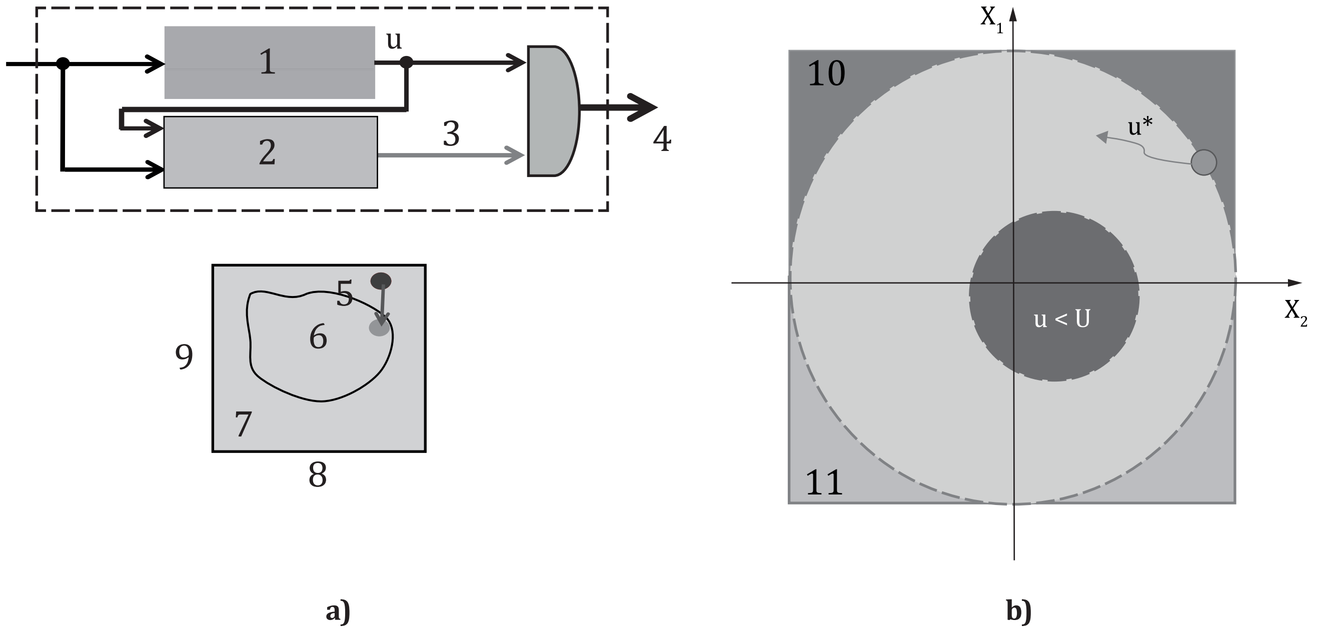
Uncertainty wrappers as described in Reference [139] that evaluate the quality of the decision are instrumented for either automated decision-making or as input for humans to decide if the AI system proposes valid decisions.

10.2.3 Use of a supervision function with constraints to control the behaviour of a system to within safe limits

With an appropriate supervisory module, it is possible that an AI system is constrained to work within a predefined safe envelope. Safe limits require that a subset of the action space (safe envelope) is determined and are minimally restrictive on safe ML component behaviour. However, simple limits on the output overly inhibit an ML component, resulting in behaviour that mimics the limiter itself, therefore negating the benefit of implementing the ML component. This subsystem architecture is sometimes referred to as a safety cage, which enforces behaviour onto the subsystem.

For example, as shown in Figure 7 a), an AI system is used as part of the intelligent control to provide an optimal decision. In this architecture, the non-AI safety function outputs an acceptable range of outputs for a given input and limits the intelligent control output.

Constraining the output based on a function of input is not usually appropriate for systems with dynamics, where the system state (x) defines the unsafe regions but does not instantly respond to a change in the controller input. Formally, minimal bounds are designed for dynamic and hybrid systems through control theory methods such as barrier functions approaches (Reference [102]), which determine an invariant set under the control of the non-AI system, see Figure 7 b). These approaches guarantee that the system does not exceed an operating region deemed safe for some limited worst-case control input (the subset u of all possible control signals U). These sets are typically conservative as they do not actively look to recover a system back towards safer regions, thus control barrier functions are used as detection mechanisms to switch in conventional stabilizing controls (producing control signal u^*, if they are available with suitable AI monitoring, as described in 10.2.1). For systems with particularly complex safety requirements, for example multichannel measuring systems that use AI technology, checking functions such as metrological self-check or self-validation are considered, see Reference [71].



a)

b)

Key

1 intelligent control
2 safety verification
3 upper/lower bounds U
4 limited output u < U
5 limited
6 safe
7 unsafe
8 input (y)
9 output (u)
10 unsafe
11 constrained

Figure 7 — Safer application of AI technology to control: a) through supervisory constraints on discrete outputs. Or b) on continuous control output through barrier certificates

10.2.4 Redundancy, ensemble concepts and diversity

Redundancy can be of different types: structural (spatial), time (frequency), functional (informational, analytical), or combined (see References [64], [65], [70], [71], [155]). When using neural networks for example, redundancies for AI technologies include:

- Using analytical redundancy (see Reference [64]): Quantitative model-based failure detection and isolation (FDI) methods rely on the comparison of a system's available measurements, with a-priori information represented by the system's mathematical model. There are two main trends of this approach, namely analytical redundancy or residual-generation methods and parameter estimation.
- Time-redundant multiple computation (see Reference [65]): For example, concurrent error correction is achieved by using time redundancy based on recomputing with triplication with voting (RETWV).
- N-version programming (see Reference [66]): In this method, several simplex models are trained independently, such that these models are unlikely to produce erroneous results for the same test cases. In this way, it is possible to design a fault-tolerant system whose output is determined by all these models cooperatively.
- Using redundant deep architectures (see Reference [67]).
- The ensemble use of neural networks to build reliable classifiers (see Reference [68]): The idea is to combine several "weak" classifiers to obtain a "strong" one, so that the classifier still works reliably if one of its members fails.
- Use of algorithm-based fault tolerance for neural networks (see Reference [69]).

Methods of metrological self-check (including self-validation, self-diagnosis) that have found application in control systems for critical equipment, such as References [70]–[74], [155] are also considered and adapted for use with AI technology.

For higher effectiveness, redundancy is combined with diversity to reduce the likelihood of systematic failures during development. This is related to multiple AI technologies exhibiting the same behaviour, but implemented:

- by different teams;
- using separate labelling rules;
- using different problem formulations;
- using different training data;
- executing on diverse hardware (also valid for non-AI technology specific failure modes);
- with diversity of sensing;
- with diversity of self-check or self-validation methods;
- with diversity of AI technology itself.

Due to its complex and indefinite nature, diversity is expressed by a multitude of metrics. These metrics answer the following basic questions: to what extent AI technologies with the same training conditions differ in their performance and robustness? Are diversity metrics suitable for selecting members to form a more robust ensemble?

Another possible approach is to identify and eliminate false detections by comparing key point decisions from different neural networks[76]. This form of diverse comparison is often combined with monitoring (see Reference [77]).

When relying on redundancy as part of a safety argument and considering the explainability of DNNs, it is possible to rely on an analytical argument for freedom from common cause failures. In this case it is relevant to base the argument on verification and validation and demonstrate through simulation the absence of common cause failures between redundant networks.

10.2.5 AI system design with statistical evaluation

Once trained many AI systems are deterministic in inference, however where the input dimension is high and continuous in value the performance of these systems is typically characterized statistically.[156][157] For example, cross fold validation of a neural network can produce different performance metrics depending on the changing allocation samples of training data on each validation 'fold'. For a specific application under specified operating conditions, a system containing AI technology is evaluated for functional safety with consideration of the statistical distribution of its output. While no hard upper bound for error rate is obtained, for a given input distribution, a statistical confidence interval on a maximum error is determined. A key assumption is that these statistics rely on the distribution of testing data being sufficiently similar to the production data.

If some events are low probability, but have high impact, it is appropriate to increase the incidence of these events in the input data, making the frequency of events in the input proportional to the risk, rather than the actual probability of occurrence.

A typical assessment approach is based on the following steps:

- analysing the AI technology, e.g. the ML model;
- treating the AI system as a normal mathematical model, but only with probabilistic behaviour.

The more flexible the model, the more complicated its analysis is. An example of a comparable analysis is found in Reference [24].

The use of this statistical information is considered in the justification for Class II and Class III AI technologies.

10.3 Increase the reliability of components containing AI technology

10.3.1 Overview of AI component methods

As a complement to architectural considerations in 10.2, this clause identifies AI supporting technologies to increase the reliability of trained systems when deployed. Subclause 10.3.2 provides examples of methods that make AI technology less sensitive to intended and unintended input data perturbations. Simplification of trained networks is a class of methods to remove dormant or unused element of a network in 10.3.3 while 10.3.4 uses attention analysis methods to identify risks in the learnt structures. Subclause 10.3.5 describes the mechanisms to protect the input and model data during training and run-time.

10.3.2 Use of robust learning

To improve robustness against disturbances of noise, device failures and possibly malicious (adversarial) inputs, several methods are used at both testing and learning stages. Possibly applicable techniques include:

- Regularization is a methodology to mitigate the over-fitting problem, and thus to improve stability. This technique is considered analogous to the methods used in regression fitting, where the weight magnitude or non-zero values in the training loss function is penalised or given a prior distribution. This is generally preferred to post training pruning of low valued weights. Alternative methods include structuring the network to share weights on node connections (e.g. on repeated filter elements in a CNN), i.e. to simplify the structure of the model. Dropout is often considered good practice to reduce overfitting in DNNs, at the cost of an additional “dropout rate” hyperparameter. This technique randomly turns off parts of the network for a small proportion of training. Since the network cannot exclusively rely on a single node to model a particular data feature, the dropped-out regions do not overspecialize. See References [80], [103], [104].

- When the AI system disturbances are predictable (e.g. for hardware errors), fault-aware training that includes error modelling during neural network training, makes neural networks more resilient to specific fault models on the device (see Reference [81]).
- Adversarial robust training is a learning method that minimises or limits the worst-case error under the training data augmented by a model of an attacker's possible perturbations. The simultaneous maximization of adversarial perturbation effect and minimization of error leads to the extension of standard gradient descent training algorithms, (see Reference [105]). Other approaches provide robustness guarantees for output invariance, e.g. formally proving that no change in classification occurs when perturbations are within given bounds. See Reference [106]. However, scaling these guarantees to large scale and heterogenous networks remains a challenge.
- Randomization approaches, such as randomized smoothing, provides an efficient equivalent to training multiple modes with data augmented with randomized noise, so as to calculate the final result value to be a mean with respect to the noise distribution. See Reference [107].
- Robustness to out of distribution input is also considered for applications subject to limited training data or data drift or concept drift. Data augmentation and enrichment reduces the distances that the AI system needs to extrapolate from training data. For example, higher performance is obtained if images are translated and rotated in the training data. This richer learning often has the complementary effect of increasing robustness. See Reference [108].

10.3.3 Optimization and compression technologies

Optimization and compression technologies, such as quantisation of parameters and computations (i.e. reduction of parameter bandwidth), pruning (i.e. removal of less important parameters from the model) and knowledge distillation to simpler surrogate models provide secondary benefits to the system. As with all modifications to a system, careful analysis of the risks of performance loss are undertaken. See References [133], [134].

Reducing input dimension though non-AI techniques (linear and nonlinear principal components, clustering, feature extraction, etc.), risks permanently discarding useful information. A potential alternative, often used in modern network designs, is to employ embedding layers (e.g. convolutional layer or low-dimensional full-connected nodes). This results in downstream simplification and also often improves training performance.

Simplified models that have a reduced dimensionality of weights (and perhaps inputs) make training easier and reduce the risk of non-convexity (and, this, multiple local minima) in the loss landscape.[154] As well as capacity for convergence improvements, reduced network dimensions intuitively make interpretability more tractable. However, the reduced network dimensions still exceed the capacity to understand the function of each parameter in relation to its contribution to requirement satisfaction (i.e. its traceability). Emerging visualization methods have shown promise, particularly for image classification, but completeness is not yet proven (see Reference [110]). Modularising the network is another pragmatic way to help understand its traceability and ease verification (including potential to make formal verification approaches computationally feasible).

Knowledge distillation was originally designed to create simpler surrogates and more computationally tractable models. The concept produces a secondary simpler model trained on the output of a larger model rather than on training data. A complex model typically creates an embedding of lower dimension than the raw data, and once learned is often modelled with the secondary linear models with negligible loss of performance and the advantage of explainability. Nonlinear secondary models contribute less to interpretability but aid in smoothing gradients. Smoother gradients provide gradient masking protection against adversarial attacks, making the change in output smaller for a given input perturbation. This is achieved by creating probabilistic labels in a first training path with the complex model and then retaining a simpler model with these ”non-crisp” probabilistic labels (see Reference [109]).

Network neuron pruning defends against training-time attacks. One approach to network pruning is achieved by post-training analysis of the neuron activation with clean inputs, iteratively removing those that have low activation and retesting. This reduces the risk of operational discovery of unwanted behaviours.

10.3.4 Attention mechanisms

The aim of the attention mechanism (see Reference [111]) is to improve the prediction performance in sequence-to-sequence models such as language translation models, speech-to-text converters and image captioning models. There are several considerations on attention mechanisms:

— Attention mechanism to learn global context: The attention mechanism learns the relationship between a sequence of features (e.g. words in a sentence) using a weighted combination of all encoded input vectors. Similarly, in machine vision applications, attention weights learn a global weighting over the entire image to solve more complex tasks, such as image captioning (see Reference [112]), of which convolution layers are not capable. Later, image transformers (see Reference [113]) have been described to capture the context in images without any sequential data (e.g. text) available for training. Lastly, the attention mechanism is being used as a suitable solution for training models on multi-domain data, especially combinations of sequential and spatial data.

— Post-hoc attention maps for sanity checking and feature manipulation: Attention map, also known as saliency map (an explanation method used for interpreting the predictions of CNNs) or sensitivity map, is a common type of machine learning explanation to point out the most important feature in a given prediction. An attention map is a type of local explanation that is limited to individual model predictions, regardless of overall model behaviour, but still suitable for investigating the edge cases for model debugging. Attention maps are obtained in different ways, such as local approximation of deep models (see References [114], [115], [116]) using shallow interpretable models. To generate saliency maps for DNNs, various gradient-based (see References [118], [119]), convolution-based (see References [120], [121]), deconvolution-based and perturbation-based (see Reference [122]) methods have been described. Note that attention map explanations are either post-hoc or integrated with the network (see Reference [123]).

— Benefits of attention maps: Reviewing machine learning explanations has benefits for designers to improve a given model during multiple stages of the machine learning life cycle. For example, identifying issues in model structure (see Reference [122]), feature-based engineering (see References [116], [117]) and training data improvement. Additionally, assuming that model explanations are consistent with the end-user reasoning and understanding of data, human review of attention maps promotes building an appropriate level of trust of autonomous systems.

— Trainable attention: Trainable attention mechanisms have attention weights that are learned during training to improve attention efficiency. For example, Reference [114] uses a multiple attention-estimator module for different network layers to encourage more refined attention maps and higher prediction performance. Explicit human supervision (e.g. gaze tracking) for attention models has been also investigated in Reference [115] but carries high data annotation costs.

— Explanation truthfulness: Since model explanations are always incomplete explanations of black box models, the correctness and completeness of explanations is greatly influenced by factors like the heuristic technique, input example, and training data size and quality.

10.3.5 Protection of the data and parameters

Data and model parameters are potentially vulnerable to random and intentional disturbances and loss, with causes from hardware failure to data poisoning in adversarial attacks. As with all data used in a system, the use of data risk assessment and management processes (see Clause 11) helps to drive the protection measures used, with consideration for particular challenges associated with data-intensive AI technology (e.g. volume, variety, velocity, variability).

Information assurance of data used for machine learning is considered by bodies such as National Institute of Standards and Technology (see for example Reference [140]). Configuration control of data is maintained throughout model life cycle, including provenance, access rights and quality metrics of the data. A configuration process similar to ISO 26262-6:2018,[13] Annex C is used. Data information assurance at run-time and during offline training considers a multitude of properties (integrity, completeness, accuracy, resolution, etc.), see Reference [141], Section 6.4, which also suggests measures to maintain these properties.

In addition to data control measures, pre-processing of the input data stream to remove unfeasible input patterns is a sensible precaution. For example, filters transparent to physical system bandwidth which remove adversarial noise complements detection mechanisms avoiding out-of-distribution data.

The high computational demands of ML drives developers to use third party high-performance computing to train a model, where information assurance is a higher risk. It is not sufficient to only mitigate intellectual property leakage (e.g. encryption, obfuscation of labels and data distribution). Training data manipulation (poisoning) is designed by an attacker to circumvent local testing, where the network behaves correctly on normal test data with dormant problems (e.g. neurons not activated by normal training). Complementing pruning techniques with the (light weight) retraining on a locally protected data source helps reducing the sensitivity to adversarial examples. See Reference [124].

11 Processes and methodologies

11.1 General

From a functional safety point of view, many life cycle issues are common to AI- and non-AI systems. These commonalities are described in 11.2.

The level of functional safety a system needs to achieve is independent of whether AI technology is used or not. The parts of the system built using non-AI software approaches are handled with existing functional safety International Standards. The methodology commonly used to develop AI models has inherent gaps from a functional safety International Standards requirement perspective, and suitable means to address and resolve the “gaps” are therefore considered.

Functional safety considers safety throughout the whole life cycle of a system.

11.2 Relationship between AI life cycle and functional safety life cycle

Traditionally the term “lifecycle” has been used for several objectives. One objective is to provide a defined set of processes within a system or hardware or software life cycle, as well as to facilitate communication amongst stakeholders of that life cycle. ISO/IEC 22989:2022 describes a high-level life cycle model of AI systems while ISO/IEC 5338[1] defines the life cycle processes of AI systems. Software life cycle processes are also described in Reference [4].

An additional objective is to identify activities at each phase throughout the whole life cycle to implement a certain level of functional safety. This is described in the IEC 61508 series[16]-[19] and other functional safety International Standards. To this end, the IEC 61508 series defines a functional safety life cycle that includes a hazard and risk analysis phase and an overall functional safety requirements allocation phase, as described in the general requirements of IEC 61508-1[16]. Additional specific requirements are given for hardware in IEC 61508-2[17] and for software in IEC 61508-3[18].

In this document, the view is taken that it is reasonable to start from a traditional functional safety life cycle and to modify and adapt the functional safety life cycle to take into account AI system-specific issues that affect functional safety. The hazard and risk analysis phase is based on the IEC 61508 series or other functional safety International Standards, modified to address the AI specific particularities listed in ISO/IEC 5338[1] as properties important for functional safety (see Clause 8).

The IEC 61508 series and other functional safety International Standards mention the V-model as the basis of the life cycle, although certain International Standards (including the IEC 61508 series[16]-[19],[22] and IEC 61511-1[20]) do recognize that the life cycle or phases are tailored to the specific implementation technology.

A functional safety life cycle for the development of an AI system is selected during functional safety planning (see Figure 8).

It is acceptable to tailor the V-model for incremental development models to fit with the AI-specific particularities, for example as shown in ISO/IEC 5338.[1] Regression validation is used when performing iterative and incremental development (e.g. iterative learning cycles).

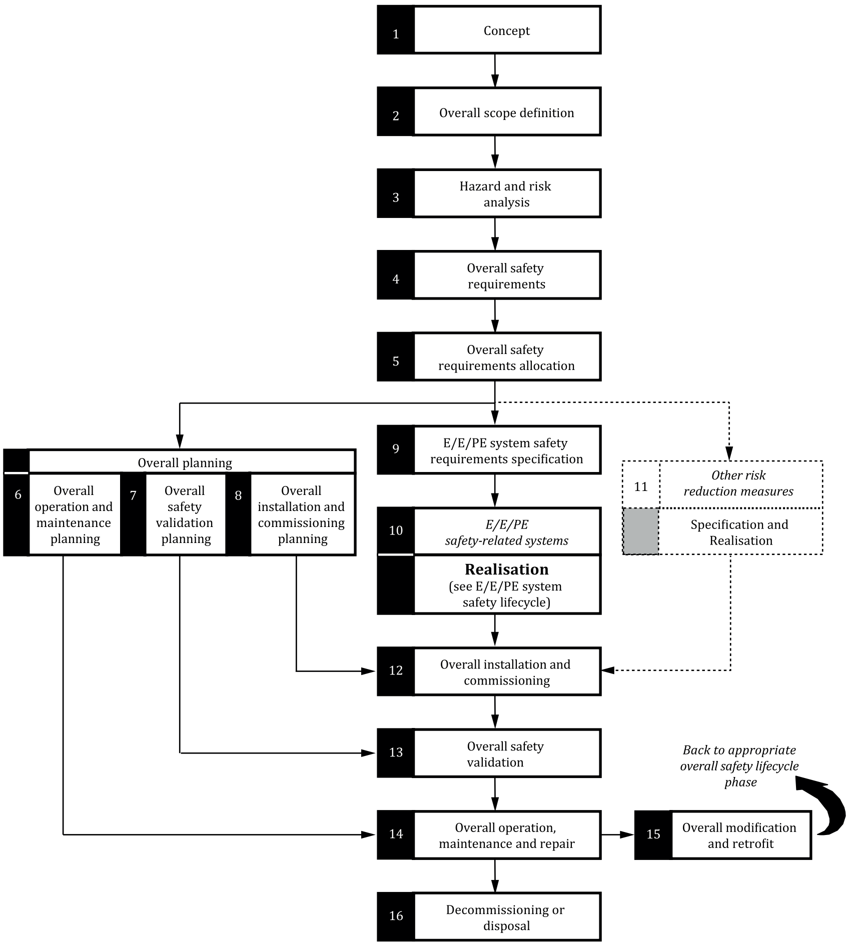

1 Concept
2 Overall scope definition
3 Hazard and risk analysis
4 Overall safety requirements
5 Overall safety requirements allocation
Overall planning
6 Overall operation and maintenance planning
7 Overall safety validation planning
8 Overall installation and commissioning planning
9 E/E/PE system safety requirements specification
10 E/E/PE safety-related systems
Realisation (see E/E/PE system safety lifecycle)
11 Other risk reduction measures
Specification and Realisation
12 Overall installation and commissioning
13 Overall safety validation
14 Overall operation, maintenance and repair
15 Overall modification and retrofit
Back to appropriate overall safety lifecycle phase
16 Decommissioning or disposal

Figure 8 — Life cycle model taken from the IEC 61508-1:2010, Figure 2

11.3 AI phases

An example of mapping between ISO/IEC 5338[1] and the IEC 61508 series is provided in Annex D.

11.4 Documentation and functional safety artefacts

Sufficient information and documentation for each phase of the functional safety life cycles contributes to subsequent phases and verification activities. It includes documenting changes to products and processes.

Issues specific to AI systems include learning processes, data relevance and sufficient documentation of training, validation and test data.

11.5 Methodologies

11.5.1 Overview

This clause describes some of the known methodologies to consider with respect to AI technologies.

11.5.2 Fault models

The concept of fault models is intended to enable systematic and possibly automated analysis of an element's behaviour in the presence of faults. The idea of fault models (fault awareness) is to cover the manifold details of reality by a sufficiently high abstraction level. This applies in particular to the area of machine learning.

A fault model is a simplifying abstraction of real effects likely to cause errors that is intended to enable a systematic analysis. Often different effects are covered by one fault. In reality, fault propagation is quite complex, but frequently different chains of propagation lead to similar errors.

When defined precisely enough, the impact of faults is simulated or analysed manually. By applying the fault model to all elements, the completeness with respect to the fault model abstraction level is ensured.

To create a fault model, a description and design of the system with the corresponding elements is required. For each of these elements the failure modes are identified, e.g. by a guide word method such as the HAZOP.

For machine learning the following aspects are relevant to fault models:

- Data sets used for training, validation and test;
- machine learning model;
- learning process;
- connection of the machine learning life cycle with the safety life cycle (also consider performing a Process FMEA).

Validation and verification aspects are discussed in Clause 9.

11.5.3 PFMEA for offline training of AI technology

FMEA is applied at the process level, the functional level or the element level, for example, it is applied during the offline training phase of the AI system.

Process FMEA (PFMEA) is used to analyse and eliminate possible sources of bias and limitation within the offline training process (e.g. during ML model training before deployment phase). Additional methods of analysis are considered, such as classification FMEA (CFMEA), which is a technique specialized to assess classification-based perception (see Reference [79]).

Annex A
(informative)

Applicability of IEC 61508-3 to AI technology elements

A.1 Overview

This annex aims to illustrate, as an example, the extent to which and the means whereby the techniques and measures listed in IEC 61508-3:2010[18] Annex A (and the relevant tables from IEC 61508-3:2010, Annex B with the descriptions from IEC 61508-7:2010 [22], Annexes B and C) are applied for the technology elements of an AI system that are compliant to current functional safety International Standards.

NOTE With respect to the classification scheme described in Clause 6, this annex applies to Class I AI technology elements, while Annex B applies for Class II elements.

A.2 Analysis of applicability of techniques and measures in IEC 61508-3:2010 Annexes A and B to AI technology elements

Tables A.1 to A.19 provide an approach to interpreting the IEC 61508-3 Annex B and Annex C tables for the technology elements of an AI system that are compliant to current functional safety International Standards.

NOTE In Tables A.1 to A.19, the references in the column entitled ‘Technique or measure’ are to the Clauses in IEC 61508-3:2010, while the “B.x.x.x”, “C.x.x.x” references in the second column of each table (with header “Ref.”) indicate detailed descriptions of techniques or measures given in IEC 61508-7:2010 [22], Annexes B and C.

Table A.1 — Interpretation of software safety requirements specification (Reference: IEC 61508-3:2010, Table A.1)

Technique or measure		Ref.	Interpretation for AI technology elements
1a	Semi-formal methods	Table A.17	There are several research papers working on this direction, see Reference [142]. Moreover, AADL[27] provides formal modelling and semantics. Its use for formal verification of certain system behaviours has been documented in literature, such as[29] and.[30] Regarding semi-formal methods for ML, just about every ML paper uses semi-formal methods to describe their architecture, in the form of block diagrams, layer descriptions and links and input flow behaviour.
1b	Formal methods	B.2.2, C.2.4	
2	Forward traceability between the system safety requirements and the software safety requirements	C.2.11	For the use case independent technology elements: applicable as for non-AI system elements. For the use case dependent technology elements, in some cases it is difficult to define a safety requirements specification for AI model (e.g. the safety goal is to detect all pedestrian on the road), but how to clearly define all possible use cases for pedestrians, (e.g. a person on a wheelchair). On the other hand, IEC TS 62988-1[143] and IEC 61496[144] for instance define a person detection function that is decomposed into software functions and traced. For instance, a certain number of pixels or measurement samples return a value with a specified tolerance. This is used regardless of underlying software technology, including AI technology.
3	Backward traceability between the safety requirements and the perceived safety needs	C.2.11	Applicable for AI system elements as well.
4	Computer-aided specification tools to support appropriate techniques or measures in Table A.1	B.2.4	Applicable for AI system elements as well.

Table A.2 — Interpretation of software design and development – software architecture design (Reference: IEC 61508-3:2010, Table A.2)

Technique or measure		Ref.	Interpretation for AI technology elements
	Architecture and design feature		
1	Fault detection	C.3.1	There are several possible methods for AI fault detection, for both runtime (inference) and offline (training), including: — Checking the operational domain for distributional shifts; — Checking for new concepts (e.g. new objects, different behaviour, new rules); — Changes occurring in the world (domain drifts, new objects, changing rules). So it is differentiated between fault detection during training and during inference.
2	Error detecting codes	C.3.2	Applicable to AI technology elements as well.
3a	Failure assertion programming	C.3.3	This is possible also for AI technology elements (see Reference [28]).
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	This is possible also for AI technology elements: monitor is either a traditionally developed mechanism or another AI technology (e.g. trained differently or implementing another AI algorithmic approach); or having a N-modular architecture with diverse DNN solving the same problem and voted. Consider not only the diversity between the software and the AI algorithm, but also the diversity between the data on which ML algorithm is trained. To consider also hardware diversity is relevant for software and it includes diversity in lower-level software implementation, diversity of compiled instruction, instruction execution, etc. Use of diverse techniques is further discussed in 10.2.4.
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	
3f	Backward recovery	C.3.6	It is also used for AI technology in principle (subject to sufficient storage state space) and increases the robustness of an AI result as well since such a methodology introduces a kind of redundancy (slight changes in the input vector).
3g	Stateless software design (or limited state design)	C.2.12	Not appropriate for AI technology elements.
4a	Re-try fault recovery mechanisms	C.3.7	It is also used for AI technology in principle (subject to sufficient storage state space) and increases the robustness of an AI technology result as well since such a methodology introduces a kind of redundancy (slight changes in the input vector).
4b	Graceful degradation	C.3.8	For AI technology elements, graceful degradation is applied in case of a lowered certainty of an output value.
5	Artificial intelligence – fault correction	C.3.9	This requirement of the IEC 61508 series is under review for future editions of IEC 61508-3 in line with the work of ISO/IEC JTC 1/ SC 42 / WG 3.

Table A.2 *(continued)*

Technique or measure		Ref.	Interpretation for AI technology elements
6	Dynamic reconfiguration	C.3.10	This requirement of the IEC 61508 series is under review for future editions of IEC 61508-3 in line with the work of ISO/IEC JTC 1/ SC 42 / WG3. There are different considerations based on the specific AI system element. For example, active learning being dynamic reconfiguration of weights due to individual robot learning, while regular updates are process managed.
7	Modular approach	Table A.19	Applicable to AI technology elements as well.
8	Use of trusted or verified software elements (if available)	C.2.10	Applicable to AI technology elements as well. To be noted that verified software it is not needed for all steps of AI model development. It is relevant for inference, but not for data collection process.
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	Applicable to AI technology elements as well.
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	Applicable to AI technology elements as well.
11a	Structured diagrammatic methods	C.2.1	Applicable to AI technology elements as well.
11b	Semi-formal methods	Table A.17	Applicable to AI technology elements as well.
11c	Formal design and refinement methods	B.2.2, C.2.4	Applicable to AI technology elements as well.
11d	Automatic software generation	C.4.6	Basic principle of software development appropriate also for AI technology elements as well.
12	Computer-aided specification and design tools	B.2.4	Applicable to AI technology elements as well.
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	Applicable to AI technology elements as well.
13b	Time-triggered architecture	C.3.11	Applicable to AI technology elements as well.
13c	Event-driven, with guaranteed maximum response time	C.3.11	Applicable to AI technology elements as well.
14	Static resource allocation	C.2.6.3	Applicable to AI technology elements as well.
15	Static synchronisation of access to shared resources	C.2.6.3	Applicable to AI technology elements as well. This is managed through the associated embedded software (e.g. runtime environment).

Table A.3 — Interpretation of software design and development – support tools and programming language (Reference: IEC 61508-3:2010, Table A.3)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Suitable programming language	C.4.5	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) while very difficult for the use case dependent elements (i.e. the models). In other words, the code running on the target still fulfils the objective of those measures that are not applicable for the rest of the AI system.
2	Strongly typed programming language	C.4.1	
3	Language subset	C.4.2	
4a	Certified tools and certified translators	C.4.3	It is difficult because Commercial Off-the-Shelf (COTS) software is typically involved. However, a distinction is made about training vs. inference. COTS like TensorFlow are used for model development and training, but TensorRT converts the models into a runtime engine for inference and it is certified.
4b	Tools and translators: increased confidence from use	C.4.4	This measure is considered for AI technology development.

Table A.4 — Interpretation of software design and development – detailed design (Reference: IEC 61508-3:2010, Table A.4)

Technique or measure		Ref.	Interpretation for AI system technology elements
1a	Structured methods	C.2.1	Also appropriate for AI technology, limited to the software aspects (i.e. the use case independent elements) and architecture (ML model architecture is usually described using diagrams, connections, etc. Modular approach is used in ML models). Rather not applicable for the data related elements.
1b	Semi-formal methods	Table A.17	
1c	Formal design and refinement methods	B.2.2, C.2.4	
2	Computer-aided design tools	B.3.5	
3	Defensive programming	C.2.5	
4	Modular approach	Table A.19	
5	Design and coding standards	C.2.6 Table A.11	Design standards are applicable to AI technology elements as well. Coding standard (white box approach) is applicable for use case independent elements (e.g. CUDA C++ libraries) while very difficult for the use case dependent elements (i.e. the models).
6	Structured programming	C.2.7	
7	Use of trusted or verified software elements (if available)	C.2.10	Applicable to AI technology elements as well.
8	Forward traceability between the software safety requirements specification and software design	C.2.11	Applicable to AI technology elements as well.

Table A.5 — Interpretation of software design and development – software module testing and integration (Reference: IEC 61508-3:2010, Table A.5)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Probabilistic testing	C.5.1	Applicable to AI technology elements as well. AI technology learns by available data: given that it is obvious that data are suitable for the desired task (in terms of amount and distribution). Attributes include: — definition of target probability; — definition of test set used for measuring the actual probability; — systematic specification of the test set (aiming for completeness to achieve the desired task, but also considering unintended behaviour).
2	Dynamic analysis and testing	B.6.5 Table A.12	Applicable to AI technology elements as well.
3	Data recording and analysis	C.5.2	Applicable to AI technology elements as well. Scope for AI: Data Engineering (e.g. Setup, Management, Specification of Training, Validation and Test Data sets)
4	Functional and black box testing	B.5.1 B.5.2 Table A.13	Applicable to AI technology elements as well.
5	Performance testing	Table A.16	Applicable to AI technology elements as well.
6	Model based testing	C.5.27	Applicable to AI technology elements as well.
7	Interface testing	C.5.3	Applicable to AI technology elements as well.
8	Test management and automation tools	C.4.7	Applicable to AI technology elements as well.
9	Forward traceability between the software design specification and the module and integration test specifications	C.2.11	Applicable to AI technology elements as well.
10	Formal verification	C.5.12	Some level is possible, but hardly possible for the whole AI system.

Table A.6 — Interpretation of programmable electronics integration (hardware and software) (Reference: IEC 61508-3:2010, Table A.6)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Functional and black box testing	B.5.1 B.5.2 Table A.13	Applicable to AI technology elements as well.
2	Performance testing	Table A.16	Applicable to AI technology elements as well.
3	Forward traceability between the system and software design requirements for hardware and software integration and the hardware and software integration test specifications	C.2.11	Applicable to AI technology elements as well.

Table A.7 — Interpretation of software aspects of system safety validation (Reference: IEC 61508-3:2010, Table A.7)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Probabilistic testing	C.5.1	See Table A.5, row 1.
2	Process simulation	C.5.18	Applicable to AI technology elements as well.
3	Modelling	Table A.15	Applicable to AI technology elements as well.
4	Functional and black box testing	B.5.1 B.5.2 Table A.13	Applicable to AI technology elements as well.
5	Forward traceability between the software safety requirements specification and the software safety validation plan	C.2.11	Applicable to AI technology elements as well.
6	Backward traceability between the software safety validation plan and the software safety requirements specification	C.2.11	Applicable to AI technology elements as well.

Table A.8 — Interpretation of modification (Reference: IEC 61508-3:2010, Table A.8)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Impact analysis	C.5.23	Applicable to AI technology elements as well. Addition: Impact analysis considers the applicability of an AI element in the operating context to which it is integrated. Change management planning considers all foreseeable trigger events that can imply a change, such as explicitly planned continuous changes, changes due to detected anomalies, or changes due to aging of demands. Since changes are already foreseen during development, change management is explicitly considered in the safety planning already (e.g. by defining a model change protocol and defining the actions to be performed in such a case). Events that can trigger change are also considered.
2	Reverify changed software module	C.5.23	Applicable to AI technology elements as well.
3	Reverify affected software modules	C.5.23	Applicable to AI technology elements as well.
4a	Revalidate complete system	Table A.7	Also appropriate for AI technology elements as well depending on the impact of a change.
4b	Regression validation	C.5.25	Applicable to AI technology elements as well.
5	Software configuration management	C.5.24	Applicable to AI technology elements as well.
6	Data recording and analysis	C.5.2	Applicable to AI technology elements as well.
7	Forward traceability between the Software safety requirements specification and the software modification plan (including reverification and revalidation)	C.2.11	Applicable to AI technology elements as well.
8	Backward traceability between the software modification plan (including reverification and revalidation) and the software safety requirements specification	C.2.11	Applicable to AI technology elements as well.

Table A.9 — Interpretation of software verification (Reference: IEC 61508-3:2010, Table A.9)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Formal proof	C.5.12	Some level is possible, but hardly possible for the whole AI application (due to the size of executable code, formal analysis works only for portions of the code)
2	Animation of specification and design	C.5.26	Applicable to AI technology elements as well.
3	Static analysis	B.6.4 Table A.18	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) while it can be more difficult for the use case dependent elements (i.e. the models). The expressiveness is not the same as in traditional code.
4	Dynamic analysis and testing	B.6.5 Table A.12	
5	Forward traceability between the software design specification and the software verification (including data verification) plan	C.2.11	Applicable to AI technology elements as well.
6	Backward traceability between the software verification (including data verification) plan and the software design specification	C.2.11	Applicable to AI technology elements as well.
7	Offline numerical analysis	C.2.13	Applicable to AI technology elements as well.

Table A.10 — Interpretation of functional safety assessment (Reference: IEC 61508-3:2010, Table A.10)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Checklists	B.2.5	Applicable to AI technology components as well, specialities of AI are addressed
2	Decision tables and truth tables	C.6.1	Applicable to AI technology elements as well.
3	Failure analysis	Table A.14	Applicable to AI technology elements as well.
4	Common cause failure analysis of diverse software (if diverse software is actually used)	C.6.3	Also appropriate for AI on system level.
5	Reliability block diagram	C.6.4	Applicable to AI technology elements as well.
6	Forward traceability between the requirements of Clause 8 and the plan for software functional safety assessment	C.2.11	Applicable to AI technology elements as well.

Table A.11 — Interpretation of design and coding standards (Reference: IEC 61508-3:2010, Table B.1)

Technique or measure		Ref.	Interpretation for AI system technology elements
1	Use of coding standard to reduce likelihood of errors	C.2.6.2	These measures are applicable for use case independent elements (e.g. CUDA C++ libraries) while very difficult for the use case dependent elements (i.e. the models). Furthermore, some of the IEC 61508-3 requirements (e.g. 2, 3a, 3b) are sometimes not suitable for state-of-the-art software development like object-oriented programming languages.
2	No dynamic objects	C.2.6.3	
3a	No dynamic variables	C.2.6.3	
3b	Online checking of the installation of dynamic variables	C.2.6.4	
4	Limited use of interrupts	C.2.6.5	
5	Limited use of pointers	C.2.6.6	
6	Limited use of recursion	C.2.6.7	
7	No unstructured control flow in programs in higher level languages	C.2.6.2	
8	No automatic type conversion	C.2.6.2	

Table A.12 — Interpretation of dynamic analysis and testing (Reference: IEC 61508-3:2010, Table B.2)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Test case execution from boundary value analysis	C.5.4	Applicable to AI technology elements as well.
2	Test case execution from error guessing	C.5.5	Applicable to AI technology elements as well.
3	Test case execution from error seeding	C.5.6	Applicable to AI technology elements as well.
4	Test case execution from model-based test case generation	C.5.27	Applicable to AI technology elements as well.
5	Performance modelling	C.5.20	Applicable to AI technology elements as well.
6	Equivalence classes and input partition testing	C.5.7	Applicable to AI technology elements as well.
7a	Structural test coverage (entry points) 100 %	C.5.8	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) as also for the code describing the model, even if the expressiveness is not the same as in traditional code. However, it can be difficult to achieve adequate test coverage of the input space.
7b	Structural test coverage (statements) 100 %	C.5.8	
7c	Structural test coverage (branches) 100 %	C.5.8	
7d	Structural test coverage – modified conditions and decisions, (Modified condition/decision coverage – MC/DC) 100 %	C.5.8	

Table A.13 — Interpretation of functional and black box testing (Reference: IEC 61508-3:2010, Table B.3)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Test case execution from cause consequence diagrams	B.6.6.2	Applicable to AI technology elements as well.
2	Test case execution from model-based test case generation	C.5.27	Applicable to AI technology elements as well.
3	Prototyping or animation	C.5.17	Applicable to AI technology elements as well.
4	Equivalence classes and input partition testing, including boundary value analysis	C.5.7 C.5.4	Applicable to AI technology elements as well.
5	Process simulation	C.5.18	Applicable to AI technology elements as well.

Table A.14 — Interpretation of failure analysis (Reference: IEC 61508-3:2010, Table B.4)

Technique or measure		Ref	Interpretation for AI system technology elements
1a	Cause consequence diagrams	B.6.6.2	Applicable to AI technology elements as well. Failure analyses also considers data engineering aspects.
1b	Event tree analysis	B.6.6.3	
2	Fault tree analysis	B.6.6.5	
3	Software functional failure analysis	B.6.6.4	

Table A.15 — Interpretation of modelling (Reference: IEC 61508-3:2010, Table B.5)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Data flow diagrams	C.2.2	Applicable to AI technology elements as well.
2a	Finite state machines	B.2.3.2	Applicable to AI technology elements as well.
2b	Formal methods	B.2.2, C.2.4	Applicable to AI technology elements as well.
2c	Time Petri nets	B.2.3.3	Applicable to AI technology elements as well.
3	Performance modelling	C.5.20	Applicable to AI technology elements as well.
4	Prototyping or animation	C.5.17	Applicable to AI technology elements as well.
5	Structure diagrams	C.2.3	Applicable to AI technology elements as well.

Table A.16 — Interpretation of performance testing (Reference: IEC 61508-3:2010, Table B.6)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Avalanche or stress testing	C.5.21	Applicable to AI technology elements as well.
2	Response timings and memory constraints	C.5.22	Applicable to AI technology elements as well.
3	Performance requirements	C.5.19	Applicable to AI technology elements as well.

Table A.17 — Interpretation of semi-formal methods (Reference: IEC 61508-3:2010, Table B.7)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Logic or function block diagrams	See IEC 61508-3:2010, Table B.7 Note 1	Applicable to AI technology elements as well.
2	Sequence diagrams	see IEC 61508-3:2010, Table B.7 Note 1	Applicable to AI technology elements as well.
3	Data flow diagrams	C.2.2	Applicable to AI technology elements as well.
4a	Finite state machines or state transition diagrams	B.2.3.2	Applicable to AI technology elements as well.
4b	Time Petri nets	B.2.3.3	Applicable to AI technology elements as well.
5	Entity-relationship-attribute data models	B.2.4.4	Applicable to AI technology elements as well.
6	Message sequence charts	C.2.14	Applicable to AI technology elements as well.
7	Decision tables or truth tables	C.6.1	Applicable to AI technology elements as well.
8	Unified Modelling Language (UML)	C.3.12	Applicable to AI technology elements as well.

Table A.18 — Interpretation of static analysis (Reference: IEC 61508-3:2010, Table B.8)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Boundary value analysis	C.5.4	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) as well as for the code describing the model, even if the expressiveness is not the same as in traditional code. However, it can be difficult to achieve adequate test coverage of the input space.
2	Checklists	B.2.5	
3	Control flow analysis	C.5.9	
4	Data flow analysis	C.5.10	
5	Error guessing	C.5.5	
6a	Formal inspections, including specific criteria	C.5.14	
6b	Walk-through (software)	C.5.15	
7	Symbolic execution	C.5.11	
8	Design review	C.5.16	Applicable to AI technology elements as well.
9	Static analysis of run time error behaviour	B.2.2, C.2.4	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) as well as for the code describing the model, even if the expressiveness is not the same as in traditional code. However, it can be difficult to achieve adequate test coverage of the input space.
10	Worst-case execution time analysis	C.5.20	Applicable to AI technology elements as well.

Table A.19 — Interpretation of modular approach (Reference: IEC 61508-3:2010, Table B.9)

Technique or measure		Ref	Interpretation for AI system technology elements
1	Software module size limit	C.2.9	Those measures are applicable for use case independent elements (e.g. CUDA C++ libraries) as well as for the code describing the model, even if the expressiveness is not the same as in traditional code. However, it can be difficult to achieve adequate test coverage of the input space. Software size (considering number of code lines) is not the criteria but rather a number of parameters, such as a limited number of neural network nodes or of connections or layers. Complexity is redefined for ML. It is combined with size, or types of connectivity between layers, since DNNs do not typically have branching statements.
2	Software complexity control	C.5.13	
3	Information hiding or encapsulation	C.2.8	
4	Parameter number limit, fixed number of subprogram parameters	C.2.9	
5	One entry one exit point in subroutines and functions	C.2.9	
6	Fully defined interface	C.2.9	Applicable to AI technology elements as well.

Annex B
(informative)

Examples of applying the three-stage realization principle

B.1 Overview

This annex describes non-exhaustive examples on how to apply the classification scheme described in Clause 6 and the three-stage realization principle described in Clause 7.

B.2 Example for an automotive use case

The example described in this clause is an automotive system comprising two layers:

- the mission layer, is responsible for perceiving the environment, taking decisions including planning routes and commanding actuation including steering, braking;
- the protection layer, which provides safety functions such as identifying conditions under which to execute a protective stop or brake command.

NOTE 1 The mission layer is referred to as the "item" using ISO 26262-1[12] terminology or the "EUC control system" using IEC 61508-4[19] terminology. The protection layer is referred to as the part of the system guaranteeing the "safety goal" using ISO 26262-1 terminology or the "safety-related system" using IEC 61508-4 terminology.

It is assumed that the system includes cameras and the related data are processed by a perception algorithm based on deep learning (DL) algorithm like a DNN. An example of this type of DNN is DriveNet[91].

A typical representation of this system is shown in Figure B.1.

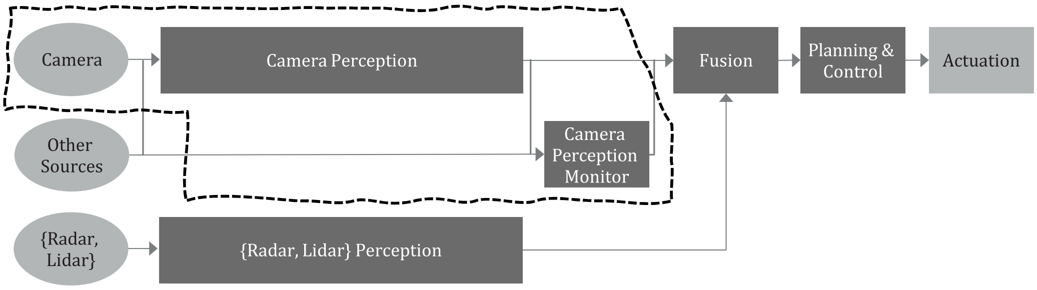


Figure B.1 — Example of an automotive system

NOTE 2 In Figure B.1 light grey boxes represent sensing inputs, actuators; dark grey represent perception related functions and related monitors.

The scope of the example is limited to the area outlined by the dashed line in Figure B.1, i.e. the camera perception DNN, the related sensing path (i.e. the camera) and related monitors. The other perception paths (lidar, radar) that is involved in the system, the related fusion and the planning and actuation functions are not in scope.

The AI technology used in this system is considered of a Usage Level A1 as described in 6.2, because it is used in a safety relevant E/E/PE system and automated decision-making of the AI system is possible. Based on the principles described in Clause 8, the following properties are identified for this use case:

- Specifiability: How to specify pedestrian appearance in an image?

- Interpretability: How to get insight into design?
- Generalization: Can the DNN interpolate across input domain?
- Domain shift: Is the DNN operating in training data domain?
- Robustness-safeness: Can small perturbations (malicious or not) change output?
- Diversity: What does diversity mean in the context of DL and how to ensure that diversity is sufficient (e.g. different DL architectures, different training data sets)?
- Confidence: How to consider confidence levels in the context of DL?

These properties are mapped to the three stages of the realization principle as shown in Table B.1.

Table B.1 — Mapping of properties to the realization principle stages

	Acquisition from inputs and data	**Knowledge induction from training data and human knowledge**	**Processing and generation of outputs**
Specifiability	-	X	X
Interpretability	-	-	X
Generalization	-	-	X
Domain shift	-	X	X
Robustness-safeness	X	-	X
Diversity	X	X	X
Confidence	-	-	X

The AI technology used in this system is considered of a Class II, because, as shown in Table B.2, it is still possible to identify a set of available methods and techniques satisfying the properties (e.g. it is still possible to use certain compensation methods of verification and validation), so that the AI technology meets outlined criteria and the development follows suitable processes.

Table B.2 provides an example of the analysis of the properties in the applicable stages of the three-stage realization principle, and identifies the topics, the KPIs and the available techniques and measures to satisfy those properties.

Table B.2 — Example property analysis

Stage: knowledge induction from training data and human knowledge **Desirable property**: Specifiability			
Topic	**Details**	**KPI**	**Available methods with references**
specification of the data set	— amount of data. — type of data needed (e.g. object classes, object data definition, weather conditions, geographic domain, background scene). — division of data between training, validation and testing.	— data set coverage. — data set distribution. — example: the data set contains images acquired for different road types during differing weather conditions and the data acquisition takes place during daytime.	— manual curation. — active learning.
specification of labelling policy	— data annotation. — treatment of occluded objects. — number of annotators annotating the same data.	— labelling quality distribution. — example: the road lane boundaries are marked pixel by pixel. Each image is annotated by two independent annotators. The amount of 10 % of randomly selected data are additionally annotated by a third annotator.	

B.3 Example for a robotics use case

The example described in this clause is an autonomous mobile platform that transports materials around an industrial warehouse.

The system is separated between application and safety domain:

- The application domain includes wireless communication with the fleet management system to receive tasks and updates, and local on-device software for carrying out the tasks (localization, navigation, mapping).
- The safety domain provides safety functions such as emergency stop, protective stop, speed limitation and muting.

The overall system is classified as a driverless industrial truck under ISO 3691-4[145] and industrial mobile robot (type B) under ANSI/RIA R15.08-1[146].

The scope of the example is limited to the implementation of the protective stop safety function, as this is the only safety function that utilizes machine learning. It is assumed that each safety function is independent of each other, and that the application domain is isolated from the safety domain.

A simplified representation of this system, limited to the components relevant to the protective stop safety function, is shown in Figure B.2. Camera sensors around the robot provide images to a neural network, which produces a depth image. The depth image is converted into a 3D view of the scene. A check is made to see if a safety violation occurs. If so, a protective stop output is sent to the motor. While additional sensors are shown on the robot, it is assumed the safety function is implemented on each sensor independently (rather than a "system of systems").

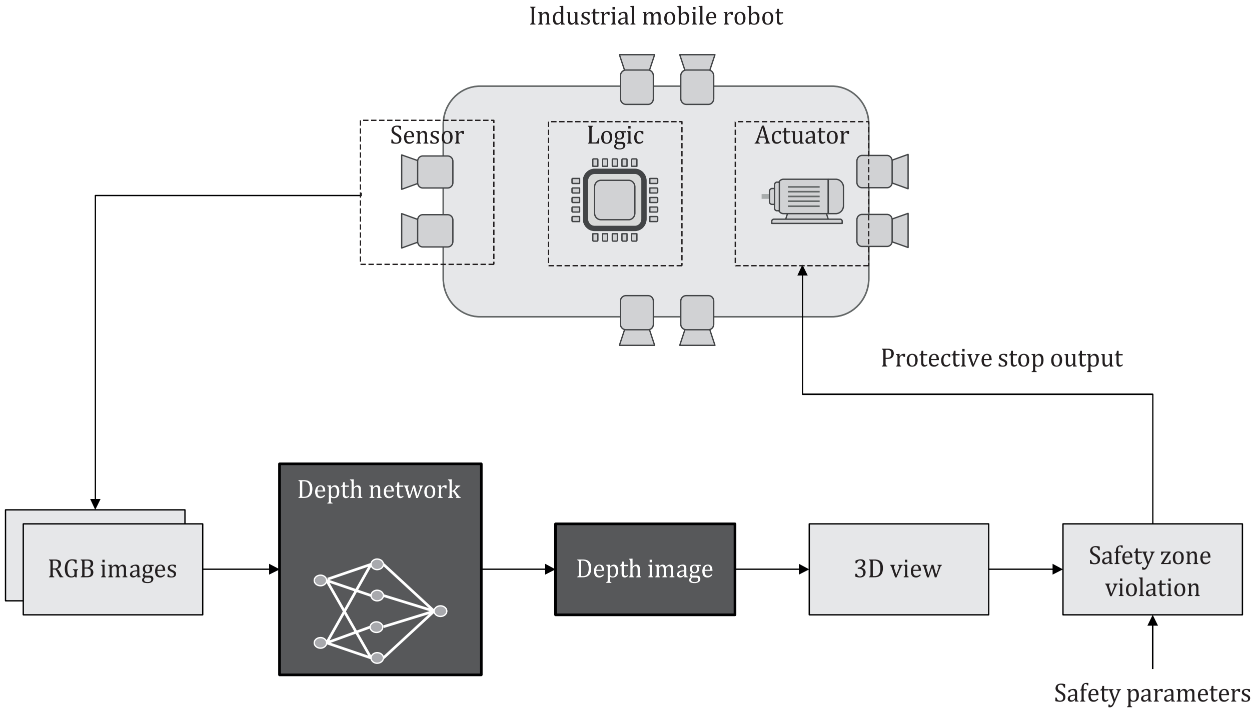


Figure B.2 — Example of industrial mobile robot

The software components in dark grey with bold black outline (i.e. depth network and depth image) represents logic and outputs directly produced by the machine learning model. All other components in grey are not within the scope of this document because they is validated using existing International Standards. The dark grey components are considered Usage Level A1 as described in 6.2, since they are used in a safety relevant E/E/PE system and automated decision-making of the AI is possible.

Based on the principles described in Clause 8, the properties addressed by the AI components are:

- Specifiability: What are the requirements of the network? How do those requirements map to existing International Standards for safety sensors, such as IEC 61496-1[144] and IEC TS 62998-1[143]? What constitutes the training images for the neural network, how are those images mapped to the operating environment? How many images, across different classes, are sufficient for training?
- Domain shift: What if the deployment environment is different from the environment used during training?
- Verifiability: How is the neural network performance assessed? How does this assessment map to existing International Standards for safety sensors, such as IEC 61496-1[144] and IEC TS 62998-1[143]?
- Robustness: How robust is the neural network to perturbation of the input data due to different causes (hardware, environmental factors, operational changes, ageing, etc.)?
- Interpretability: Are the results produced by the network understandable? Do the produced results correspond to the expected results, as defined by the safety requirements?
- Transparency: Are the components that make up the machine learning model understood? Is there a reason for design choices? Do those choices map to input requirements?

Table B.3 maps the properties to the three-stage realization principle of Clause 7.

Table B.3 — Mapping of properties to the realization principle stages

	Acquisition from inputs or data	Knowledge induction from training data and human knowledge	Processing and generation of outputs
Specifiability	X	X	X
Domain shift	-	X	X
Verifiability	-	X	X
Robustness	X	-	X
Interpretability	-	X	X
Explainability	-	X	-

Table B.4 provides an example analysis from the third stage of the three-stage realization principle in relation to the verifiability property.

Table B.4 — Example property analysis

Stage: processing and generation of outputs		
Desirable property: verifiability		
Topic	**Details**	**Compliance criteria**
How is the neural network performance assessed?	— For a given input, definition of what constitutes a “correct” output by the network. — Definition of what range of inputs is evaluated.	— Pixel level KPIs. — Image level KPIs. — Sequence level KPIs. — Data set level KPIs.
How does the network performance map to existing safety International Standards and metrics?	— Mapping of measured network performance criteria to existing standard criteria. — Requirement tracing from standards performance requirements to network requirements.	— Requirement traceability or mapping documents.
How to determine verification process is accurate (e.g. unexpected behaviour due to combination of factors that cannot be seen by testing across single factors)?	— Single-dimensional vs multi-dimensional testing. — Statistical analysis of random variate testing. — Independent verification process. — Evaluation and or certification of verification tools	— Test plans. — Independently reviewed results. — Statistical analysis. — Tool qualification — Process FMEA.
How to determine when verification is complete?	— Amount of verification data — Type of verification data and how it’s split into relevant parameters — Frequency with which verification is carried out — Frequency with which verification data are refreshed — Stopping criteria for verification	— Test plans — Predetermined stopping criteria — Process FMEA

The same analysis is applied to all other identified properties in Table B.3, identifying a set of available methods and techniques to satisfy the property. As such, the AI technology used in this system is considered Class II as defined in 6.2.

Annex C
(informative)

Possible process and useful technology for verification and validation

C.1 General

This annex complements the content of 9.3 by providing examples of possible process and technical methods. The content is mainly quoted with rearrangement from Reference [94].

C.2 Data distribution and HARA

According to Reference [94], to keep sufficient levels of relation between data distributions and hazard and risk analysis as described in 9.3.2 and 9.3.3 a), the following measures are described in literature.

- For low-level requirements:
  - Examination and recording of the major cause of possible deterioration of safety.
  - Based on the examination results, design of the data and reflection of those in applicable attributes.
- For middle-level requirements, in addition to those described for low-level requirements:
  - Analysis of risks of deterioration of quality in use in overall system and their impact with a certain level of engineering coverage and record the results in documents.
  - Analysis if any measure applies to any of those risks, and analysis of attributes related to the risk that are contained in an input to machine learning components.
  - Analysis and recording of the application-specific characteristics of environments that generate machine learning input, with regards to the difficulty for machine learning and other aspects.
  - Examination of sets of attributes and attribute values, based on the results of those analysis and record the background of such decisions.
- For high-level requirements, in addition to those described for low-level and middle-level requirements:
  - Investigation of documents on their own past examination results and those of others with regard to elements to be extracted as characteristics of system environment and recording of the background of examinations leading to the extraction of applicable subsets.
  - Investigation of past examination results in line with application fields of systems with regard to deterioration risks of qualities in use of overall systems and recording of the examination results including the background of selection.
  - Moreover, extraction of deterioration risks of qualities in use of overall systems using engineering analysis such as fault tree analysis and record their results.

For identifying attribute sets related to identified risk, example aspects to consider include:

- Basic knowledge on the existing functional safety design, the existing hazard lists in each application domain, or brainstorming results based on existing International Standard hazard list, for example in ISO 12100[3] or one from NASA[126].
- Analysis of cases on prior systems and similar machine learning based systems.
- Domain knowledge by brainstorming with users.

- Knowledge about data employed preliminarily and exceptional cases identified in the trials of the training in the proof of concept stage.

C.3 Coverage of data for identified risks

According to Reference [94], one of the considerations is having enough and well-diverse data for each identified risk. If the subset of data related to a specific risk is determined, it is relatively easy to check the distribution of the attribute values. However, as the number of attributes can become intractable in typical machine-learning use cases, checking for all combinations of attribute values is likely to be unrealistic. Some kind of coverage metric to estimate the likelihood that all important cases have been addressed is described as alternative in the Reference [94]. In the area of software testing study, several metrics for test coverage are used; one possible solution is to reuse such a concept for defining the coverage, especially a concept of combinatorial testing.

The explainability features of the model is used to check completeness of data, in addition to checking against specifications.

In this way, some known measures for this subproblem are:

- For low-level requirements:
  - Ensuring having some inputs for each of attributes corresponding to major risk factors.
  - ensuring some input for corresponding to combinations of composite risk factors.
  - extracting attributes of differences in particularly relevant environmental factors and preparing data corresponding to combinations with serious risk factors.
- For middle-level requirements, in addition to those described for low-level requirements:
  - Ensuring data corresponding to particularly relevant risk factors to satisfy, in principle, the International Standards for pair-wise coverage.
- For high-level requirements, in addition to those described for medium-level and low-level requirements:
  - Based on engineering consideration, setting standards for coverage of attributes and establishing sets of combinations of attribute values that satisfy standards for coverage.
  - The level of strictness of the standards for coverage (pair-wise coverage, triple-wise coverage, etc.) required, taking into account system usage and risk severity. Standards are set individually for each risk as appropriate.

C.4 Data diversity for identified risks

According to Reference [94], data collection process management is used to achieve diversity of data having similar attribute values, because little is known about such data diversity beforehand in usual machine-learning application. Data source and its processing is carefully examined to ensure that the data are not biased to some specific input conditions.

Some known measures for this subproblem are:

- For low-level requirements:
  - Considering the source and method of acquiring test data sets to ensure that no bias is found in application situations.
  - Extracting samples without bias from original data for each case to ensure that no bias is found.
  - Recording activities carried out to prevent bias from entering.

  - Checking that there are sufficient training data and test data for each analysed case in the training phase, validation phase, etc.
  - When sufficient training data are not acquired for any case, reviewing and loosening the coverage standards and record what is checked individually by system integration tests in line with the original standards.
- For middle-level requirements, in addition to those described for low-level requirements:
  - Estimating an approximate probability of occurrence for each attribute value or each case.
  - Checking if acquired data are not deviated from the expected distribution.
  - For each case, checking the diversity of the data subset corresponding to the case, using some numerical metrics or logical criteria. For example, in each case, when there is any attribute not included in that case, extracting the distribution related to that attribute and check if there is no significant bias.
- For high-level requirements, in addition to those described for middle-level and low-level requirements:
  - Acquiring certain indicators for coverage of data included in each case. For example, checking if there is no correlation between data other than attribute values included in combinations of cases using feature extraction or any other technique.

Refer to 9.3.6 and 9.4 for using data augmentation and simulators for expanding the data size.

C.5 Reliability and robustness

According to Reference [94], for reliability and robustness of the generated machine learning model, usual AI metrics and test data set are typically used. However, ensuring robustness in machine learning is currently a difficult problem.

The text in 9.3.3 d), gives some hints about what method is used. For applications with lower-level requirements, such general cautions are often sufficient. For those high-level requirements, some forms of numerical or formal analysis for stability is expected.

Some of the technologies known:

- For low-level requirements:
  - Regularization.
  - Cross validation.
  - Randomized training.
  - Model size exploration.
- For middle-level requirements, in addition to those described for low-level requirements:
  - Adversarial training.
  - Smoothing.
  - Adversarial example generation or detection tests.
- For high-level requirements, in addition to those described for middle-level and low-level requirements:
  - Evaluation of maximum safe radius.
  - Formally checked robust training and smoothing.

Annex D
(informative)

Mapping between ISO/IEC 5338 and the IEC 61508 series

ISO/IEC 5338[1] describes a life cycle for AI systems as a series of processes. The IEC 61508 series describes a safety life cycle as a series of phases.

NOTE IEC 61508 defines normative inputs, activities and outputs while ISO/IEC 5338, based on ISO/IEC/IEEE 15288 defines processes throughout the whole life cycle within a system.

Both International Standards define a set of processes and phases throughout the whole life cycle within a system. Both International Standards allow to tailor the sequence of execution of each process or phase and also to repeat a process or phase is possible as long as the impact of the additional work is also taken into account according all the other process or phase that are being affected by the additional work.

Table D.1 provides a mapping for the technical processes of ISO/IEC 5338[1] and the IEC 61508 series life cycle phases. Other processes of ISO/IEC 5338[1] are taken into account also, especially the risk management process.

Table D.1 — Mapping between ISO/IEC 5338 and the IEC 61508 series

Safety life cycle (IEC 61508 series)	**AI system life cycle (ISO/IEC 5338)**
	Technical processes
Concept (IEC 61508-1:2010 Figure 2: 1 Concept)	— Business or mission analysis process. — Stakeholder needs and requirements definition process.
Overall scope definition (IEC 61508-1:2010 Figure 2: 2 Overall scope definition)	— Stakeholder needs and requirements definition process. — System requirements definition process.
Hazard and risk analysis (IEC 61508-1:2010 Figure 2: 3 Hazard and risk analysis)	— Risk management process.
Overall safety requirements (IEC 61508-1:2010 Figure 2: 4 Overall safety requirements)	— Stakeholder needs and requirements definition process. — System requirements definition process.
Overall safety requirements allocation (IEC 61508-1:2010 Figure 2: 5 Overall safety requirements allocation)	— Architecture definition process. — Stakeholder needs and requirements definition process. — System requirements definition process.
Overall operation and maintenance planning (IEC 61508-1:2010 Figure 2: 6 Overall operation and maintenance planning)	— Operation process. — Maintenance process.
Overall safety validation planning (IEC 61508-1:2010 Figure 2: 7 Overall safety validation planning)	— Validation process.
Overall installation and commissioning planning (IEC 61508-1:2010 Figure 2: 8 Overall installation and commissioning planning)	— Transition process.

Table D.1 *(continued)*

Safety life cycle (IEC 61508 series)	**AI system life cycle (ISO/IEC 5338)**
System safety requirements specification (IEC 61508-1:2010 Figure 2: 9 E/E/PES System safety requirements specification)	— Stakeholder needs and requirements definition process. — System requirements definition process.
Overall installation and commissioning (IEC 61508-1:2010 Figure 2: 12 Overall installation and commissioning)	— Transition process.
Overall safety validation (IEC 61508-1:2010 Figure 2: 13 Overall safety validation)	— Validation process.
System design requirements specification (IEC 61508-1:2010 Figure 3: 10.1 E/E/PES System design requirements specification)	— Architecture definition process. — Design definition process. — System analysis process.
System safety validation planning (IEC 61508-1:2010 Figure 3: 10.2 E/E/PES System safety validation planning)	— Validation process.
System design and development (IEC 61508-1:2010 Figure 3: 10.3 E/E/PES System design and development including ASICs and software)	— System analysis process. — Implementation process.
System integration (IEC 61508-1:2010 Figure 3: 10.4 E/E/PE system integration)	— Integration process.
Software safety requirements specification (IEC 61508-1:2010 Figure 4: 10.1 Software safety requirements specification)	— Stakeholder needs and requirements definition process. — System analysis process.
Software design and development (IEC 61508-1:2010 Figure 4: 10.3 Software design and development)	— Knowledge acquisition process. — AI data engineering process. — Implementation process.
Validation plan for software aspects of safety system (IEC 61508-1:2010 Figure 4: 10.2 Validation plan for software aspects of system safety)	— Verification process. — Validation process.
Integration (hardware and software) (IEC 61508-1:2010 Figure 4: 10.4 PE Integration)	— Integration process.
Software operation and maintenance procedures (IEC 61508-1:2010 Figure 4: 10.5 Software operation and maintenance procedures)	— Operation process. — Maintenance process.
Software aspects of system safety validation (IEC 61508-1:2010 Figure 4: 10.6 Software aspects of system safety validation)	— Validation process.
System installation, commissioning, operation and maintenance procedures (IEC 61508-1:2010 Figure 3: 10.5 E/E/PE System installation, commissioning, operation and maintenance procedures)	— Transition process.
System safety validation (IEC 61508-1:2010 Figure 3: 10.6 E/E/PE System safety validation)	— Validation process. — Continuous validation process.
Overall operation, maintenance and repair (IEC 61508-1:2010 Figure 2: 14 Overall operation, maintenance and repair)	— Operation process. — Maintenance process.

Table D.1 *(continued)*

Safety life cycle (IEC 61508 series)	AI system life cycle (ISO/IEC 5338)
Overall modification and retrofit (IEC 61508-1:2010 Figure 2: 15 Overall modification and retrofit)	— Maintenance process.
16 Decommissioning or disposal (IEC 61508-1:2010 Figure 2: 16 Decommissioning or disposal)	— Disposal process.

Bibliography

[1] ISO/IEC 5338, *Information technology — Artificial intelligence — AI system life cycle processes*

[2] ISO/IEC/TS 6254:—[2], *Information technology — Artificial intelligence — Objectives and methods for explainability of ML models and AI systems*

[3] ISO 12100:2010, *Safety of machinery — General principles for design — Risk assessment and risk reduction*

[4] ISO/IEC/IEEE 12207:2017, *Systems and software engineering — Software life cycle processes*

[5] ISO 13849-1:2023, *Safety of machinery — Safety-related parts of control systems — Part 1: General principles for design*

[6] ISO/IEC/IEEE 15288:2023, *Systems and software engineering — System life cycle processes*

[7] ISO 21448:2022, *Road vehicles — Safety of the intended functionality*

[8] ISO 13849-2:2012, *Safety of machinery — Safety-related parts of control systems — Part 2: Validation*

[10] ISO/IEC/TR 24027:2021, *Information technology — Artificial intelligence (AI) — Bias in AI systems and AI aided decision making*

[11] ISO/IEC/TR 24028:2020, *Information technology — Artificial intelligence — Overview of trustworthiness in artificial intelligence*

[12] ISO 26262-1:2018, *Road vehicles — Functional safety — Part 1: Vocabulary*

[13] ISO 26262-6:2018, *Road vehicles — Functional safety — Part 6: Product development at the software level*

[14] ISO 26262-11:2018, *Road vehicles — Functional safety — Part 11: Guidelines on application of ISO 26262 to semiconductors*

[15] ISO 26262-5:2018, *Road vehicles — Functional safety — Part 5: Product development at the hardware level*

[16] IEC 61508-1:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 1: General requirements*

[17] IEC 61508-2:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems*

[18] IEC 61508-3:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 3: Software requirements*

[19] IEC 61508-4:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 4: Definitions and abbreviations*

[20] IEC 61511-1:2016, *Functional safety - Safety instrumented systems for the process industry sector — Part 1: Framework, definitions, system, hardware and application programming requirements*

[21] IEC 62061:2021, *Safety of machinery — Functional safety of safety-related control systems*

[22] IEC 61508-7:2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems — Part 7: Overview of techniques and measures*

[2] Under preparation. Stage at the time of publication: ISO/IEC CD TS 6254:2023.

[23] BS EN 50128:2011+A2:2020, *Railway applications. Communication, signalling and processing systems. Software for railway control and protection systems*

[24] Braband J., Schäbe H., On Safety Assessment of Artificial Intelligence. Available from: https://arxiv.org/abs/2003.00260

[25] Wei X., Zhang R., Liu Y., Yue H., Jingwejia Tan, Evaluating the Soft Error Resilience of Instructions for GPU Applications, 2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)

[26] Ibrahim Y., Wang H., Liu J. et al., Soft errors in DNN accelerators: a comprehensive review, Microelectronics Reliability, vol. **115**, Article ID 113969, 2020

[27] Feiler P., Gluch D., Hudak J. *The Architecture Analysis & Design Language (AADL): An Introduction*, CMU/SEI-2006-TN-011. Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, (2006)

[28] Daniel Kang, Deepti Raghavan, Peter Bailis, Matei Zaharia, Model Assertions for Debugging Machine Learning. Available from: https://ddkang.github.io/papers/2018/omg-nips-ws.pdf

[29] Hadad Abeer, Ma Chunyan, Ahmed Adeeb, 2020). Formal Verification of AADL Models by Event-B. IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.2987972

[30] Liu H., David P., Gluch, Formal verification of AADL behavior models: a feasibility investigation, ACM-SE 47: Proceedings of the 47th Annual Southeast Regional Conference, March 2009 Article No.: 36, Pages 1–6. Available from: https://doi.org/10.1145/1566445.1566495

[31] Moreno-Torres, Jose G., Raeder, Troy, Alaiz-Rodríguez, Rocío, Chawla, Nitesh V. and Herrera, Francisco, A unifying view on dataset shift in classification, Pattern Recognition, January 2012. Vol. **45**, no. 1, p. 521–530. DOI 10.1016/j.patcog.2011.06.019

[32] Storkey Amos J., *When training and test sets are different: characterising learning transfer.* In Dataset Shift in Machine Learning. MIT Press, 2009. p. 3–28

[33] Ioffe S., Szegedy C., Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167 [cs] [online]. 2 March 2015. Available from: https://arxiv.org/abs/1502.03167 arXiv: 1502.03167

[34] Garcia B. Manuel, del Campo-Ávila, José, Fidalgo, Raúl, Bifet, Albert, Gavaldà, R and Morales-Bueno, R. Early drift detection method, Fourth international workshop on knowledge discovery from data streams*.* 2006. p. 77–86

[35] Klinkenberg Ralf, Joachims Thorsten, Detecting concept drift with support vector machines, ICML. 2000. p. 487–494

[36] Gama J., Medas P., Castillo G., Rodrigues P., Learning with Drift Detection, Bazzan, Ana L. C. and Labidi, Sofiane [eds.], Advances in Artificial Intelligence – SBIA 2004 [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. p. 286–295. Lecture Notes in Computer Science. ISBN 978-3-540-23237-7. Available from: http://link.springer.com/10.1007/978-3-54028645-5_29

[37] Goldenberg Igor, Webb Geoffrey I., Survey of Distance Measures for Quantifying Concept Drift and Shift in Numeric Data. Knowl. Inf. Syst. August 2019. Vol. **60**, no. 2, p. 591–615. DOI 10.1007/s10115-018-1257-z

[38] Gama João, Žliobaitė Indrė, Bifet Albert, Pechenizkiy Mykola, Bouchachia Abdelhamid, A survey on concept drift adaptation. ACM Computing Surveys. April 2014. Vol. **46**, no. 4, p. 1–37. DOI 10.1145/2523813

[39] Doshi-Velez F., Kim B., Towards A Rigorous Science of Interpretable Machine Learning, arXiv:1702.08608 [cs, stat] [online]. 2 March 2017. Available from: https://arxiv.org/abs/1702.08608 arXiv: 1702.08608

[40] Murdoch W., James, Singh, Chandan, Kumbier, Karl, Abbasi-Asl, Reza and Yu, Bin. Interpretable machine learning: definitions, methods and applications. Proceedings of the National Academy of Sciences. 29 October 2019. Vol. **116**, no. 44, p. 22071–22080. DOI 10.1073/pnas.1900654116. arXiv: 1901.04592

[41] Zeiler M.D., Fergus R., Visualizing and Understanding Convolutional Networks. In: Fleet, David, Pajdla, Tomas, Schiele, Bernt and Tuytelaars, Tinne [eds.], Computer Vision – ECCV 2014 [online]. Cham: Springer International Publishing, 2014. p. 818– 833. Lecture Notes in Computer Science. ISBN 978-3-319-10589-5. Available from: http://link.springer.com/10.1007/978-3-319-10590-1_53

[42] Bach Sebastian, Binder Alexander, Montavon Grégoire, Klauschen Frederick, Müller Klaus-Robert, Samek Wojciech, On Pixel- Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation. Suarez, Oscar Deniz [ed.], PLOS ONE. 10 July 2015. Vol. 10, no. 7, p. e0130140. DOI 10.1371/journal.pone.0130140

[43] Selvaraju R.R., Das A., Vedantam R., Cogswell M., Parikh D., Batra D., Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization. CoRR [online]. 2016. Vol. abs/1610.02391. Available from: https://arxiv.org/abs/1610.02391

[44] Stacke K., Eilertsen G., Unger J., Lundstrom C., A Closer Look at Domain Shift for Deep Learning in Histopathology. P. 8. Available from: https://arxiv.org/abs/1909.11575

[45] Petsiuk V., Das A., Saenko K., RISE: Randomized Input Sampling for Explanation of Black box Models. arXiv:1806.07421 [cs] [online]. 25 September 2018. Available from: https://arxiv.org/abs/1806.07421 arXiv: 1806.07421

[46] Bertsimas Dimitris, Dunn Jack, Optimal classification trees. Machine Learning. July 2017. Vol. **106**, no. 7, p. 1039–1082. DOI 10.1007/s10994-017-5633-9

[47] Vidal T., Pacheco T., Schiffer M., Born-Again Tree Ensembles. arXiv:2003.11132 [cs, stat] [online]. 27 August 2020. Available from: https://arxiv.org/abs/2003.11132 arXiv: 2003.11132

[48] Lipton Z.C., The Mythos of Model Interpretability. arXiv:1606.03490 [cs, stat] [online]. 6 March 2017. Available from: https://arxiv.org/abs/1606.03490 arXiv: 1606.03490

[49] Su Jiawei, Vargas Danilo Vasconcellos, Sakurai Kouichi, One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation. 2019. Vol. **23**, no. 5, p. 828–841

[50] Madry A., Makelov A., Schmidt L., Tsipras D., Vladu A., Towards Deep Learning Models Resistant to Adversarial Attacks. arXiv:1706.06083 [cs, stat] [online]. 4 September 2019. Available from: https://arxiv.org/abs/1706.06083 arXiv: 1706.06083

[51] Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I. et al., Intriguing properties of neural networks. arXiv:1312.6199 [cs] [online]. 19 February 2014. Available from: https://arxiv.org/abs/1312.6199 arXiv: 1312.6199

[52] Kurakin A., Goodfellow I., Bengio S., Adversarial examples in the physical world. arXiv:1607.02533 [cs, stat] [online]. 10 February 2017. Available from: https://arxiv.org/abs/1607.02533 arXiv: 1607.02533

[53] Eykholt K., Evtimov I., Fernandes E., Li B., Rahmati A., Xiao C. et al., Robust Physical-World Attacks on Deep Learning Models. arXiv:1707.08945 [cs] [online]. 10 April 2018. Available from: https://arxiv.org/abs/1707.08945 arXiv: 1707.08945

[54] Goodfellow I.J., Shlens J., Szegedy C., Explaining and Harnessing Adversarial Examples. arXiv:1412.6572 [cs, stat] [online]. 20 March 2015. Available from: https://arxiv.org/abs/1412.6572 arXiv: 1412.6572

[55] He W., Wei J., Chen X., Carlini N., Song D., Adversarial Example Defenses: Ensembles of Weak Defenses are not Strong. arXiv:1706.04701 [cs] [online]. 14 June 2017. Available from: https://arxiv.org/abs/1706.04701 arXiv: 1706.04701

[56] Liao F., Liang M., Dong Y., Pang T., Hu X., Zhu J. Defense Against Adversarial Attacks Using High-Level Representation Guided Denoiser. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018

[57] Meng D., Chen H. MagNet: a two-pronged defense against adversarial examples. In: *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. 2017. p. 135–147

[58] Samangouei P., Kabkab M., Chellappa R., Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. arXiv:1805.06605 [cs, stat] [online]. 17 May 2018. Available from: https://arxiv.org/abs/1805.06605 arXiv: 1805.06605

[59] Carlini N., Wagner D., MagNet and "Efficient Defenses Against Adversarial Attacks" are Not Robust to Adversarial Examples. arXiv:1711.08478 [cs] [online]. 22 November 2017. Available from: https://arxiv.org/abs/1711.08478 arXiv: 1711.08478

[60] Xie C., Wang J., Zhang Z., Ren Z., Yuille A., Mitigating Adversarial Effects Through Randomization. arXiv:1711.01991 [cs] [online]. 28 February 2018. Available from: https://arxiv.org/abs/1711.01991 arXiv: 1711.01991

[61] Liu X., Cheng M., Zhang H., Hsieh C.-J. Towards robust neural networks via random self-ensemble. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018. p. 369–385

[62] Guo C., Rana M., Cissé M., van der Maaten L., Countering Adversarial Images using Input Transformations. arXiv:1711.00117 [cs] [online]. 25 January 2018. Available from: https://arxiv.org/abs/1711.00117 arXiv: 1711.00117

[63] Athalye A., Engstrom L., Ilyas andrew and Kwok, Kevin. Synthesizing robust adversarial examples. In: International conference on machine learning. PMLR, 2018. p. 284–293

[64] Shahbaz M. H., Amin A. A., Design of Active Fault Tolerant Control System for Air Fuel Ratio Control of Internal Combustion Engines Using Artificial Neural Networks, in IEEE Access, vol. **9**, pp. 46022-46032, 2021, doi: 10.1109/ACCESS.2021.3068164

[65] Yuang-Ming Hsu V., Piuri and E. E. Swartzlander, Time-redundant multiple computation for fault-tolerant digital neural networks, 1995 IEEE International Symposium on Circuits and Systems (ISCAS), Seattle, WA, USA, 1995, pp. 977-980 vol.2, doi: 10.1109/ISCAS.1995.519929

[66] Xu H., Chen Z., Wu W., Jin Z., Kuo S., Lyu M., NV-DNN: Towards Fault-Tolerant DNN Systems with N-Version Programming, 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Portland, OR, USA, 2019, pp. 44-47, doi: 10.1109/DSN-W.2019.00016

[67] Luo Y., Lu J., Jiang X., Zhang B. Learning From Architectural Redundancy: Enhanced Deep Supervision in Deep Multipath Encoder-Decoder Networks, in *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2021.3056384

[68] Gao Z. et al., Reliable Classification with Ensemble Convolutional Neural Networks, 2020 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Frascati, Italy, 2020, pp. 1-4, doi: 10.1109/DFT50435.2020.9250837

[69] Zhao K. et al., Algorithm-based fault tolerance for convolutional neural networks, arXiv preprint arXiv:2003.12203, 2020

[70] GOST R 8.673-2009, *State system for ensuring the uniformity of measurements. Intelligent sensors and intelligent measuring systems. Basic terms and definitions, Moscow, Standartinform, 2010*

[71] GOST R 8.734-2011, *State system for ensuring the uniformity of measurements. Intelligent sensors and intelligent measuring systems. Methods of metrological self-checking, Moscow, Standartinform, 2012*

[72] UK standard BSI (2005) Specification for Data Quality Metrics of Industrial Measurement and Control Systems, BS7986:2005, British Standards Institute, 389 Chiswick High Rd, London W4 4AL

[73] VDI/VDE Guideline 2650 (2006), *Requirements Regarding Self-monitoring and Diagnosis in Field Instrumentation. General Faults and Fault Conditions of Field Devices, Part 2*

[74] NAMUR, 2005), Self–monitoring and Diagnosis of Field Devices, Recommendation NE 107

[75] Ferreira R.S., Towards safety monitoring of ML-based perception tasks of autonomous systems, 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Coimbra, Portugal, 2020, pp. 135-138, doi: 10.1109/ISSREW51248.2020.00052

[76] Schlosser P., Ledermann C., Using Diverse Neural Networks for Safer Human Pose Estimation: Towards Making Neural Networks Know When They Don't Know, 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 10305-10312, doi: 10.1109/IROS45743.2020.9341634

[77] Biondi A., Nesti F., Cicero G., Casini D., Buttazzo G., Safe A, Secure and Predictable Software Architecture for Deep Learning in Safety-Critical Systems, in IEEE Embedded Systems Letters, vol. **12**, no. 3, pp. 78-82, Sept. 2020, doi: 10.1109/LES.2019.2953253

[78] Ibrahim Y., Wang H., Adam K., Analyzing the Reliability of Convolutional Neural Networks on GPUs: GoogLeNet as a Case Study, 2020 International Conference on Computing and Information Technology (ICCIT-1441), Tabuk, Saudi Arabia, 2020, pp. 1-6, doi: 10.1109/ICCIT-144147971.2020.9213804

[79] Salay R., Angus M., Czarnecki K., A Safety Analysis Method for Perceptual Components in Automated Driving, 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE), Berlin, Germany, 2019, pp. 24-34, doi: 10.1109/ISSRE.2019.00013

[80] Vasisht Duddu N., Rajesh Pillai, D. Vijay Rao, Valentina E. Balas, Fault Tolerance of Neural Networks in Adversarial Settings, arXiv:1910.13875

[81] Zahid U., Gambardella G., Fraser N.J., Blott M., Vissers K., FAT: Training Neural Networks for Reliable Inference Under Hardware Faults, 2020 IEEE International Test Conference (ITC), Washington, DC, USA, 2020, pp. 1-10, doi: 10.1109/ITC44778.2020.9325249

[82] Powers D. M. W., Atyabi A., The Problem of Cross-Validation: Averaging and Bias, Repetition and Significance, 2012 Spring Congress on Engineering and Technology, **2012**, pp. 1-5, doi: 10.1109/SCET.2012.6342143

[83] Hari S. et al., Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications, The International Conference for High Performance Computing, Networking, Storage and Analysis (SC), Nov 2017

[84] Wu M., Robustness Evaluation of Deep Neural Networks with Provable Guarantees, PhD thesis, Oxford, 2020

[85] Chen T.Y., Chung S.C., Yiu S.M., Metamorphic Testing - A New Approach for Generating Next Test Cases, HKUST-CS98-01, The Hong Kong University of Science and Technology, 1998

[86] Krizhevsky A., Sutskever I., Hinton G.E., ImageNet Classification with Deep Convolutional Neural Networks, In Adv. NIPS **2012**, pp.1097-1105, 2012

[87] Goodfellow I.J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A., Bengio Y., Generative Adversarial Nets, In Adv. NIPS **2014**, pp.2672-2680, 2014

[88] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras and Adrian Vladu, Towards Deep Learning Models Resistant to Adversarial Attacks, The Sixth International Conference on Learning Representations, (ICLR 2018). 2018

[89] Pei K., Cao Y., Yang J., Jana S., DeepXplore: Automated Whitebox Testing of Deep Learning Systems, The 26th Symposium on Operating Systems Principles (SOSP 2017), pp.1-18, 2017

[90] Xu W., David Evans and Yanjun Qi, Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks, Network and Distributed Systems Security Symposium (NDSS), 2018

[91] Mohammed S., Majdi, Sundaresh Ram, Jonathan T. Gill, Jeffery J. Rodriguez, Drive-Net: Convolutional Network for Driver Distraction Detection, 2018 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI), Las Vegas, NV, 2018, pp. 1-4

[92] *Data Safety Guidance* (Version 3.3) by the SCSC Data Safety Initiative Working Group

[93] *The SCSC Data Safety Initiative Working Group (DSIWG)* Ref: ISBN-13: 9798590573202

[94] National institute of Advanced Industrial Science and Technology *Machine Learning Quality Management Guideline* – 2nd English edition. https://www.digiarc.aist.go.jp/en/publication/aiqm/guideline-rev2.html

[95] ISO/IEC/TR 19791:2010, *Information technology — Security techniques — Security assessment of operational systems*

[96] Samek Wojciech, Binder Alexander, Montavon Gregoire, Bach Sebastian, Müller Klaus-Robert, Evaluating the visualization of what a Deep Neural Network has learned, arXiv.org, 2015

[97] Mohseni Sina, Pitale Mandar, Singh Vasu, Wang Zhangyang, 2020. Practical Solutions for Machine Learning Safety in Autonomous Vehicles. In SafeAI@AAAI

[98] Pimentel Marco A.F., Clifton David A., Clifton Lei, Tarassenko Lionel2014. , A Review of Novelty Detection. Signal Processing **99** (June): 215–49. https://doi.org/10.1016/J.SIGPRO.2013.12.026

[99] Goodfellow I.J., Shlens J., Szegedy C., 2015. Explaining and Harnessing Adversarial Examples. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings

[100] Xu W., David Evans, and Yanjun Qi. Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks. In Network and Distributed Systems Security Symposium (NDSS), 2018

[101] Ma S., Liu Y., Tao G., Lee W.-C., Zhang X., NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In Network and Distributed Systems Security Symposium (NDSS), 2019

[102] Ames A.D., Coogan S., Egerstedt M., Notomista G., Sreenath K., Tabuada P., 2019. Control Barrier Functions: Theory and Applications. 2019 18th European Control Conference, ECC 2019, June, 3420–31. https://doi.org/10.23919/ECC.2019.8796030

[103] Nowlan Steven, Hinton Geoffrey, Simplifying neural networks by soft weight-sharing. Neural Computation, **4**(4), 1992

[104] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, Salakhutdinov Ruslan, Dropout: A Simple Way to Prevent Neural Networks from Overfitting, Journal of Machine Learning Research **15**(56):1929–1958, 2014

[105] Madry A., Makelov A., Schmidt L., Tsipras D., Vladu A., Towards Deep Learning Models Resistant to Adversarial Attacks. In the Sixth International Conference on Learning Representations (ICLR 2018), 2018

[106] Eric Wong and J. Zico Kolter, Provable defenses against adversarial examples via the convex outer adversarial polytope. The 35th International Conference on Machine Learning (ICML 2018), PMLR vol. 80, pp. 5283–5292, 2018

[107] Liu X., Cheng M., Zhang H., Hsieh C., Towards robust neural networks via random self-ensemble. In Proceedings of the European Conference on Computer Vision (ECCV 2018), 2018

[108] Rebuffi, Sylvestre-Alvise, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A. Mann. Data Augmentation Can Improve Robustness. Advances in Neural Information Processing Systems **34** (2021)

[109] Papernot N., McDaniel P., Wu X., Jha S., Swami A., 2016. Distillation as a Defense to Adversarial Perturbations against Deep Neural Networks. In 2016 IEEE Symposium on Security and Privacy

[110] SHAFIQUE Muhammad, NASEER Mahum, THEOCHARIDES Theocharis, KYRKOU Christos, MUTLU Onur, OROSA Lois, CHOI Jungwook2020. , Robust Machine Learning Systems: Challenges, Current Trends, Perspectives, and the Road Ahead. IEEE Design and Test **37** (2): 30–57, https://doi.org/10.1109/MDAT.2020.2971217

[111] BAHDANAU D., CHO K., BENGIO Y.2014). , Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473

[112] XU K., BA J., KIROS R., CHO K., COURVILLE A., SALAKHUDINOV R. et al., 2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR

[113] PARMAR N., VASWANI A., USZKOREIT J., KAISER L., SHAZEER N., KU A. et al., 2018, July). Image transformer. In International Conference on Machine Learning (pp. 4055-4064). PMLR

[114] JETLEY S., LORD N.A., LEE N., TORR P.H.2018). , Learn to pay attention. arXiv preprint arXiv:1804.02391

[115] YU Y., CHOI J., KIM Y., YOO K., LEE S.H., KIM G., 2017). Supervising neural attention models for video captioning by human gaze data. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 490-498)

[116] RIBEIRO M.T., SINGH S., GUESTRIN C., 2016, August). Why should i trust you? Explaining the predictions of any classifier. In Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (pp. 1135-1144)

[117] LUNDBERG S. M., LEE S. I.2017). , A unified approach to interpreting model predictions. Advances in neural information processing systems, 30

[118] SIMONYAN K., VEDALDI A., ZISSERMAN A., 2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. In Workshop at International Conference on Learning Representations

[119] SMILKOV D., THORAT N., KIM B., VIÉGAS F., WATTENBERG M.2017). , Smoothgrad: removing noise by adding noise. arXiv preprint arXiv:1706.03825

[120] ZHOU B., KHOSLA A., LAPEDRIZA A., OLIVA A., TORRALBA A., 2016). Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2921-2929)

[121] SELVARAJU R.R., COGSWELL M., DAS A., VEDANTAM R., PARIKH D., BATRA D., 2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626)

[122] ZEILER M.D., FERGUS R., 2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham

[123] BOJARSKI M., CHOROMANSKA A., CHOROMANSKI K., FIRNER B., ACKEL L.J., MÜLLER U. et al., 2018, May). Visualbackprop: Efficient visualization of CNNs for autonomous driving. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 4701-4708). IEEE

[124] SHAFIQUE Muhammad, NASEER Mahum, THEOCHARIDES Theocharis, KYRKOU Christos, MUTLU Onur, OROSA Lois, CHOI Jungwook2020. , Robust Machine Learning Systems: Challenges, Current Trends, Perspectives, and the Road Ahead. IEEE Design and Test **37** (2): 30–57, https://doi.org/10.1109/MDAT.2020.2971217

[125] ISO/IEC 5259 (all parts), *Artificial intelligence — Data quality for analytics and machine learning (ML)*[3)]

[126] DECKERT G., NASA Hazard Analysis Process. Johnson Space Center, National Aeronautics and Space Administration, 2010. https://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20100040678.pdf

3) Under preparation.

[127] Chen X., Liu C., Li B., Lu K., Song D.2017. , Targeted backdoor attacks on deep learning systems using data poisoning. arXiv preprint arXiv:1712.05526

[128] Gu T., Dolan-Gavitt B., Garg S.2017. , Badnets: Identifying vulnerabilities in the machine learning model supply chain. arXiv preprint arXiv:1708.06733

[129] Tramèr F., Zhang F., Juels A., Reiter M.K., Ristenpart T., 2016. Stealing machine learning models via prediction APIs. In 25th {USENIX} Security Symposium ({USENIX} Security 16) (pp. 601-618)

[130] Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D., Goodfellow I. et al.2013. , Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199

[131] Geirhos R., Jacobsen J.H., Michaelis C., Zemel R., Brendel W., Bethge M., Wichmann F.A.2020. , Shortcut learning in deep neural networks. Nature Machine Intelligence, **2**(11), pp.665-673

[132] Alfarra M., Pérez J.C., Thabet A., Bibi A., Torr P.H., Ghanem B.2021. , Combating Adversaries with Anti-Adversaries. arXiv preprint arXiv:2103.14347

[133] Tian Y., Suya F., Xu F., Evans D.2021. , Stealthy Backdoors as Compression Artifacts. arXiv preprint arXiv:2104.15129

[134] Rakin A.S., He Z., Fan D., 2019. Bit-flip attack: Crushing neural network with progressive bit search. In Proceedings of the IEEE/CVF International Conference on Computer Vision (pp. 1211-1220)

[135] Fischer M., Balunovic M., Drachsler-Cohen D., Gehr T., Zhang C., Vechev M., 2019, May. Dl2: Training and querying neural networks with logic. In International Conference on Machine Learning (pp. 1931-1941). PMLR

[136] Ashmore R., Calinescu R., Paterson C.2021. , Assuring the machine learning lifecycle: Desiderata, methods, and challenges. ACM Computing Surveys *(CSUR)*, **54**(5), pp.1-39

[137] Banks A., Ashmore R., 2019. Requirements assurance in machine learning. In Workshop on Artificial Intelligence Safety, 2019

[138] Chris Hobbs, Embedded Software Development for Safety-Critical Systems, 2nd ed

[139] Kläs M., Adler R., Sorokos, I., Joeckel, L., Reich, J. Handling Uncertainties of Data-Driven Models in Compliance with Safety Constraints for Autonomous Behaviour, preprint on researchgate.net

[140] NIST 800-35, *Guide to Information Technology Security Services*

[141] Data Safety Guide, Data Systems Safety Club, (v3.3, 2021)

[142] Urban C., Miné A., A Review of Formal Methods applied to Machine Learning, arXiv: 2104.02466

[143] IEC/TS 62998-1:2019, *Safety of machinery - Safety-related sensors used for the protection of persons*

[144] IEC 61496 (all parts), *Safety of machinery - Electro-sensitive protective equipment*

[145] ISO 3691-4:2023, *Industrial trucks — Safety requirements and verification — Part 4: Driverless industrial trucks and their systems*

[146] ANSI/RIA R15 08-1-2020, *Industrial Mobile Robots - Safety Requirements - Part 1: Requirements For The Industrial Mobile Robot*

[147] ISO 31000, *Risk management — Guidelines*

[148] ISO/IEC 25010, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Product quality model*

[149] ISO/IEC 27001:2022, *Information security, cybersecurity and privacy protection — Information security management systems — Requirements*

[150] ISO/IEC 18045:2022, *Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Methodology for IT security evaluation*

[151] ISA/IEC 62443 (all parts), *Security for Industrial Automation and Control Systems*

[152] ISO/IEC/TR 29119-11:2020, *Software and systems engineering — Software testing — Part 11: Guidelines on the testing of AI-based systems*

[153] ISO/IEC/TR 24029-1, *Artificial Intelligence (AI) — Assessment of the robustness of neural networks — Part 1: Overview*

[154] Boyd S., Vandenberghe L. 2006). *Convex Optimization*

[155] Taymanov R., Sapozhnikova K., Intelligent Measuring Instruments. Maximum Reliability of Measuring Information, Minimum Metrological Maintenance 2003 in Proceedings of the XVII IMEKO World Congress, Dubrovnik, Croatia, 22–27 June, 2003, 1094–1097 Available: https://www.imeko.org/index.php/proceedings?start=0 Access: 2022.09.09

[156] Ghahramani Z., Probabilistic machine learning and artificial intelligence. Nature **521**:452–459 (2016)

[157] Gal Y., Uncertainty in deep learning. University of Cambridge 1, 3 (2016)

[158] Transparency and explainability (Principle 1.3). OECD. Available at https://oecd.ai/en/dashboards/ai-principles/P7

[159] Recommendation on the Ethics of Artificial Intelligence. UNESCO. Available at https://www.unesco.org/en/artificial-intelligence/recommendation-ethics

Bureau of Indian Standards

BIS is a statutory institution established under the *Bureau of Indian Standards Act*, 2016 to promote harmonious development of the activities of standardization, marking and quality certification of goods and attending to connected matters in the country.

Copyright

Review of Indian Standards

Amendments are issued to standards as the need arises on the basis of comments. Standards are also reviewed periodically; a standard along with amendments is reaffirmed when such review indicates that no changes are needed; if the review indicates that changes are needed, it is taken up for revision. Users of Indian Standards should ascertain that they are in possession of the latest amendments or edition by referring to the website-www.bis.gov.in or www.standardsbis.in.

This Indian Standard has been developed from Doc No.: LITD 30 (24862).

Amendments Issued Since Publication

Amend No.	Date of Issue	Text Affected

BUREAU OF INDIAN STANDARDS

Headquarters:

Manak Bhavan, 9 Bahadur Shah Zafar Marg, New Delhi 110002
Telephones: 2323 0131, 2323 3375, 2323 9402 *Website*: www.bis.gov.in

Regional Offices: *Telephones*

Central : 601/A, Konnectus Tower -1, 6th Floor,
DMRC Building, Bhavbhuti Marg, New
Delhi 110002
{ 2323 7617

Eastern : 8th Floor, Plot No 7/7 & 7/8, CP Block, Sector V,
Salt Lake, Kolkata, West Bengal 700091
{ 2367 0012
2320 9474

Northern : Plot No. 4-A, Sector 27-B, Madhya Marg,
Chandigarh 160019
{ 265 9930

Southern : C.I.T. Campus, IV Cross Road, Taramani, Chennai 600113
{ 2254 1442
2254 1216

Western : Manakalya, 4th Floor, NTH Complex (W Sector), F-10, MIDC, Andheri
(East), Mumbai 400093
{ 283 25838

Branches : AHMEDABAD, BENGALURU, BHOPAL, BHUBANESHWAR, CHANDIGARH, CHENNAI, COIMBATORE, DEHRADUN, DELHI, FARIDABAD, GHAZIABAD, GUWAHATI, HARYNA, HUBLI, HYDERABAD, JAIPUR, JAMMU & KASHMIR, JAMSHEDPUR, KOCHI, KOLKATA, LUCKNOW, MADURAI, MUMBAI, NAGPUR, NOIDA, PARWANOO, PATNA, PUNE, RAIPUR, RAJKOT, SURAT, VIJAYAWADA.

Published by BIS, New Delhi