



Structured glossary – Embedded AI

Version 1.1, 22-07-2026

Authors

Eliane Fourgeau	Quantyss
Pierre Gaillard	CEA-List
Bertrand Braunschweig	European Trustworthy AI Association
Frédéric Cetlin	Leminnov
Florent Zara	Eclipse Foundation

In collaboration with:

- **CEA List** - list.cea.fr
- **Leminnov** - www.leminnov.com
- **DeepGreen** - www.deepgreen.ai
- **European Trustworthy AI Association** - www.trustworthy-ai-association.eu
- **Eclipse Foundation** - www.eclipse.org
- **Liane** - liane.fr

Version 1.1, 22-07-2026.

This glossary serves to document and clarify the terminology associated with embedded artificial intelligence. Its aim is to provide a reference framework designed to facilitate the understanding, sharing and adoption of concepts, without claiming to be exhaustive..

The authors accept no responsibility for the content of this document, which does not reflect the official, strategic or normative positions of the companies, institutions or organisations that contributed to its development. It is neither a standardisation document nor a formal standard or recommendation, but rather a tool to support reading and the structuring of knowledge in a rapidly evolving field.

This glossary has been compiled based on a comparative analysis of various sources, including in particular the Confiance.ai taxonomy, encyclopedic resources (such as French and English Wikipedia), European regulatory texts relating to AI, as well as technical, educational and scientific publications.

To quote this document: “Structured Glossary of Embedded AI by Embedded France with the Eclipse Foundation | Licence CC BY 4.0“

Table of content

1. General introduction	4
2. Data framework, acquisition and processing	9
2.1. Data framework	10
2.2. Data acquisition	12
2.3. Data Processing	14
3. Artificial intelligence in embedded systems	17
3.1. Major Families of AI	18
3.2. Models for embedded AI	21
3.3. Optimisation techniques for embedded AI	23
4. System architecture for embedded AI	27
5. Software and tools for embedded AI	30
5.1. Languages	30
5.2. Model management	31
5.3. Platforms and development kits	32
5.4. Hardware libraries	34
5.5. Compilers	34
5.6. Runtime	35
5.7. Operating systems	36
6. Hardware for Embedded AI	39
6.1. Hardware architectures and computing paradigms	39
6.2. Specialised processors and accelerators	41
6.3. Specialised circuits and embedded architectures	42
6.4. Memory and interconnect	43
7. Communications & connectivity in embedded systems	45
7.1. Physical networks	45
7.2. Application protocols	46
8. Advanced and emerging paradigms in embedded AI	48
8.1. Embedded generative AI	48
8.2. Agent-based AI and autonomous systems	49
8.3. Physical AI and interaction with the real world	50
8.4. Eco-design and resource efficiency in embedded AI	51
9. Testing, validation & certification of embedded AI	53
9.1. Metrics	53
9.2. Benchmarks	55
10. Trustworthy Embedded AI	56
10.1. General considerations on ethics	56
10.2. Reliability	56
10.3. Bias and fairness	57

10.4. Explainability and interpretability	57
10.5. Transparency and traceability	58
10.6. Cybersecurity	59
10.7. Qualification, certification, evidence.....	60
11. Lifecycle and Governance of Embedded AI.....	62
11.1. Lifecycle of Embedded AI.....	62
11.2. Validation and lifecycle management.....	63
11.3. Governance of Embedded AI.....	64
11.4. Controlled operation and monitoring.....	65
11.5. Updates and Operational Continuity.....	66
12. Regulations and standards for embedded AI	67
12.1. European regulations	67
12.2. Cross-cutting directives with significant impact.....	68
12.3. European standards supporting AI regulation	68
12.4. International standards for governance and risk management	69
12.5. Sector-specific standards	70
13. Applications and Use Cases	75
13.1. General concepts	75
13.2. Major families of sector-specific applications.....	76
13.3. Typical application functions	77
13.4. Structural cross-cutting constraints.....	78
13.5. Representative use cases	79
13.6. Emerging use cases	80

1. General introduction

A key initiative for the field

This glossary is an initiative of [Embedded France](#)'s "Embedded AI" working group, led by experts from industry, academia and public institutions at the forefront of the discipline. It reflects a collective desire to establish a common language around embedded artificial intelligence technologies and is part of a commitment to transparency and open dissemination, with the aim of fostering collective adoption of these concepts.

A clear ambition, a humble approach

This document is first and foremost a taxonomy-oriented reference guide, intended to define the essential terms that underpin the practices, architectures and applications of embedded AI. It does not claim to be exhaustive or prescriptive, but offers a rigorous, contextualised and operational foundation, open to collective contributions.

We want it to remain a living, evolving document, accessible to all stakeholders in the sector. That is why we are making it freely available, in the spirit of transparency, sharing and open structuring.

An exploration through time: past, present and future

This glossary explores embedded artificial intelligence in terms of its historical depth, current uses and future trajectories - from expert systems and real-time architectures to frugal, distributed and generative models, which support autonomy and technological sovereignty.

Aware of the **rapid pace and diversity of developments** in this field, the glossary will be reviewed and updated at regular intervals to keep pace with technological advances and the sector's needs.

Intrinsic value based on usage

This document is not merely a compilation of definitions:

- It links technical concepts and architectures with their practical implications in embedded, distributed and real-time systems,
- It describes hardware components (processors, sensors, interconnections) and software environments (optimisation, execution),
- It defines learning paradigms (supervised, federated, transfer learning, etc.) and forms of embedded inference,
- It structures use cases in vision, audio, natural language, time series and intelligent interfaces, and incorporates advanced concepts (generative AI, agent-based AI, etc.) as well as emergent behaviours (physical AI, digital twins, etc.).

In dialogue with existing initiatives

This glossary complements other works (non-exhaustive list):

- The [Confiance.ai programme glossary](#) (IRT SystemX), focused on trustworthy AI in critical systems,

- The [AI Act terminology framework](#) (European Commission), which sets out regulatory obligations,
- The [ISO/IEC JTC 1/SC 42 technical reports](#), which define standard AI concepts, including embedded AI.

It goes further by:

- Targeting **embedded AI in its hardware, software, algorithmic and application dimensions**, with particular attention paid to frugality, explainability and distributed autonomy.
- Exploring embedded artificial intelligence **in terms of its historical depth, current uses and future trajectories**.

In summary, this glossary aims to provide a **clear, operational and contextualised mapping** of the fundamental concepts of embedded AI, as they are applied in industrial, academic and institutional projects; it pursues several **complementary objectives**:

- **Establishing a common language**: to set out clear, shared definitions of the concepts, architectures, models and tools specific to embedded AI.
- **Facilitating understanding and adoption**: provide stakeholders in the sector with an accessible reference framework that links technical concepts to their practical applications in embedded systems.
- **Strengthening credibility and disciplinary coherence**: to propose a rigorous taxonomy that consolidates the place of embedded AI within the technological and academic landscape.
- **Supporting innovation and technological sovereignty**: providing a useful framework for industrial projects, research work and institutional initiatives.
- **Fostering collective development**: enabling members of the community to enrich and update the glossary, so that it remains a relevant and sustainable tool.

Methodological note on the scope of terminology

It should be noted that the boundaries between artificial intelligence (AI), embedded AI and embedded systems remain somewhat blurred; the fundamental terminology of these fields overlaps due to their shared history, technical interdependencies and the rapid evolution of their applications. In this glossary, some of the terms included are not strictly reserved for embedded AI. They fall more broadly within the scope of AI or embedded systems in the general sense. Their inclusion serves a dual purpose:

- To ensure consistency and continuity of vocabulary: many generic concepts are essential building blocks for understanding and deploying AI in embedded environments. * To facilitate adoption by the community: by incorporating shared definitions, the glossary provides industry stakeholders with a common reference framework, avoiding ambiguities and enhancing the clarity of communication.

This methodological choice reflects the aim of providing an operational and inclusive document that links the specific features of embedded AI with the broader foundations of AI and embedded systems.

READING GUIDE

The **reading guide** below provides an overview of the glossary's structure by presenting, for each chapter, the topics covered, their role within the document and their links to other chapters. It aims to guide the reader towards a gradual understanding of the key concepts of embedded artificial intelligence and to facilitate navigation across the different sections of the glossary.

Chapter	Role in the glossary	Main content covered	Links to other chapters
1. General introduction	Define the framework, scope and objectives of the glossary	Scope of embedded AI, structure of the document, working method	Provides the common framework for reading all chapters (2 to 13)
2. Data framework and concepts for embedded AI	Establish the foundations for data in embedded AI	Data, datasets, quality, flows, traceability	Provides the conceptual basis for the chapters on models, evaluation and the lifecycle (4, 5, 9, 11)
3. Introduction to artificial intelligence for embedded systems	Introduce the fundamentals of AI tailored to embedded systems	Principles of AI, types of learning, inference under constraints	Serves as the conceptual foundation for the chapters on advanced methods, architectures and paradigms (4, 6, 8)
4. AI methods and models for embedded systems	Describe algorithmic approaches suitable for embedded systems	Models, learning methods, inference and adaptation	Builds on the foundations of AI (3) and informs the chapters on execution and evaluation (6, 9, 11)
5. Data, acquisition and processing for embedded AI	Describe the data chain from the sensor to the model	Sensors, acquisition, preprocessing, structuring of data streams	Complements the data framework (2) and interfaces with methods and architectures (4, 6)
6. Hardware and software architectures	Present the execution platforms for embedded AI	Processors, accelerators, embedded systems, OS	Supports the implementation of models and tools (4, 7), and determines operation (11)
7. Tools, frameworks and environments	Describe development and execution tools	Frameworks, libraries, embedded toolchains	Provides practical implementation of the methods and architectures described previously (4, 6)
8. Advanced and emerging paradigms in embedded AI	Explore developments and non-traditional approaches	Embedded generative AI, agent-based AI, physical AI, frugality	Complements the foundations of AI (3) without redefining concepts, and sheds light on future applications (13)

Chapter	Role in the glossary	Main content covered	Links to other chapters
9. Performance, metrics and evaluation	Qualify and measure the behaviours of embedded AI	Latency, accuracy, power consumption, robustness	Provides cross-cutting criteria applicable to models and architectures (4, 6), and informs governance (11)
10. Safety, robustness and trust	Define the reliability and control of systems	Robustness, operational safety, trust	Extends the technical assessment (9) with concepts of operational control and reliability (11)
11. Lifecycle and governance of embedded AI	Structure design, deployment and operation	Lifecycle, validation, monitoring, updates	Linking technical design (4, 6), operation (9, 10) and compliance (12)
12. Regulations, standards and reference frameworks	Present the applicable regulatory frameworks	European regulations, standards and reference frameworks	Complements operational governance (11) by providing the regulatory framework
13. Applications and use cases for embedded AI	Illustrate practical and sector-specific uses	Application functions, sectors, benefits and constraints	Places all the technical chapters into context through practical applications (2 to 12)

The **rules for formulating** definitions aim to ensure consistency, readability and operational use of the glossary. Each entry presents a term in French, accompanied by its English equivalent where relevant. Where the term is commonly referred to by an acronym, this is specified and included in the terminology to facilitate the reading and use of the glossary. In some cases, the English term is preferred where it is the dominant reference in scientific, technical or industrial literature.

Definitions are formulated concisely and in context, with an explicit link to embedded AI where the concept applies. Where necessary, so as to help understanding, clarifications or examples of usage have been added. Definitions drawn from recognised standards (such as Confiance.ai, ISO/IEC or equivalent frameworks) are adopted and adapted to ensure terminological continuity and consistency with existing practices.

Concepts are defined in a non-redundant manner within the glossary: a term is formalised in a reference chapter and may be used in other chapters without being redefined. All definitions adopt a descriptive and operational approach, without being prescriptive or commercial in nature.

Some **translations** were produced with the support of the DeepL tool. Furthermore, generative AI tools (such as Perplexity, ChatGPT, Copilot and Mistral) were used to assist with research, structuring and reformulating content, under the supervision and validation of the editors.

This **editorial approach** allows the glossary to be approached as a structured and coherent whole, facilitating the understanding of concepts and their interrelationships, from the technical

foundations to the application-level and governance-related challenges of embedded AI.

2. Data framework, acquisition and processing

In an embedded AI system, data, its acquisition and its processing cannot be considered as independent elements. They form a coherent whole that determines the system's ability to perceive its environment, make reliable decisions and operate autonomously under operational constraints.

Data forms the raw material of embedded AI. Its nature, quality and representation determine what the system can perceive and interpret. Acquisition links this data to the real world, ensuring a continuous or event-driven production of information tailored to usage contexts. Finally, data processing plays a role in adaptation and stabilisation, transforming this information into inputs compatible with the models deployed on resource-constrained embedded platforms.

This structure is specific to embedded AI, as it operates within an environment where resources are limited, operating conditions are variable, and real-time and safety requirements are high. The choices made at each level directly affect the others, and the balance between these three dimensions determines the overall system performance.

2.1. Data framework

This chapter brings together the terminology used to define what constitutes usable data in an embedded AI system. It specifies the concepts needed to characterize the nature of the data, its structure, representation and use within the context of models deployed on constrained platforms, i.e. embedded devices with limited computational, memory, storage and energy resources, and subject to real-time and robustness requirements.

Understanding these terms is essential for assessing the relevance and reliability of the data used by embedded AI. A clear definition of data-related concepts helps to avoid ambiguous interpretations and ensures that models are based on information suited to local execution constraints.

Data Engineering

A discipline aimed at organizing, structuring, selecting, transforming and tracing the data used by embedded AI systems, to guarantee its quality, availability, operational relevance and traceability throughout the system's lifecycle, subject to real-time, hardware resource and safety constraints.

Data

Raw information derived from observations of the physical or logical world, primarily produced by embedded sensors, and used as input for embedded AI processing and models for local inference and decision-making.

Digital Data

A coded representation, usually binary, of physical or logical information, intended to be processed by an embedded system running AI algorithms, under constrained computing, memory and latency conditions.

Dataset

A structured collection of data used for training, validating or testing AI models, generally compiled prior to deployment and serving as a reference for models intended to be run on embedded systems.

Sample

An individual instance of a dataset representing a single observation (sensor measurement, image, signal, sequence), used as input to an embedded AI model during training or inference.

Feature

A measurable variable or attribute derived from a sample, describing a relevant aspect of the data and serving as an input representation for an embedded AI model, often extracted or calculated locally under resource constraints.

Feature Space

The set of features used to represent data in an on-device AI model, defining the dimension and structure of the model's input, with a direct impact on the complexity, performance and cost of inference.

Test Data

A separate dataset used to objectively measure the final performance of an AI model prior to its integration and deployment in an embedded system.

Label

A value, class or category associated with a sample in the context of supervised learning, serving as a reference for the training and evaluation of embedded AI models.

Data Sampling

The process of selecting a representative subset of data from a larger dataset, used to limit volumes, balance classes or adapt the data to the constraints of training and embedded deployment.

Training Data

A subset of a dataset used to adjust the parameters of an AI model, typically processed offline prior to the model's deployment on an embedded platform.

Validation Data

A subset of data used during the training phase to evaluate a model's interim performance, adjust hyperparameters, and prevent overfitting prior to on-device deployment.

Data Card or Datasheet for Datasets

A document providing a concise and standardized description of a dataset's characteristics (origin, composition, collection methods, intended uses, limitations, identified biases), to ensure the transparency, traceability and compliance of the embedded AI models derived from it.

Data Stream

A continuous or event-driven stream of data produced by on-board sensors and processed locally by AI algorithms, often subject to constraints regarding latency, bandwidth and energy consumption.

Real-time data

Data collected and processed within strictly defined timeframes to enable an immediate or near-immediate response from an embedded AI system, particularly in critical or autonomous applications.

Data Quality

The set of properties (accuracy, consistency, completeness, freshness, noise) that determine the reliability of the inferences produced by an embedded AI system and its ability to operate safely and robustly in real-world conditions.

Missing or Incomplete Data

A situation in which certain expected information is unavailable or incomplete, requiring appropriate management mechanisms to preserve the stability and safety of the embedded AI.

Embedded Data Traceability

The ability to trace the origin, transformations, versions and uses of data within an embedded AI system, in support of safety, certification, auditability and regulatory compliance requirements.

Embedded Data Security

The set of technical mechanisms designed to protect data used by embedded AI against unauthorized access, tampering and loss, whilst respecting the system's computational and energy constraints.

Data Annotation

The process of associating descriptive information (labels, categories, metadata) with data, generally carried out prior to embedded deployment, to enable the training and evaluation of supervised AI models intended for local inference.

Data Augmentation

A technique involving the artificial generation of new samples from existing data (geometric transformations, noise, signal variations), to improve the robustness and generalization of AI models intended for on-device execution.

2.2. Data acquisition

This chapter brings together terminology relating to the devices and mechanisms used to generate the data utilised by embedded AI, describing the sensors and concepts associated with their operation in embedded systems exposed to variable environments.

The concepts presented help to understand how data is adapted to the requirements of embedded AI execution, in terms of format, representation and robustness against disturbances or signal variations. Data processing thus plays an essential role in the stability and predictability of the system's behaviour under operational conditions.

Sensor

An embedded hardware device that measures a physical quantity (temperature, pressure, light, sound, movement, etc.) and converts it into an electrical or digital signal that can be processed by embedded AI models, generally in real time and under resource constraints.

Imager

An optical sensor that converts light into digital signals to produce images that can be processed by on-board computer vision algorithms for perception, detection or classification.

Event-based camera

An asynchronous optical sensor that does not capture images at a fixed frequency but generates events only when there are local variations in brightness across the pixels, enabling on-board perception with very low latency and low energy consumption.

LiDAR sensor – Light Detection and Ranging

An active sensor that emits laser pulses to measure the distance to surrounding objects and produce a three-dimensional representation of the environment, usually in the form of a point cloud, used for perception and localisation in on-board AI.

RADAR sensor – Radio Detection and Ranging

A sensor that uses radio waves to detect the presence, distance, speed and sometimes the direction of objects; it is particularly robust in adverse weather conditions (rain, fog) and is

widely used for on-board perception.

Inertial sensors – Inertial Measurement Unit (IMU)

A set of sensors generally comprising an accelerometer, a gyroscope and sometimes a magnetometer, used to measure linear and rotational movements, which are essential for navigation, stabilisation and multi-sensor fusion in on-board AI.

Geolocation sensors

Sensors or systems used to estimate the position or trajectory of an on-board system (GNSS, UWB, local beacons), often combined with inertial sensors to improve the robustness of on-board perception and navigation.

Audio sensors

Sensors that capture sound or vibration signals (microphones, ultrasonic sensors), used in embedded AI for speech recognition, acoustic event detection, diagnostic or monitoring.

Proximity and short-range sensors

Sensors measuring short-range distances (ultrasound, infrared, short-range ToF), used in robotics, connected devices and embedded systems for obstacle detection and interaction with the immediate environment.

Environmental sensors

Sensors measuring physical or chemical parameters of the environment (temperature, humidity, pressure, gases, particles, vibrations), used in embedded AI for monitoring, contextual analysis and anomaly detection.

Biometric and physiological sensors

Sensors measuring biological or physiological signals (heart rate, muscle or brain activity, oxygen saturation), used in embedded AI systems for healthcare, wearables and human monitoring.

Force, pressure and torque sensors

Sensors measuring mechanical forces or pressures, used in embedded AI for robotics, human-machine interaction and the control of physical systems.

Multispectral and hyperspectral sensors

Optical sensors capturing information across multiple spectral bands, enabling detailed analysis of materials or the environment, used in embedded AI in specialised fields (agriculture, environment, defence).

Smart sensor

A sensor incorporating local processing, filtering or feature extraction capabilities, enabling the reduction of transmitted data volumes and the optimisation of the performance and energy consumption of embedded AI.

Multi-sensor synchronisation

A mechanism ensuring the temporal alignment of data from heterogeneous sensors, essential for sensor fusion and the consistency of inferences in embedded AI.

Time-stamping

The association of a time reference with sensor data to ensure its consistency, traceability and correct use in real-time embedded systems.

Sensor drift

A gradual change in a sensor's characteristics due to the environment, wear and tear, or operating conditions, which may degrade data quality and the performance of embedded AI systems.

2.3. Data Processing

This chapter covers the terminology relating to operations applied to data to make it usable by embedded AI models. It describes the transformations required to adapt the data to the computational, memory and real-time constraints of embedded systems.

The concepts covered highlight the importance of data processing for the performance and overall stability of embedded AI. A shared understanding of these terms enables the design of processing chains tailored to operational requirements.

Sensor Data Conditioning Pipeline

A set of operations applied to data from embedded sensors, after acquisition and before inference, aimed at transforming raw data into inputs that can be used by embedded AI models, whilst respecting the system's constraints regarding latency, resources, determinism and safety.

On-device preprocessing

Preprocessing carried out locally on the embedded device (microcontroller, SoC, DSP, NPU), forming an integral part of the inference chain, and aimed at improving the quality of model inputs whilst controlling computational, memory and energy costs.

Data cleaning

Operations designed to correct, filter or ignore inconsistent, erroneous or unusable data from sensors, in order to limit inference errors and undesirable behaviour in embedded AI.

Missing data handling

Mechanisms for handling the occasional absence of measurements (controlled ignorance, default values, simple estimation), in order to maintain stable, safe behaviour of the embedded system.

Denoising and filtering

Digital processing aimed at reducing noise and disturbances in sensor signals (frequency filtering, smoothing), in order to improve the robustness of the model inputs without introducing excessive latency.

Correction and compensation

A set of software mechanisms designed to correct residual biases or drifts in digital data (offset, slow drift), in order to maintain the consistency of the model inputs over time.

Data scaling

Transformation of numerical values to adapt them to the range expected by embedded models and accelerators, directly linked to quantification and digital representation constraints.

Input quantisation

Conversion of input data to reduced-precision representations (e.g. integers), in order to optimise execution on embedded targets whilst controlling the error introduced during inference.

Data formatting

Structural transformation of data (dimensions, channel organisation, numerical formats) to correspond exactly to the inputs expected by the embedded AI model.

Data encoding

Representation of discrete or categorical information in a form that can be utilised by the model (binary encoding, compact embeddings), with a focus on efficiency and embedded compatibility.

Feature extraction

The process of transforming raw data into relevant variables that reduce the complexity of the inputs, in order to improve the robustness, interpretability and efficiency of embedded inference.

Dimensionality reduction

Techniques aimed at limiting the number of input variables whilst retaining useful information, in order to reduce computational load, memory usage and latency on the embedded target.

Temporal windowing

The division of a data stream into sliding or fixed time segments, enabling local analysis adapted to the real-time constraints of embedded systems.

Partitioning

Logical division of data (signal, image, sequence) to restrict the area or period of interest analysed by the model, thereby reducing the cost of embedded processing.

Sampling rate adaptation

Software adjustment of the sampling frequency of digital data to adapt the volume of information to the computational capabilities and requirements of the embedded application.

Synchronisation and Temporal Alignment

Alignment of data streams from multiple sensors to ensure the temporal consistency of inputs to a multimodal embedded AI model.

Outlier filtering

Detection and processing of abnormal or inconsistent digital measurements to prevent erroneous or unstable inferences in disrupted embedded environments.

Deterministic preprocessing

A preprocessing pipeline designed to produce reproducible results within bounded execution

times, essential for integrating AI into real-time and mission-critical embedded systems.

Frugal preprocessing

Preprocessing optimised to minimise the use of hardware resources (computing power, memory, energy), in order to preserve the battery life and overall performance of the embedded system.

Preprocessing pipeline

A structured and validated sequence of data processing steps between acquisition and inference, considered an integral component of the embedded AI system.

Preprocessing validation

A process designed to verify that the preprocessing chain meets functional, timing and safety requirements, and that it does not degrade the overall system performance of the embedded AI under operational conditions.

3. Artificial intelligence in embedded systems

Embedded AI differs from ‘general’ AI due to a set of constraints specific to physical systems and operational environments. It must operate locally, often in real time, on limited resources (computing power, memory, energy), whilst ensuring robustness, security, reliability and regulatory compliance. These constraints profoundly influence the choice of models, hardware architectures, optimisation techniques and tools. The following chapters will highlight these specific features, whilst also reviewing the fundamentals of ‘general’ AI, which are essential for the educational completeness of this glossary.

The figure below illustrates the main paradigms of artificial intelligence, classifying them into two categories: data-driven (machine learning) and knowledge-driven (symbolic AI). AI referred to as ‘hybrid’ or ‘neurosymbolic’ is composed of these two paradigms. Conventional statistical methods, which are conceptually close to neural networks, are not usually considered part of AI: some considerations on this subject are set out in the “[Commission Guidelines on the definition of an artificial intelligence system established by Regulation \(EU\) 2024/1689 \(AI Act\)](#)”.

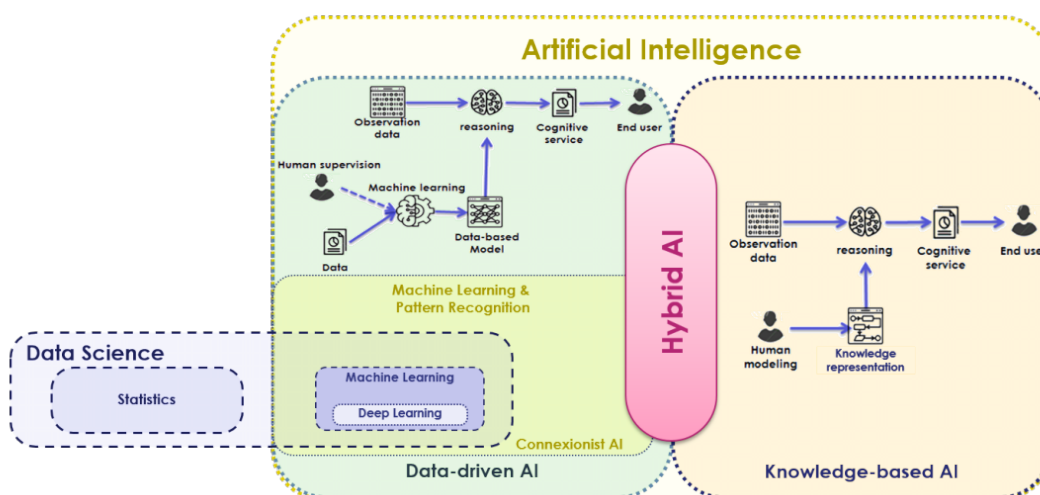


Figure 1. Figure reproduced courtesy of [Confiance.ai](#)

Artificial Intelligence (AI)

A set of mathematical and computational techniques enabling a machine (a computer) to simulate human cognitive functions (reasoning, learning, perception).

AI System

An automated system designed to operate at different levels of autonomy and capable of adapting after deployment, which, for explicit or implicit objectives, deduces from the inputs it receives how to generate outputs such as predictions, content, recommendations or decisions that may influence physical or virtual environments. (official European definition).

Multimodal AI

AI capable of simultaneously processing multiple types of data (text, images, audio, sensors, etc.). Important for smart embedded interfaces (e.g. voice assistants, smart glasses, etc.).

Inference

The phase in which a trained model is executed to produce a prediction or a decision. Inference follows training and precedes evaluation. It is the process by which an artificial intelligence model applied to new data produces a prediction, a decision or a classification, based on what it has learnt during its training phase. It is the practical implementation of the model in a real or embedded environment, often in real time, without retraining the model.

3.1. Major Families of AI

The fundamental approaches to artificial intelligence describe the major conceptual frameworks used to design intelligent systems. In the context of embedded AI, these approaches determine how systems perceive, reason and make decisions under constraints of resources, real-time processing and safety.

Connectionist AI

Based on statistical learning and neural networks; this is an approach to AI based on models inspired by the functioning of the human brain, in particular artificial neural networks. It excels at learning from big data and processing complex signals (image, sound, text).

Symbolic AI

An approach based on explicit representations of knowledge (rules, logic, ontologies) and inference mechanisms. It prioritises structure, explainability and the manipulation of abstract concepts.

Hybrid AI

AI that combines the strengths of the two previous approaches to meet requirements for robustness, explainability and adaptability: learning capability on the one hand, and explainability and control on the other. It is particularly relevant for embedded systems requiring robustness, safety and adaptability.

3.1.1. Machine learning paradigms

Machine learning paradigms define the different ways in which an AI system can acquire, adapt and utilise knowledge from data. In the context of embedded AI, the choice of learning paradigm strongly influences the system's architecture, its deployment method and its ability to operate under resource, real-time and environmental constraints.

Some paradigms rely on offline training phases followed by local inference, whilst others allow for partial or continuous adaptation as close as possible to the field, directly on embedded devices or in a distributed manner. These approaches address a variety of challenges, such as the limited availability of annotated data, the preservation of confidentiality, adaptation to evolving environments, and coordination between multiple embedded systems.

Machine Learning (ML)

A sub-discipline of artificial intelligence that involves designing algorithms capable of learning from data without being explicitly programmed for each task. The algorithms identify patterns in the data to build models capable of making predictions or decisions on new data.

There are different types of learning:

Supervised learning

A learning strategy where the model learns from labelled data (input-output pairs), adjusting its parameters to minimise the error between its predictions and the provided labels. Supervised learning is commonly used for classification and regression tasks.

Unsupervised learning

A learning strategy in which the model discovers the structures and patterns inherent in the data without reference labels. Unsupervised learning is used for tasks such as clustering, dimensionality reduction and latent feature extraction.

Semi-supervised learning

A learning paradigm that combines labelled and unlabelled data, exploiting the structure of the unlabelled data to improve model performance. Semi-supervised learning is particularly useful when data labelling is costly or limited.

Reinforcement learning

A learning strategy involving making optimal decisions by interacting with an environment, involving rewards or penalties based on one's actions. Reinforcement learning is applied to control, robotics and gaming tasks, where the model must learn a strategy without explicit labelled data. Trial-and-error learning with rewards based on actions.

Federated learning

A decentralised learning strategy enabling multiple devices or servers to collaborate to train a shared model without sharing their raw data. Each device (smartphone, sensor, robot, etc.) trains a local model on its own data, then sends only the model parameters (weights, gradients) to a central server. The server aggregates these updates to create a global model, which is then redistributed to all devices. As the data remains localised on each device, federated learning ensures greater privacy and enhanced regulatory compliance (e.g. GDPR). Also known as decentralised collaborative learning.

Transfer Learning

A learning strategy that involves reusing the knowledge acquired by a model on a source task to improve its performance on a target task, which is generally different but related. Rather than training a model from scratch, one starts with a pre-trained model (often on a large dataset) and adapts it to a new task, often with little data available. Fine-tuning, which involves retraining (partially or fully) a pre-trained model to adapt it more precisely to the target task, is an example of transfer learning. For example: a model trained to recognise cars in images can be reused (via fine-tuning) to recognise lorries, with little additional data. This reflects a form of learning by analogy, similar to human reasoning.

Few-shot / Zero-shot Learning

A learning strategy aimed at enabling a model to generalise to new classes or tasks from a very limited number of annotated examples (few-shot) or without any direct examples (zero-shot) in order to reduce the cost of data collection and annotation.

Online Learning

A learning technique in which the model is updated incrementally as data arrives, without retraining on the entire training dataset. Particularly useful in embedded AI for processing continuous streams (sensors, signals, events) and adapting in real time under resource constraints.

Adaptive Learning

The ability of a model or system to adjust to changes in context, distribution or environment. It may rely on online learning, but also includes broader mechanisms (drift detection, domain adaptation, meta-learning). Essential for maintaining the performance and robustness of embedded AI in evolving real-world conditions.

3.1.2. Connectionist models

Neural networks and deep learning now form the algorithmic foundation of many embedded AI applications, particularly for perception, signal analysis and automated decision-making. Their ability to learn complex representations from raw data makes them powerful tools, but their implementation in embedded systems requires specific adaptations to meet constraints relating to computation, memory, latency and energy consumption.

Artificial Neural Network (ANN)

A computational model inspired by the functioning of the human brain. It consists of interconnected artificial neurons, organised into layers (input, hidden, output), which transform input data into output via mathematical calculations. Each neuron receives input signals, weights them, transforms them via an activation function, and then transmits the result. The network learns by adjusting the connection weights to minimise the error between the predicted output and the actual output.

Deep Learning

A subcategory of machine learning that uses deep artificial neural networks (with multiple layers, e.g. CNN, RNN) to automatically extract complex representations from raw data. It is particularly suited to unstructured data such as images, sounds or text.

3.1.3. Knowledge representation and symbolic AI

Knowledge-based systems constitute an approach to artificial intelligence based on the explicit representation of knowledge and on logical inference mechanisms. In the context of embedded AI, these systems provide reasoning, traceability and explainability capabilities that are particularly suited to constrained or critical environments.

Knowledge-based system or Expert System

An information processing system that helps solve problems in a specific application domain by making inferences from a knowledge base. The term ‘knowledge-based system’ is sometimes used as a synonym for ‘expert system’, which is normally restricted to expert knowledge. Some knowledge-based systems are capable of learning. (ref. definition [Confiance.ai](#)).

Knowledge Base

A database containing inference rules and information relating to human experience and

expertise in a particular field. In self-learning systems, the knowledge base also contains information derived from the resolution of previous problems. (definition: [Confiance.ai](#)).

Knowledge Graphs

In the field of knowledge representation, a knowledge graph is a knowledge base that models data in the form of graphs. Knowledge graphs facilitate data access and integration, add context and depth to other AI techniques, and serve as bridges between humans and systems, for example through human-readable explanations.

Resource Description Framework (RDF)

A graph model designed to formally describe web resources and their metadata, in order to enable the automatic processing of these descriptions. Developed by the W3C, RDF is the foundational language of the Semantic Web. An RDF triplet is an association (subject, predicate, object): the subject represents the resource to be described; the predicate represents a type of property applicable to that resource; the object represents a piece of data or another resource: this is the value of the property.

Ontology

A data model containing concepts and relationships that enable the modelling of a body of knowledge within a given domain. The concepts are organised into a graph, where the relationships may be semantic relationships or subsumption relationships.

Web Ontology Language (OWL)

A knowledge representation language built on the RDF data model. It provides the means to define structured web ontologies. A W3C recommendation for the Semantic Web since 2012.

3.2. Models for embedded AI

AI models are mathematical and algorithmic representations learnt from data, used by an embedded AI system to perform inference tasks locally, such as perception, classification, prediction or decision-making. In an embedded context, a model is designed, adapted or optimised so that it can be executed on platforms with constraints in terms of computing power, memory, energy and latency, whilst ensuring controlled, reproducible behaviour that is compatible with the system's operational

3.2.1. The main AI models

AI models constitute the algorithmic building blocks that enable embedded systems to perceive, analyse and interpret data from sensors or time series. In the context of embedded AI, the choice of model directly determines the system's performance, latency, energy consumption and ability to operate in real time on constrained platforms.

Convolutional Neural Network (CNN)

A neural network specialised in processing visual data (images, videos, spatial signals). It generally combines three types of layers: convolution layers that extract local features via filters, pooling layers that reduce dimensionality whilst preserving relevant information, and fully connected layers that produce the final output. This hierarchical structure enables the network to progressively learn patterns of increasing complexity, from simple visual elements to complex

concepts. They are particularly effective for applications such as image classification, object detection or face recognition, and can also be adapted for time series processing.

Recurrent Neural Network (RNN)

A neural network designed for processing sequential or temporal data. RNNs are characterised by their ability to retain past information through recurrent connections, enabling them to use previous outputs as inputs for subsequent stages. The most widely used architectures, LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit), were developed to improve long-term memory management.

Attention-based model

A family of neural network models incorporating an attention mechanism, whose role is to dynamically assign importance weights to different parts of an input (words, time points, image regions, etc.). This mechanism allows the model to focus on the information most relevant to a given task and to better capture dependencies, including long-range ones. Attention mechanisms can be integrated into various architectures (recurrent, convolutional or other networks) and are widely used in natural language processing, computer vision and time series analysis.

Transformer

A neural network architecture belonging to the family of attention-based models. Transformers process data in parallel, which improves learning efficiency and the modelling of long-range dependencies. Initially developed for natural language processing, they have been extended to other domains such as vision (Vision Transformers – ViT), audio and multimodal tasks. Optimised and compact variants (e.g. TinyBERT, MobileBERT, MobileViT) enable their deployment on embedded AI systems subject to computational, memory and energy consumption constraints.

Spiking Neural Network (SNN)

A bio-inspired neural network in which neurons communicate via discrete spikes rather than continuous activations. Neurons generate spikes when their membrane potential exceeds a threshold, creating an event-driven, asynchronous communication system comparable to biological neurons. These networks process information in a temporal and event-driven manner, making them particularly effective for processing sensory data and real-time tasks.

3.2.2. Other models

Beyond the most commonly used neural models, embedded AI also utilises other families of models designed to meet specific needs for processing, reasoning or dynamic adaptation. These models are distinguished by their ability to represent complex structures, temporal dependencies or probabilistic relationships, whilst offering various trade-offs between expressiveness, efficiency and robustness.

Large Language Model (LLM)

An AI model, generally based on Transformer-type architectures, trained on vast text corpora to understand and generate natural language. In embedded AI, LLMs are used in adapted forms (reduced, quantised or distilled models) to meet the computational, memory and latency constraints of embedded platforms.

Small Language Model (SLM)

A compact language model, designed for specific tasks and requiring fewer resources than large language models. In embedded AI, SLMs are favoured for real-time local inference, thanks to their limited memory footprint, controlled latency and compatibility with constrained embedded platforms.

Liquid Neural Network (LNN)

A recurrent neural network whose internal dynamics are described by time-dependent differential equations. Unlike classical recurrent neural networks (RNNs), these networks adapt their parameters and behaviour according to the input signal, which gives them great flexibility and robustness. They enable the modelling of complex dynamic systems with few neurons and are particularly well-suited to embedded applications in robotics, adaptive control and the analysis of non-stationary signals.

Graph Neural Network (GNN)

A neural network designed to learn from data structured as graphs. Each node aggregates information from its neighbours to construct an overall representation. GNNs are used for modelling sensor networks, robotics and event data processing.

State Space Model (SSM)

A neural network representing temporal data through a sequence of hidden states that evolve according to differential equations. A state space model maintains a continuous internal state that updates based on received inputs, enabling the capture of complex temporal dependencies and long-term dynamics.

Bayesian Network

A probabilistic model that uses Bayesian inference for probability calculations using a directed acyclic graph. (definition: [Confiance.ai](#)).

3.3. Optimisation techniques for embedded AI

Embedded systems have strict constraints in terms of memory, computing power, latency and energy consumption. In this context, the deployment of classical artificial intelligence models, such as deep neural networks, requires tailored optimisation techniques to ensure efficient execution whilst maintaining acceptable performance.

These techniques cover several complementary areas:

- **Numerical optimisations**, aimed at reducing the precision of weights and activations (**quantisation**), in order to reduce the memory footprint and speed up inference.
- **Compression optimisations**, which reduce the number of parameters or the complexity of models.
- **Architectural optimisation**, encompassing approaches that enable the model structure to be defined or adapted to hardware and application constraints.

3.3.1. Numerical optimisations

Numerical optimisations are a key lever for adapting artificial intelligence models to the constraints of embedded systems. By acting directly on the representation and precision of calculations, they enable a reduction in memory footprint, faster inference and lower energy consumption, whilst maintaining a level of performance compatible with application requirements.

Quantisation

A method that involves reducing the numerical precision of the weights and activations of an AI model, by switching from floating-point formats (e.g. 32-bit) to more compact integer formats (e.g. 8-bit, 4-bit, or even binary). It reduces memory size, speeds up inference, and lowers energy consumption, whilst maintaining acceptable performance. Essential in embedded systems for deployment on microcontrollers, NPUs and DSPs, where resources are limited.

Mixed-Precision Quantisation

A technique combining several levels of numerical precision within a single model (e.g. FP32, FP16, INT8) to adapt the speed, memory consumption and accuracy of predictions according to the sensitivity of each layer.

Binarisation

An extreme form of quantisation where weights and sometimes activations take only two distinct values (e.g. $-1/+1$ or $0/1$), thereby replacing multiplications with basic logical operations.

Ternarisation

A variant of binarisation allowing weights to take three possible values (e.g. $-1, 0, +1$).

Post-training quantisation (PTQ)

A set of techniques applying quantisation after the model has been fully trained.

Quantisation Aware Training (QAT)

A set of techniques that integrate quantisation during training, allowing the model to adapt to the loss of accuracy and achieve better performance.

3.3.2. Compression optimisations

Compression optimisations aim to reduce the size and complexity of artificial intelligence models to make them compatible with the constraints of embedded systems. By reducing the number of parameters to be stored and processed, these techniques help to limit memory footprint, speed up execution and reduce energy consumption, whilst maintaining acceptable functional performance.

Pruning

A method that involves removing connections or neurons deemed non-essential in a neural network, by exploiting the natural sparsity of the weights. This reduces the model's complexity, speeds up computations and saves memory, without any significant loss of accuracy; useful for making them compatible with edge architectures, for example.

Magnitude Pruning

Removal of weights with the lowest magnitude.

Structured pruning

Removal of structural elements from the network, such as convolutional filters, channels, or entire layers.

Low-Rank Factorisation

A technique that approximates weight matrices using lower-rank factors, reducing the number of parameters and computations.

Encoding-based Compression

A technique that reduces the size of the stored model by applying encoding techniques (e.g. Huffman coding, entropy coding) to represent parameters more compactly.

Compression by clustering – Weight Clustering

A technique that groups weights into cluster centres and then replaces them with indices. This is a form of vector compression that reduces memory usage.

Weight Sharing Compression

A technique that uses the same weights in multiple parts of the network, thereby reducing the memory required and the complexity of the model.

3.3.3. Architecture optimisation

Architectural optimisations aim to adapt the very structure of artificial intelligence models to meet the specific constraints of embedded systems. By modifying the organisation of networks, computation paths or execution mechanisms, these techniques enable an optimal balance to be struck between accuracy, latency, energy consumption and robustness.

Distillation

A method whereby a complex model (“teacher”) transfers its knowledge to a simpler model (“student”), teaching it to reproduce its predictions. It enables the performance of a heavy model to be retained whilst reducing its size and complexity.

Neural Architecture Search (NAS)

An automated method for finding the best network architecture based on constraints (latency, accuracy, memory). Embedded use: enables the generation of customised models for a given hardware target.

Early Exit

A technique that adds intermediate output points within the network, enabling early inference if confidence is sufficient. Embedded use: reduces latency and energy consumption, particularly useful in real-time systems.

Topological optimisation

A set of techniques that modify the model’s structure by merging several consecutive operations into a single optimised operation or by reordering their execution to reduce memory accesses and latency.

Tiling

A set of optimisation techniques involving the partitioning of computations and data into blocks of a size suited to the target hardware architecture, in order to improve memory locality, maximise the use of fast memory and effectively exploit hardware parallelism.

4. System architecture for embedded AI

The architectures and concepts of embedded AI form the foundation upon which distributed, resource-efficient and mission-critical intelligent systems are built. They describe hardware organisations (specialised processors, real-time systems, neuromorphic computing), software and organisational architectures (edge computing, fog computing, AIoT, distributed systems, inference pipelines), as well as more advanced architectural concepts and patterns (digital twins, agent-based architectures, edge mesh or federated architectures).

This diversity reflects the hybrid nature of embedded AI:

- It combines significant physical constraints such as latency, energy consumption and memory with high software requirements in terms of performance, reliability and behavioural control.
- It also fits within hierarchical and distributed architectures (edge-fog-cloud), in which processing can be distributed between the periphery, gateways and central infrastructures.
- It incorporates emerging approaches, such as agent-based systems, federated computing and neuromorphic computing, paving the way for more autonomous, resilient and sovereign systems.

The aim of this chapter is to provide a clear and contextualised overview of the main architectures and concepts used in embedded AI, in order to facilitate their adoption by the industrial, academic and institutional communities.

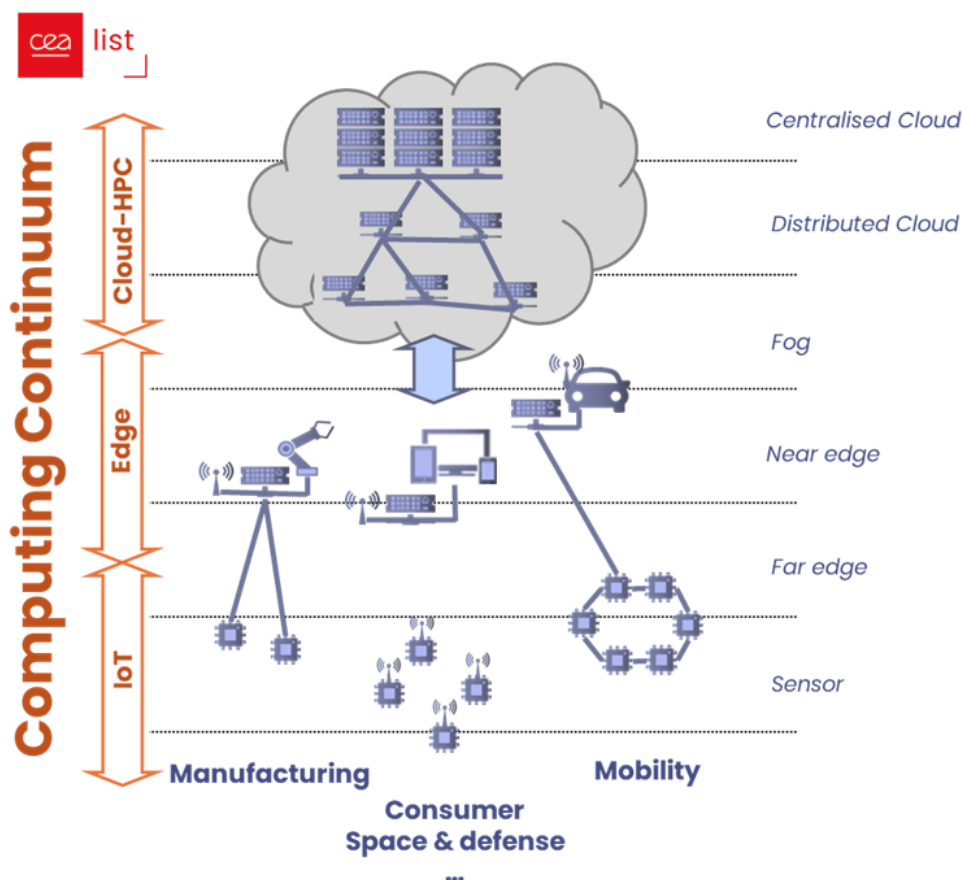


Figure 2. Diagram illustrating a distributed architecture extending from the HPC cloud to IoT sensors (© Denis Dutoit, List Tech Days 2025)

Computing Continuum

Continuous distribution of computing and data processing from embedded devices to the cloud, enabling the dynamic selection of the optimal location based on latency, energy consumption and bandwidth.

Cloud Computing

A centralised architecture where data and computations are processed in remote data centres. In the context of embedded AI, the cloud is used for model training, storage and service management, in conjunction with the edge and fog layers.

Distributed Systems

Distribution of processing across multiple system nodes to improve resilience and reduce latency.

Fog Computing

An intermediate layer between the edge and the cloud, enabling distributed and hierarchical data processing, often at the level of gateways or local networks.

Edge-to-cloud architecture

A hierarchical organisation where processing is distributed between the edge, the fog and the cloud, depending on constraints relating to latency, security and computing power.

Edge Computing

A decentralised computing paradigm where data processing is carried out as close as possible to the sensors or devices. This reduces latency and energy consumption, improves data privacy and reduces the load on the network.

Edge AI

A form of AI specifically run on edge devices, combining optimised models with embedded constraints.

Internet of Things (IoT)

A network of connected physical objects capable of collecting, processing and exchanging data via the internet or a local network. Embedded AI enables local decision-making, reducing reliance on the cloud and latency. Embedded systems are at the heart of the IoT and decentralised AI applications.

Artificial Intelligence of Things (AIoT)

The convergence of AI and IoT, where connected objects incorporate local learning and inference capabilities.

TinyML

Deployment of ML models on microcontrollers and embedded devices with very low power consumption, low cost and limited resources, for real-time local inference.

Cyber-Physical Systems (CPS)

Close integration between physical components and software, where AI provides real-time perception and control.

Real-Time Systems

Architectures guaranteeing deterministic responses within strict time constraints. In the context of embedded AI, they enable the execution of processing and decision-making algorithms with minimal latency, which is crucial for mission-critical applications.

Edge Mesh

A distributed architecture in which edge nodes (sensors, microcontrollers, cameras, robots, gateways) are interconnected in a mesh network, enabling them to communicate and cooperate directly without relying systematically on the cloud. Edge mesh promotes distributed collective intelligence, improves resilience and service continuity, and strengthens local data sovereignty. It is often associated with collaborative or federated learning approaches.

5. Software and tools for embedded AI

Having presented the architectures and concepts that underpin embedded AI, this chapter examines the software building blocks and tools essential for transforming these concepts into operational solutions. These elements constitute the development and runtime environment that ensures the portability, efficiency and control of AI models on heterogeneous embedded platforms.

The software and tools described in this chapter offer development interfaces tailored to the specific constraints of embedded systems, such as real-time operation, resource constraints and security requirements. They facilitate the integration, optimisation, execution and validation of embedded AI models, including in high-criticality systems.

However, these software building blocks cannot be separated from the specialised hardware components that support them — processors, accelerators, dedicated circuits and memory. The following chapter is devoted to these hardware foundations of embedded AI and highlights their decisive role in the performance, energy consumption and resource efficiency of systems.

This chapter provides a structured overview of the main software components of embedded AI, distinguishing in particular between languages, model formats, frameworks, development kits, hardware libraries and operating systems.

5.1. Languages

The programming languages used in embedded AI play a decisive role in the ability to design, integrate and execute artificial intelligence models on constrained platforms. They determine access to hardware, control over resources (computing power, memory, energy), as well as the performance and reliability of systems.

From optimised low-level development to the integration of prototyping and rapid deployment tools, these languages cover the entire embedded AI implementation cycle, adapting to the real-time, safety and portability requirements specific to embedded environments.

C / C++

Compiled languages offering fine-grained control over hardware and resources, widely used for the development of embedded AI, drivers and low-level optimisations on CPUs, GPUs and NPUs.

Rust

A modern compiled language guaranteeing memory safety and secure concurrency management, increasingly used for critical embedded applications and high-performance, reliable AI systems.

Python

A very popular interpreted language for rapid prototyping of AI models; in embedded AI, it is used via MicroPython on resource-constrained devices or via CPython on embedded platforms for running models and high-level scripts.

Java

An object-oriented language used on mobile and embedded platforms (Android, IoT), enabling

the development of applications that incorporate AI features, with automatic memory management and a rich ecosystem of libraries.

JavaScript

An interpreted language primarily used in web environments and certain embedded systems with a JavaScript engine, enabling the execution of AI models on the client side and the rapid deployment of interactive AI applications.

WebAssembly (Wasm)

A portable and secure binary format serving as a compilation target for languages such as C, C++ or Rust, enabling the execution of high-performance compiled code in web or embedded environments.

5.2. Model management

Model formats are a key element in the embedded AI deployment chain. They determine how models are represented, exchanged and integrated between the training, optimisation and execution phases on heterogeneous embedded platforms.

By promoting interoperability, portability and model optimisation, these formats enable AI architectures to be adapted to the specific constraints of embedded systems, such as resource limitations, performance requirements, certification and hardware compatibility.

5.2.1. Model formats

In the field of artificial intelligence, a file format for a model defines how weights, network structures and associated metadata are stored and exchanged. These formats play a key role in the deployment, portability and optimisation of models, facilitating their use across different frameworks, hardware platforms and environments.

Open Neural Network Exchange (ONNX)

An open, standard interoperability format for serialising and transferring machine learning models between different deep learning frameworks. It offers an abstract, framework-independent representation, acting as a standardised bridge between training and deployment. It facilitates portability across various hardware platforms (CPUs, GPUs, mobile devices, edge) thanks to its support by numerous inference runtimes optimised for embedded systems. A working group, named S-ONNX, has been set up to further develop the format and better address the constraints associated with certification processes.

Neural Network Exchange Format (NNEF)

An open standard enabling the exchange of neural network models between training and inference tools. Designed for interoperability on embedded systems, it facilitates optimised deployment across various hardware platforms (ARM CPUs, mobile GPUs, DSPs, FPGAs, ASICs).

GPT-Generated Unified Format (GGUF)

A binary format designed to efficiently store language models and their metadata. It natively supports quantisation (4-bit, 8-bit) and partial loading into memory, facilitating the deployment of LLMs on resource-constrained embedded devices (edge devices, smartphones).

Neuromorphic Intermediate Representation (NIR)

A standard designed to facilitate interoperability between different neuromorphic platforms. It defines a set of event-driven computational primitives, enabling models to be transferred to neuromorphic processors (such as Intel Loihi and SpiNNaker) or specialised accelerators.

5.2.2. Model libraries

Model libraries are a key entry point for the development of embedded AI solutions, providing pre-trained models that can be selected, evaluated and adapted to specific hardware and application contexts. They help accelerate the prototyping and deployment phases whilst facilitating the reuse of proven models.

Model Zoo

A collection of pre-trained AI models. *Model zoo* facilitate the selection and adaptation of models. Examples: [ONNX Model Zoo](#), [OpenVINO Model Zoo](#), [Qualcomm AI Hub](#) or [Qualcomm on Hugging Face](#), [STMicroelectronics on Hugging Face](#), [Eclipse Aidge on Hugging Face](#)

Hugging Face

An open source collaborative AI platform offering a vast model zoo hosting millions of pre-trained models.

Model Card

A non-standardised document generally describing the technical characteristics, performance, limitations, potential biases and recommended use cases of an AI model. It ensures transparency and guides developers in the appropriate selection and deployment of models, particularly for embedded applications.

5.3. Platforms and development kits

Platforms dedicated to embedded AI provide the structural environment for industrialising the development and deployment of AI models on resource-constrained systems. They orchestrate the entire software chain, from model design and training through to optimisation, integration and execution on heterogeneous hardware architectures.

By providing tools, libraries and methodologies tailored to the constraints of embedded systems, these platforms facilitate the portability, performance and reliability of AI solutions. They also play a key role in the traceability, validation and, where applicable, certification of systems, ensuring a consistent link between models, software tools and the hardware components on which they are deployed.

5.3.1. Platforms dedicated to embedded AI

Platform

A structured suite of software tools offering a coherent architecture for designing, training, optimising and deploying artificial intelligence models. It provides out-of-the-box functionality (data, model and metrics management) and enforces a development methodology. In the context of embedded AI, it simplifies deployment on hardware with limited resources.

Eclipse Aidge

An open source, collaborative platform hosted by the Eclipse Foundation, developed primarily by European academic and industrial stakeholders. This platform provides a comprehensive toolchain for designing, training, optimising and deploying embedded AI models. Aidge enables the generation of optimised and auditable source code for various hardware architectures, with the aim of facilitating deployment on heterogeneous architectures. It also offers native features for benchmarking and for meeting certification requirements (traceability, hardware fault tolerance, etc.).

TensorFlow, LiteRT

Developed by Google, TensorFlow is a comprehensive platform for designing, training, optimising and deploying artificial intelligence models. It provides data management tools, training modules and interfaces for inference on CPUs, GPUs or embedded devices. It is extensible via specialised libraries (e.g. TensorFlow Federated, TensorFlow Decision Forests) that target specific use cases, including on resource-constrained hardware (LiteRT).

PyTorch, ExecuTorch

Developed by Meta, PyTorch is an open-source platform designed for the creation, training and deployment of deep learning models. It enables the design of complex neural networks, the execution of tensor calculations, and the optimisation of models on CPUs or GPUs. ExecuTorch is a platform that complements PyTorch, specialising in the optimised inference of pre-trained models, enabling their efficient deployment on embedded or resource-constrained systems.

Edge Impulse

Acquired by Qualcomm, Edge Impulse is a platform dedicated to the creation, training and deployment of artificial intelligence models optimised for edge devices. It offers tools, in the form of a cloud platform, enabling users to collect data, design models and easily integrate them into resource-constrained devices such as microcontrollers or IoT sensors.

5.3.2. Development kits

Within the embedded AI ecosystem, hardware component suppliers generally provide their own development kits to facilitate the implementation and optimisation of models on their platforms. These kits typically include hardware-specific libraries, configuration tools and middleware layers.

Software Development Kit (SDK)

A set of tools and libraries made available to enable developers to create applications or interact with a specific platform, service or hardware. For example:

- **STM32Cube:** STMicroelectronics' kit for its STM32 microcontrollers.
- **JetPack SDK:** NVIDIA's kit for its Jetson modules with CUDA & TensorRT support
- **OpenVINO:** Intel's kit for its processors and accelerators.
- **Snapdragon Neural Processing SDK:** a kit from Qualcomm for its SoCs.

5.4. Hardware libraries

Hardware-optimised libraries play a key role in accelerating critical computations such as convolutions, matrix multiplication and activations. They provide high-performance implementations tailored to each type of hardware — microcontrollers, CPUs, GPUs, NPUs or FPGAs. These libraries are often integrated into the frameworks and runtimes mentioned above and used by compilers to generate optimised native code, enabling the efficient deployment of AI models. For example:

- **CMSIS-NN**: an open-source library optimised for ARM's Cortex-M processors
- **XNNPACK**: an open-source library optimised for ARM and x86 CPUs
- **cuDNN**: NVIDIA library for GPUs

5.5. Compilers

Compilers and optimisation pipelines play a central role in the implementation of embedded AI, ensuring the transformation of models and high-level code into executables optimised for constrained hardware platforms. They form the essential link between the algorithmic abstractions of AI and efficient execution on specific architectures such as microcontrollers, embedded processors, accelerators or FPGAs.

By combining intermediate representations, automatic optimisations and targeted techniques (quantisation, operator fusion, vectorisation, auto-tuning), these tools enable AI models to be adapted to the performance, power consumption and latency constraints specific to embedded systems, whilst ensuring portability and control over runtime behaviour.

Compiler

A tool that translates source code into native executable code for a specific hardware platform. It generally performs optimisations (topological optimisation such as layer fusion, vectorisation, pruning) to improve execution.

Low Level Virtual Machine (LLVM)

An open-source compilation infrastructure that provides a modular compiler and optimisation tools to generate native code on various architectures (x86, ARM, RISC-V, GPU). LLVM enables the intermediate representation of programmes, facilitating the analysis, transformation and optimisation of code for various targets, including embedded systems and AI.

TVM

An open-source automatic compilation framework designed to transform pre-trained AI models into optimised low-level code, deployable on a wide variety of hardware — from the cloud to edge devices (CPUs, GPUs, FPGAs, microcontrollers). It generates low-level code using LLVM or CUDA, whilst applying advanced optimisation techniques such as quantisation, operator fusion and pruning.

Multi-Level Intermediate Representation (MLIR)

An open-source compiler infrastructure providing a generic and extensible architecture for the intermediate representation of code at multiple levels of abstraction. MLIR enables the

construction of compilers and optimisation tools for machine learning, facilitating the translation of high-level models into code optimised for various hardware targets (CPUs, GPUs, neuromorphic accelerators). Its multi-level nature allows for progressive optimisations, from the semantic level of the model down to machine instructions.

Accelerated Linear Algebra (XLA)

An MLIR-based compiler developed by Google for TensorFlow and JAX, which transforms computational graphs into code optimised for CPUs, GPUs or TPUs by applying graph fusion and eliminating redundant operations.

There are also dedicated compilers for certain hardware targets, designed to optimise model execution on specific hardware as effectively as possible.

For example, these include:

ARM-GCC

A widely used compiler for ARM microcontrollers and processors, offering optimisations for speed and memory size.

FINN

A compiler developed by Xilinx, specifically designed to generate code and pipelines optimised for FPGAs and AMD accelerators.

CUDA NVCC

Compiler for NVIDIA platforms such as Jetson, Orin or RTX GPUs, often used in conjunction with TensorRT or DeepStream for AI inference. Converts CUDA kernels into highly optimised executable code for GPUs. In addition to topological optimisation techniques, compilers generally rely on specific optimisation techniques

Auto-tuning

A technique where the compiler or runtime **tests different computation configurations and kernel parameters** to select the combination that maximises performance on the target hardware (CPU, GPU, NPU).

Vectorisation

An optimisation that transforms scalar operations into operations on **vectors or data blocks**, enabling the use of SIMD instructions and accelerating computations on CPUs or GPUs.

Loop unrolling

A technique that replaces a loop with several consecutive non-iterative iterations to reduce the cost of control instructions and improve parallelisation.

5.6. Runtime

Inference engines constitute the software layer responsible for executing artificial intelligence models on embedded platforms. They ensure the loading of models, their efficient execution and the optimal utilisation of available hardware resources, taking into account the latency, power consumption and numerical precision constraints specific to embedded systems.

By leveraging optimisation mechanisms and controlled access to the underlying hardware, inference engines play a central role in the performance and reliability of embedded AI systems. They enable models from different frameworks to be made operational, whilst ensuring their portability and integration into real-time or mission-critical environments.

Inference engine or Runtime

A software layer responsible for loading and executing an AI model in a given environment (cloud or embedded), potentially relying on the OS to access the hardware. It optimises resource usage (CPU, GPU, memory), latency and sometimes numerical **precision** (FP32, FP16, INT8), to enable the model to operate efficiently.

Software platforms dedicated to embedded AI generally offer a runtime. The following runtimes are also worth mentioning:

ONNX Runtime

An open-source inference engine designed for inferencing machine learning models in ONNX format. It offers optimised (quantisation, topological optimisation) and cross-platform execution, compatible with various hardware architectures (CPU, GPU, accelerators).

TensorRT

An optimised inference engine, integrated into embedded frameworks such as JetPack SDK or DeepStream, enabling the optimisation (quantisation, topological optimisation, pruning) and execution of AI models (from PyTorch, TensorFlow, ONNX, etc.) on NVIDIA GPUs, particularly those designed for embedded systems (Jetson, Orin) Xavier, or RTX) in embedded or real-time environments (Jetson, Orin, Xavier, or RTX).

5.7. Operating systems

Operating systems constitute the fundamental software layer enabling the execution of embedded AI applications in direct interaction with the hardware. They determine how computing, memory and peripheral resources are allocated and managed, directly influencing the performance, latency and reliability of locally performed inferences.

Depending on the level of constraints and time requirements, embedded AI can rely on bare-metal environments, embedded Linux distributions or real-time operating systems. These choices determine the system's ability to meet real-time, power efficiency and safety requirements.

Embedded Operating System

Basic software that manages the hardware resources of an embedded system and provides the services necessary for running embedded AI applications under real-time, memory and power constraints.

Micro-ROS

A lightweight version of ROS 2 adapted for microcontrollers, enabling the integration of embedded AI perception and decision-making functions into systems with very limited resources.

5.7.1. Linux operating systems

Yocto Project

A set of open-source tools for creating customised and optimised Linux distributions for embedded systems, reducing the size of the operating system and memory footprint according to the specific needs of the application.

Android / Android Things

A Linux-based operating system developed by Google; Android is designed for smartphones and tablets, whilst Android Things is a version optimised for embedded devices and the IoT, enabling the Android ecosystem to be used on microcontrollers and SoCs for connected objects and smart devices.

Ubuntu Core

A modular and secure embedded Linux distribution, designed for connected and edge devices, facilitating the deployment and updating of software components, including embedded AI applications.

Buildroot

A set of tools for generating minimalist, optimised embedded Linux systems, suitable for deploying embedded AI pipelines on resource-constrained platforms.

5.7.2. Real-time operating systems

Real-Time Operating System (RTOS)

A specialised operating system designed to execute tasks with strict time constraints, ensuring that certain operations occur within a specified and predictable timeframe.

Free RTOS

A lightweight open-source real-time operating system (RTOS) designed for microcontrollers and embedded processors, providing multitasking, queue management, semaphores and timers for real-time critical applications.

Zephyr RTOS

An open-source real-time operating system (RTOS) for IoT devices and microcontrollers, offering native support for peripherals, network communications and small embedded AI applications with an optimised memory footprint.

Eclipse ThreadX

A comprehensive, certified open-source real-time operating system (RTOS) for mission-critical applications, designed specifically for resource-constrained embedded systems and Internet of Things (IoT) devices (microcontrollers). Known as Azure RTOS at Microsoft, the project has been transferred to the **Eclipse Foundation**.

ROS

A set of libraries, tools and services that facilitate the development of software for robots (sensors, actuators, perception, planning, etc.)

Mbed OS

A lightweight real-time operating system for microcontrollers, facilitating the development of ultra-constrained IoT and embedded AI applications, with fine-grained power and peripheral management.

Robotics middleware – ROS 2

A distributed software framework that facilitates the integration of sensors, AI algorithms and actuators into embedded robotic systems, with guarantees of real-time communication and reliability.

6. Hardware for Embedded AI

Hardware forms the physical foundation of embedded AI. It determines the computational capabilities, power consumption, latency and reliability of systems. Without a precise alignment between hardware components and software models, embedded AI cannot meet the performance, power efficiency and security constraints specific to critical environments.

This chapter explores the main hardware building blocks:

- Specialised processors and accelerators (CPU, GPU, FPGA, ASIC, NPU) that ensure the execution of AI models in embedded environments.
- Dedicated circuits and architectures (neuromorphic, heterogeneous, real-time) that optimise performance and energy efficiency.
- Memory and interconnects that ensure smooth data flows and rapid processing.

The aim is to provide a clear and contextualised overview of the hardware used in embedded AI, highlighting its role in implementing the architectures presented in Chapter "[System architecture for embedded AI](#)" and its integration with the software tools in Chapter "[Software and tools for embedded AI](#)".

6.1. Hardware architectures and computing paradigms

Hardware architectures and computing paradigms form the physical and computational foundations of embedded AI. They determine how models are represented, executed and optimised on constrained platforms, directly influencing performance, energy consumption and the ability to meet real-time requirements.

In the context of embedded AI, these concepts cover both the fundamental interfaces between software and hardware, processor design philosophies, parallel and vector computing mechanisms, and the data structures manipulated by the models. They provide the necessary framework for understanding how AI computations are actually performed on heterogeneous embedded targets, from microcontrollers to specialised system-on-chip devices.

Instruction Set Architecture (ISA)

A specification defining the set of instructions, registers, addressing modes and execution model visible to the software, constituting the interface between the hardware and compilers, operating systems and AI runtimes.

Reduced Instruction Set Computer (RISC)

A processor design philosophy based on a simple and regular instruction set, favouring pipelining, parallelism and energy efficiency, particularly suited to embedded systems.

Open Hardware

A hardware design approach in which specifications, interfaces and sometimes hardware descriptions are open and freely accessible, allowing for the modification, redistribution and customisation of hardware architectures, particularly for AI processors and accelerators.

RISC-V

An open, royalty-free ISA based on RISC principles and designed in a modular fashion, allowing the addition of specialised extensions (vector processing, security, AI acceleration) to cover a wide range of processors, from microcontrollers to SoCs for embedded AI.

ARM

A family of proprietary RISC processor architectures, widely used in embedded and mobile systems, benefiting from a mature software ecosystem and extensions dedicated to vector computing and AI.

Tensor Operator Set Architecture (TOSA)

A specification for high-level tensor operators proposed by ARM and used as an intermediate format in compilation pipelines to express neural network operations independently of the hardware.

Tensor

A multidimensional data structure used to represent the inputs, weights and outputs of a deep learning model. A tensor can be viewed as a generalisation of matrices to an arbitrary number of dimensions (for example, a 4D tensor for a batch of RGB images).

Data layout

The in-memory organisation of a tensor's dimensions. The two most common formats are **NCHW** (batch, channels, height, width) and **NHWC** (batch, height, width, channels). The choice of layout has a significant impact on performance, depending on the hardware architecture and the hardware library used.

- **Batch:** A set of samples processed simultaneously by the model during a training or inference iteration. Batch processing improves computational efficiency by exploiting hardware parallelism.
- **Channels:** A dimension representing the different components of a dataset. For an image, this generally corresponds to **the colour channels** (e.g. R, G, B). For an audio signal or sensor data, this may represent different sources or measurement dimensions.
- **Height:** The vertical size of two-dimensional data (for example, the number of vertical pixels in an image).
- **Width:** The horizontal size of a two-dimensional data set (e.g. the number of horizontal pixels in an image).

Multiply-Accumulate (MAC)

A computational unit that performs a multiplication followed by an accumulation ($a \times b + c$) in a single operation, a fundamental building block of matrix and convolutional computations in AI.

Vector Computation

A computing method that utilises vector registers and instructions to accelerate parallel operations.

Single Instruction Multiple Data (SIMD)

An execution paradigm in which a single instruction operates on multiple data items

simultaneously.

6.2. Specialised processors and accelerators

Specialised processors and accelerators form the hardware foundation on which embedded AI runs. They determine the systems' ability to process artificial intelligence models locally under strict constraints regarding performance, latency, energy consumption and cost.

In the context of embedded AI, the choice and combination of these computing units determine the possible architectures, the applicable optimisation techniques and the addressable use cases. The concepts presented in this chapter provide an understanding of the role and complementarity of the different types of processors and accelerators deployed for the efficient execution of AI as close as possible to the physical system.

Embedded AI accelerator

A hardware computing unit dedicated to the efficient execution of AI models (neural networks, tensor computations), optimised to reduce latency, power consumption and memory footprint on embedded platforms.

ARM Ethos Neural Accelerator

A family of hardware accelerators designed for the execution of embedded AI models on ARM architectures, optimised for quantised neural networks and low-power inference.

Embedded TPU Accelerator – Edge TPU

Specialised hardware accelerator for the inference of embedded AI models, optimised for tensor operations and quantisation, enabling fast, energy-efficient local execution on edge devices.

Kendryte Embedded AI SoC

A family of system-on-chip solutions integrating general-purpose compute cores and dedicated AI accelerators, demonstrating the tight integration of AI computing into low-power embedded platforms.

Central Processing Unit (CPU)

General-purpose processor that executes logical and arithmetic instructions.

Microcontroller Unit (MCU)

A low-power integrated processor, suitable for simple tasks or ultra-lightweight AI models.

Graphics Processing Unit (GPU)

A parallel processor optimised for matrix calculations, used for AI training and inference.

Neural Processing Unit (NPU)

General-purpose hardware accelerator designed to efficiently run deep learning models, not just CNNs. It optimises operations typical of neural networks: matrix multiplications, convolutions, activations, quantisation, etc.

Convolutional Neural Network Accelerator (CNN Accelerator)

This convolutional neural network accelerator is a hardware block dedicated specifically to

convolution operations and the layers typical of CNNs. It is more specialised than an NPU, often simpler and more energy-efficient, but less versatile.

Digital Signal Processor (DSP)

A processor specialised in digital signal processing (audio, sensors).

Tensor Processing Unit (TPU)

An AI accelerator designed by Google for large-scale tensor computations.

Vision Processing Unit (VPU)

A processor dedicated to embedded computer vision.

Neuromorphic processor

A circuit inspired by the functioning of the human brain, it is based on architectures composed of hardware neurons and synapses, often implemented as spiking circuits that process information asynchronously and in a massively parallel manner. For embedded systems, these architectures offer very high energy efficiency, low latency and local data processing, making them particularly well-suited to power-constrained and real-time applications, such as perception and autonomous control.

6.3. Specialised circuits and embedded architectures

Integrated hardware architectures define how computing, memory and interconnection components are physically organised to run embedded AI applications. They determine the overall performance, energy efficiency, compactness and integration capability of intelligent systems in constrained environments.

In the context of embedded AI, these architectures encompass both physical and modular platforms as well as specialised or reconfigurable integrated circuits. They offer different trade-offs between flexibility, specialisation and performance, and are a key enabler for adapting AI solutions to the operational requirements of embedded systems.

Printed Circuit Board (PCB)

A printed circuit board on which electronic components (CPU, GPU, memory, interfaces, etc.) are mounted, serving as a physical substrate and interconnection for embedded systems and AI modules.

Field Programmable Gate Array (FPGA)

A hardware-reconfigurable circuit, enabling fine-tuning of AI functions.

Application-Specific Integrated Circuit (ASIC)

A chip dedicated to a specific task, highly efficient for embedded AI.

Massively Parallel Processor Array (MPPA)

A parallel architecture comprising hundreds of cores, suitable for mission-critical applications.

System on Chip (SoC)

An integrated circuit combining a CPU, GPU, NPU, memory and communication interfaces on a

single chip, offering a compact and high-performance solution.

Network-on-Chip (NoC)

Internal interconnect infrastructure within an SoC enabling rapid communication between processors, caches, accelerators and memory, essential for effectively utilising the memory hierarchy and maximising data throughput.

Heterogeneous Architecture

Hardware platforms combining several types of specialised computing units, processors and accelerators within a single system. This organisation allows each task to be assigned to the most efficient computing engine, optimising the trade-off between performance, power consumption and latency. For constrained embedded systems, heterogeneous architectures thus offer good energy efficiency, a degree of execution flexibility and the ability to meet real-time requirements.

Chiplet

A modular integrated circuit containing a well-defined subset of functionalities (e.g. processor cores, memory, input/output).

Computer on Module (CoM)

A compact electronic board integrating a processor, memory and interfaces, designed for integration into embedded systems; used for modular embedded AI architectures.

System on Module (SoM)

A more comprehensive embedded module than the CoM, often including specific components (GPU, FPGA, industrial interfaces, etc.); used for demanding embedded applications (vision, robotics, etc.).

Intelligence Processing Unit (IPU)

A parallel architecture designed for AI, optimised for computational graphs.

Reconfigurable Processing Unit (RPU)

Task-adaptable processor, often FPGA-based, for customising AI acceleration

6.4. Memory and interconnect

Communication, memory management and interconnection mechanisms play a central role in the efficient execution of embedded AI. They ensure the flow of data between sensors, processors, accelerators and models, whilst managing constraints relating to latency, bandwidth and energy consumption.

In the context of embedded AI, these elements constitute the invisible yet crucial infrastructure that enables the orchestration of computations, data storage and optimised data transfers. The concepts presented in this chapter provide the keys to understanding how middleware, memory and hardware interfaces contribute to the performance and reliability of embedded intelligent systems.

Middleware

Intermediate software that ensures communication, coordination and data management

between different applications, systems or layers within a computing environment. It provides common services (such as message, transaction or resource management) enabling software components to work together without direct dependencies.

Static and Dynamic RAM (SRAM and DRAM)

Fast random access memory for the temporary processing of AI data.

Non-Volatile Memory (NVM)

Memory that retains data even without a power supply. It is the opposite of RAM (volatile).

Flash / eMMC / UFS / NAND

Types of non-volatile memory (NVM) for storing AI models, firmware and logs.

Magnetoresistive and Resistive RAM (MRAM / ReRAM)

Emerging, non-volatile, fast and energy-efficient memory.

L1 / L2 / L3 Caches

Fast memories integrated into the processor to reduce latency in accessing data and instructions, organised into hierarchical levels (L1 close to the core, L3 shared).

Direct Memory Access (DMA)

A hardware mechanism enabling data transfer between memory and peripherals or accelerators without involving the CPU, thereby reducing latency and power consumption.

In-Memory Computing (IMC)

A hardware paradigm in which computation (often matrix or logical operations) is performed directly within memory cells, minimising data transfers and power consumption, and improving performance for embedded AI inference.

PCIe / MIPI / UART / SPI / I2C communication interfaces

Interfaces enabling data transfer and communication between processors, sensors and AI modules in embedded systems.

7. Communications & connectivity in embedded systems

Communication and connectivity are essential elements of embedded AI. They enable devices to collect, exchange and synchronise data, but also to cooperate in distributed environments, linking sensors, processors and infrastructure within an edge-fog-cloud continuum. This connectivity - which is simultaneously local, distributed and global - paves the way for the developments outlined in Chapter 8, where the embedded AI of tomorrow is emerging through new forms of intelligence, frugality and technological sovereignty.

In the context of embedded AI, connectivity must meet specific constraints:

- **Latency and real-time performance:** ensuring fast and reliable data exchange for critical applications.
- **Energy efficiency:** optimising protocols and data flows to limit device power consumption.
- **Security and reliability:** ensuring the protection of transmitted data and the robustness of communications against disruptions.
- **Interoperability:** enabling the integration of multiple standards and protocols (CAN, SPI, I²C, UART, MIPI, Ethernet, 5G, and others).

This chapter provides an overview of the main protocols, buses, networks and connectivity mechanisms used in embedded AI, outlining their role in connecting sensors, processors, distributed systems and edge-fog-cloud infrastructures.

7.1. Physical networks

Physical networks constitute the hardware infrastructure (wired or wireless) of embedded connectivity. They define how data flows between sensors, processors and infrastructures.

Internal buses

Universal standards for connecting sensors and processors over short distances, ubiquitous in the automotive, embedded electronics and industrial sectors due to their reliability and low cost. They ensure fast, reliable data exchange suited to the real-time constraints of embedded systems. For example: I²C, SPI, UART, CAN.

Industrial networks and Ethernet (Industrial Ethernet, PROFIBUS)

Wired or hybrid communication technologies designed for industrial environments, where reliability, robustness and determinism are essential. They facilitate data exchange between PLCs, sensors, actuators and control systems, often in environments subject to severe constraints (electromagnetic noise, vibrations, high temperatures). They guarantee stable, secure and predictable communications, which are essential for industrial embedded systems and real-time applications.

Short-range wireless networks

Local radio technologies, suited to constrained and low-power environments, widely used in home automation, smart buildings and wearable devices. They enable local communication

between sensors and gateways. They facilitate mobility and integration in constrained environments, with low energy consumption. For example: BLE, Zigbee and Thread.

Long-range wireless networks

Extended connectivity between devices and infrastructure, such as: Wi-Fi, LTE/4G, 5G URLLC, NB-IoT, LTE-M, LoRaWAN. 5G URLLC is essential for critical applications requiring low latency, whilst NB-IoT, LTE-M and LoRaWAN are designed for large-scale deployments of low-power connected devices. Depending on the protocol, they prioritise high bandwidth, coverage or energy efficiency, and enable edge-cloud integration.

Mesh networks

Mesh architectures enabling resilience and extended coverage, used in distributed sensor networks and smart urban environments. Examples include 6LoWPAN and Thread mesh.

7.2. Application protocols

Application protocols are software layers that define how data is structured, exchanged and interpreted over physical networks. They ensure the interoperability and consistency of distributed systems.

Lightweight IoT protocols

Protocols designed for the frugal and scalable transmission of sensor data, widely adopted in IoT networks.

Message Queuing Telemetry Transport (MQTT)

A very lightweight protocol based on the publish/subscribe model, designed for environments with limited bandwidth and power, and widely adopted in the IoT (home automation, distributed sensors, telemetry).

CoAP (Constrained Application Protocol)

A protocol inspired by HTTP, but optimised for devices with limited resources. It operates over UDP, using a request/response model. It is used in industrial automation, environmental monitoring and highly constrained sensors.

Advanced Message Queuing Protocol (AMQP)

A feature-rich, message-oriented protocol with delivery guarantees and security. It is heavier than MQTT or CoAP, and therefore less suitable for highly constrained devices; it is used in enterprise environments (finance, critical systems, cloud).

Modbus®

A historic protocol (dating from the 1970s), simple and robust, based on a client-server model, still widely used in industry, particularly for connecting programmable logic controllers (PLCs), sensors and actuators. Its simplicity and widespread adoption make it a universal standard, but it remains limited in terms of security and data richness.

Open Platform Communications Unified Architecture (OPC UA)

A modern, object-oriented protocol designed for interoperability and the standardisation of industrial data exchange. It is widely adopted in automation, Industry 4.0 and edge-cloud

architectures, as it enables the connection of heterogeneous systems. It is extensible, secure (incorporating encryption and authentication), and supports rich, hierarchical data models.

Edge-fog-cloud interoperability

Standardised mechanisms that streamline communication between heterogeneous layers (edge, fog, cloud) and ensure compatibility between distributed systems, equipment and services. They guarantee operational continuity in hybrid architectures and facilitate multi-technology integration. Examples: REST API, gRPC, OPC UA.

Model Context Protocol (MCP)

Application-level protocol designed to standardise interactions between AI agents and external tools (services, data sources, APIs, execution environments). MCP provides a structured framework for exposing capabilities, exchanging messages and orchestrating actions, enabling seamless integration of AI agents within distributed architectures (edge - fog - cloud). Created initially by Anthropic, MCP is now an open standard and is rapidly becoming the common protocol for agentic AI, much like REST became the foundational standard for the web.

REST

An HTTP-based communication interface using resources identified by URLs; it is universal and simple, and has established itself as the web and cloud standard for exposing and consuming services.

Google Remote Procedure Call (gRPC)

A modern protocol based on HTTP/2 and Protobuf, designed for remote function calls, which meets the needs of distributed architectures and microservices, with fast, efficient exchanges, including bidirectional streaming.

8. Advanced and emerging paradigms in embedded AI

Embedded AI is a rapidly evolving discipline, driven by algorithmic, hardware and architectural advances. Beyond traditional approaches to machine learning and embedded inference, new paradigms are emerging and opening up unprecedented possibilities for intelligent systems deployed on constrained platforms.

These paradigms do not redefine the foundations of artificial intelligence introduced in the previous chapters. They extend and enrich these concepts by exploring advanced trajectories specific to embedded AI, characterised by greater autonomy, enhanced integration with the physical world, and explicit consideration of resource constraints and environmental impact.

8.1. Embedded generative AI

The concepts associated with embedded generative AI describe the mechanisms enabling the local production of content and representations from trained models, within a context of limited resources. They cover the types of models used, the means of guiding generation, the ways in which generic models are adapted for embedded use, as well as the specific risks associated with autonomous content generation. These elements enable the characterisation of generative uses compatible with execution on embedded platforms and distinguish them from centralised approaches.

Generative AI

An approach to artificial intelligence aimed at producing new content (text, images, signals, code) based on trained models. In the context of embedded AI, generative AI is implemented in specialised and constrained forms, enabling local generation adapted to the computational, memory, latency and energy consumption limitations of embedded platforms.

General-Purpose AI System (GPAIS)

An AI system designed to be adaptable to a variety of tasks. In embedded AI, these systems are generally specialised, distilled or restricted to targeted functions so that they can be executed locally and integrated into specific embedded applications.

Foundation Model

A large-scale artificial intelligence model, trained in a self-supervised manner on vast datasets. In embedded AI, foundation models primarily serve as a starting point for compact or specialised variants, adapted for local inference on constrained systems.

Prompt Engineering

The process of designing and structuring queries to guide the behaviour of a generative model. In embedded AI, prompt engineering aims to maximise the relevance, robustness and predictability of generated outputs, whilst keeping computational and memory costs under control.

Retrieval-Augmented Generation (RAG)

A technique combining a generative model with a knowledge base search mechanism. In embedded AI, RAG relies on local, embedded or partially synchronised knowledge bases to limit dependence on remote connectivity whilst enriching the generated responses.

Embedding

A compact vector representation of objects such as words, phrases, images or signals, organised to reflect their semantic or functional similarities.

Token

A basic processing unit used by language models to represent fragments of text or symbolic data. In embedded AI, managing the number and size of tokens is a key factor in controlling latency and memory footprint.

Hallucination

The generation by a generative model of an incorrect, incoherent or factually unfounded response, presented as plausible. In embedded AI systems, hallucinations pose a critical risk, particularly in the fields of healthcare, transport and defence, and require mechanisms for mitigation, detection or validation.

Joint-Embedding Predictive Architecture (JEPA)

A self-supervised learning architecture designed to predict the future evolution of a state or scene in a latent space, rather than directly generating content. JEPA is particularly well-suited to embedded AI for building models of the world that enable anticipation, planning and adaptation to dynamic environments.

Hierarchic Reasoning Model (HRM) and Tiny Recursive Model (TRM)

Two small-scale model architectures, proposed by Sapiient Intelligence and Samsung respectively, which use hierarchical looping or recursive reasoning to solve problems. These models are particularly efficient in terms of space and computational time and may be of interest for embedded systems.

Embedded World Models

Internal representations enabling an embedded AI system to model the current state and likely evolution of its environment. These models support the anticipation and reasoning capabilities essential for autonomous systems.

8.2. Agent-based AI and autonomous systems

Concepts relating to agent-based AI describe the autonomous decision-making capabilities of intelligent embedded systems. They enable the formalisation of the concept of an agent, goal-oriented decision-making mechanisms, interactions between agents and distributed architectures. These concepts are used to describe embedded systems capable of acting proactively, in a coordinated and adaptive manner, in direct interaction with their environment, without permanent reliance on external supervision.

Agency

The ability of an AI system to initiate actions, make decisions and pursue objectives autonomously within a given environment, without continuous human supervision.

AI Agent

An autonomous software or hardware entity capable of perceiving its environment, reasoning on the basis of its observations, and acting to achieve a goal. In embedded AI, an agent is generally deployed on a constrained platform and acts in direct interaction with the physical world.

Multi-Agent Systems (MAS)

A set of agents interacting within a shared environment, through cooperation, coordination or negotiation. In embedded AI, these systems are used for collective robotics, intelligent sensor networks or distributed cooperative devices.

Agent-based architectures

Architectures based on autonomous agents capable of perception, planning and action, promoting distributed and context-aware decision-making on embedded systems.

Goal-Driven Approach

A paradigm in which an agent acts according to explicit goals, evaluating different courses of action and adapting its strategy in response to changes in the environment.

Perception–action loop

A continuous cycle linking environmental perception, decision-making and action, characteristic of autonomous embedded systems capable of adapting to their context in real time.

AI Orchestration

The coordination of multiple agents or AI models to perform complex tasks in a coherent manner. In embedded AI, orchestration primarily concerns distributed cooperative systems operating with limited resources and connectivity.

8.3. Physical AI and interaction with the real world

The concepts associated with physical AI concern the physical embodiment of artificial intelligence in embedded systems capable of perceiving and acting upon the real world. They enable the characterisation of forms of integration between algorithms, sensors and actuators, as well as tools for representing and simulating physical behaviour. These concepts are essential for describing systems where embedded AI directly influences physical and mechanical processes.

Physical AI

A form of AI integrated into hardware devices capable of perceiving, deciding and acting upon the real world. Physical AI combines sensory perception, reasoning and motor action, and is optimised to operate in real time under embedded constraints.

Digital Twin

A faithful virtual representation of a physical system enabling the simulation, analysis and

optimisation of its behaviour. In embedded AI, digital twins are used to anticipate behaviour, improve resilience and support predictive maintenance.

8.4. Eco-design and resource efficiency in embedded AI

The concepts of eco-design and resource efficiency formalise approaches to the design of embedded AI that explicitly incorporate environmental impacts and resource constraints. They describe the principles for assessing, reducing and controlling energy consumption, computational usage and data usage throughout the entire lifecycle of intelligent embedded systems. These concepts reflect the shift from the technical constraints of embedded systems to explicit design objectives.

Eco-design

An approach involving the integration of environmental impact reduction from the design stage of embedded AI systems, throughout their entire lifecycle, whilst taking into account computational, energy and hardware constraints.

Green AI

An approach to artificial intelligence aimed at reducing the environmental impact of AI models and systems, particularly in terms of energy consumption, carbon emissions and resource use.

Frugal AI

An approach to the design and use of AI based on resource efficiency, the limitation of data and computation, and the alignment of functional requirements with the deployed solution, particularly suited to constrained embedded systems.

Computational Sobriety

A principle aimed at limiting the amount of computation required to run a model, by prioritising lightweight architectures, efficient algorithms and appropriate inference strategies.

Hardware-software co-optimisation

An approach involving the co-design of the model, algorithms and hardware architecture to reduce power consumption, computational load and memory usage, whilst ensuring functional performance.

Thermal-Aware Design

Taking thermal constraints into account when designing models and inference pipelines, in order to prevent overheating, throttling and performance degradation.

Sustainable Design

Design aimed at maximising hardware lifespan, reparability, modularity and reusability, whilst minimising the overall environmental impact.

Circular economy

A strategy aimed at reducing waste and reusing components.

Environmental impact

The effect on the environment associated with the production, use and end-of-life of a system.

Life Cycle Assessment (LCA)

An assessment of the environmental impacts of a product or system across all stages, from manufacture to end-of-life.

Multi-criteria eco-assessment

An assessment method that simultaneously takes into account performance, energy, memory, carbon footprint, material costs and lifespan.

Constraint-Driven Design

A design approach guided by resource constraints (energy, memory, computation, thermal), rather than by algorithmic performance alone.

9. Testing, validation & certification of embedded AI

The rise of embedded AI, and even more so its future developments (generative AI, agent-based AI, edge-to-cloud, emerging architectures), necessitates ensuring **the reliability, security and compliance** of systems prior to their deployment in critical environments. Testing, validation and certification are therefore essential steps in building trust and ensuring industrial, regulatory and societal acceptance.

This chapter provides an overview of testing methods, validation metrics and certification schemes, highlighting their central role in the credibility and widespread adoption of embedded AI.

These practices address several major challenges:

- **Technical reliability:** verifying the robustness of models and architectures under real-world conditions.
- **Safety and security:** ensuring that embedded systems do not pose risks to users or the environment.
- **Regulatory compliance:** meeting European and international standards governing AI and critical systems.
- **Sector-specific certification:** obtaining specific labels or approvals (automotive, aerospace, medical, etc.) to validate the use of embedded AI.

9.1. Metrics

Metrics are a fundamental tool for characterising, comparing and managing the performance of embedded AI systems. They enable an objective assessment of a model's ability to operate effectively under constraints, taking into account not only the quality of the results produced, but also the hardware resources utilised and the time and energy requirements.

In the context of embedded AI, metrics play a key role in decisions regarding architecture, optimisation and deployment. They enable trade-offs to be made between accuracy, latency, energy consumption and memory usage, in order to ensure behaviour that is predictable, robust and compatible with embedded environments.

9.1.1. Memory metrics

Memory Footprint

Total amount of RAM required to load and run a model inference on the target hardware.

Model Size

The size of the file containing the AI model weights, expressed in kilobytes or megabytes (or even gigabytes). It determines the storage space required on the device.

Memory Bandwidth

The maximum data transfer rate between memory and the processor, expressed in gigabytes per

second (GB/s).

9.1.2. Temporal performance metrics

Latency

The delay between the receipt of data and the system producing a result, generally expressed in milliseconds (ms) or microseconds (μ s). It is critical in real-time embedded applications, where a delayed decision can compromise safety or performance.

Tail latency

The maximum latency observed for a given percentage of inferences (e.g. 99th percentile), used to ensure real-time predictability.

Worst-Case Execution Time (WCET)

The maximum guaranteed execution time that an AI model may take in the worst-case scenario on a given system. WCET ensures compliance with strict real-time constraints.

Throughput

The amount of data processed by the system over a given period, typically expressed in frames per second (fps), samples per second, or operations per second (ops/s). In the context of embedded AI, throughput measures inference speed and helps to assess the system's overall responsiveness.

9.1.3. Energy metrics

Power Consumption

The amount of energy consumed by the system during operation, expressed in milliwatts (mW) or watts (W). This is a major constraint in standalone or battery-powered devices.

Energy per inference

The amount of energy required to perform a single inference, expressed in microjoules (μ J) or millijoules (mJ). This metric is essential for assessing overall energy efficiency over the battery's lifespan.

9.1.4. Accuracy metrics

Accuracy

The proportion of correct predictions out of all predictions made by the model.

Top-1 accuracy

The proportion of predictions where the class predicted in first place matches the ground truth.

Top-5 accuracy

The proportion of predictions where the ground truth is included among the top 5 predicted classes.

Intersection over Union (IoU)

An evaluation metric used in object detection and segmentation to measure the overlap between

the area predicted by the model and the actual area. A high value indicates good localisation accuracy.

Mean Average Precision (mAP)

The average precision across different overlap thresholds (Intersection over Union), used to evaluate object detection models.

Mean Intersection over Union (mIoU)

The average of the overlap between the predicted areas and the actual areas, used to evaluate image segmentation models.

F1 Score – F-measure

A performance metric used to evaluate a classifier, which provides a balance (the harmonic mean) between recall and precision. (definition: Con fiance.ai)

9.2. Benchmarks

Benchmarks are reference tools for objectively evaluating and comparing the performance of embedded AI systems. They provide standardised frameworks for measuring the effectiveness of models, software chains and hardware platforms under conditions representative of embedded constraints.

In the context of edge AI, benchmarks play a key role in quantifying trade-offs between accuracy, latency, power consumption and resource usage. They help guide technology choices, compare heterogeneous solutions and provide reproducible results for evaluating hardware and inference pipelines deployed at the edge.

MLPerf Edge and MLPerf Tiny

Standardised benchmarks from MLCommons for evaluating the performance of AI inference on embedded platforms. MLPerf Edge targets edge systems (SoC, CPU, GPU, NPU) and measures latency, throughput, power consumption and accuracy, whilst MLPerf Tiny is dedicated to microcontrollers and TinyML environments with severe resource constraints.

These benchmarks serve as an industry reference for comparing embedded hardware and inference pipelines.

AIoT Benchmarks

A suite of benchmarks developed by BenchCouncil designed to evaluate the performance of artificial intelligence systems for the Internet of Things (AIoT). AIoTBench offers a set of representative workloads covering various embedded AI scenarios, such as computer vision, audio processing and sensor data analysis, and enables the comparison of models, software platforms and hardware capabilities.

10. Trustworthy Embedded AI

Trust in embedded AI is based on a set of attributes such as reliability, robustness, fairness, transparency, etc., which can contribute in particular to certification and qualification (see also the chapter on norms and standards). The terms below form the lexical foundation of trustworthy AI. They cover general requirements, processes (certification, qualification, audit), tools (SBOM, traceability, etc.) and criteria (evidence, robustness, explainability, etc.).

10.1. General considerations on ethics

The integration of artificial intelligence functions into embedded systems raises specific ethical issues related to decision-making autonomy, direct interaction with the physical world and local execution under constraints. Unlike centralised systems, embedded AI often operates in real time, in close proximity to users, infrastructure or critical environments, which heightens the requirements for control, accountability and trust.

This chapter addresses the principles and mechanisms for regulating the behaviour of embedded AI systems to ensure their responsible, transparent and controlled use. It focuses on preventing misuse, limiting bias, managing unforeseen situations and aligning automated decisions with the ethical, regulatory and societal frameworks applicable to intelligent embedded systems.

10.2. Reliability

Reliability is a fundamental pillar of trustworthy embedded AI, particularly when intelligent systems are integrated into critical, autonomous environments or those subject to high safety requirements. It encompasses all the properties that ensure an embedded AI system behaves in a predictable, correct and controlled manner throughout its operation.

In the context of embedded AI, reliability is not limited to software or hardware components: it also depends on the quality and provenance of the data, robustness against disruptions and attacks, and the system's ability to be maintained and secured over time. The concepts presented in this chapter provide a framework for analysing, designing and evaluating embedded AI systems capable of inspiring confidence under operational conditions.

Dependability

The ability to provide a service that can be relied upon. This implies:

- **Availability:** Availability of a proper service
- **Reliability:** continuity of a correct service,
- **Confidentiality:** absence of unauthorised disclosure of information,
- **Integrity:** absence of unauthorised alterations to the system,
- **Maintainability:** the ability to undergo modifications and repairs.

Security

The simultaneous existence of availability for authorised users only, confidentiality and integrity.

Robustness

The ability of an embedded AI system to function correctly despite disruptions, uncertainties or attacks.

Correctness

The degree to which a system, component or software element (code, documentation, specifications) is free from errors and correctly meets the defined requirements.

Data Quality

The set of properties (accuracy, consistency, completeness, freshness, noise) that determine the reliability of the inferences produced by an embedded AI system and its ability to operate safely and robustly in real-world conditions.

Data Provenance

The origin and history of the data used to train or feed an AI system, which is essential for ensuring its reliability and compliance.

10.3. Bias and fairness

Bias

An imbalance or systematic distortion in the results produced by a model, often caused by biases present in the data or algorithms.

Bias is not a one-off error, but a recurring pattern that reflects structural flaws in the system. It occurs when the decisions or predictions of an AI system are unfair, discriminatory or inaccurate towards certain groups or individuals.

This may be due to biased training data (e.g. over-representation of a group, lack of diversity), poorly designed algorithms that amplify unfair correlations, or human biases unintentionally incorporated during the design or annotation of the data.

Algorithmic fairness – Fairness

The principle that AI and machine learning systems must produce decisions that do not unfairly discriminate against individuals or groups, particularly on the basis of sensitive characteristics such as gender, ethnicity or age.

In practice, algorithmic fairness refers to the set of methods and criteria used to detect, measure and correct biases in models, so that results are as fair as possible across different groups.

10.4. Explainability and interpretability

Explainability and interpretability are essential components of trustworthy embedded AI, particularly when intelligent systems are integrated into critical, autonomous or regulated environments. They enable the understanding, analysis and justification of decisions produced locally by an AI model operating under hardware and time constraints.

XAI – Explainable and Interpretable AI (XAI)

A set of techniques enabling the understanding and justification of decisions made by AI.

Interpretability

The ability to understand and explain the decisions or predictions of an artificial intelligence model.

Explainability

The ability to provide a comprehensible justification for the decisions or predictions of an AI model, often required in qualification and certification processes.

According to NIST, explainability must adhere to four principles:

- **Explanations:** the system provides or contains evidence or reasons accompanying the results and/or processes.
- **Relevance:** the system provides explanations that are understandable to the intended users.
- **Accuracy:** the explanation correctly reflects the reason why the result was generated and/or accurately reflects the system's process.
- **Knowledge limits:** the system operates only under the conditions for which it was designed and when it achieves a sufficient level of confidence in its results.

10.5. Transparency and traceability

Traceability and auditability are fundamental requirements for the implementation of trustworthy embedded AI, particularly in sectors subject to regulatory, safety or certification constraints. They enable the documentation, monitoring and verification of all the constituent elements of an embedded AI system, from data and models to software and integrated components.

Traceability

The ability to track the origin, evolution and modifications of a model, software or hardware component, which is essential for qualification and certification. More specifically for embedded AI, this involves documenting the development, training and deployment stages of an AI system to ensure its transparency and auditability. Broadly speaking, it involves tracking the data, decisions and components of an AI system.

Audit

A systematic and independent examination of processes, evidence and results to confirm compliance with standards and regulations.

Auditability

The ability to verify, monitor and evaluate the operation of an AI system, particularly in the event of an incident or deviation; essential for embedded systems in regulated sectors.

Software Bill of Materials (SBOM)

A detailed inventory of software components (libraries, dependencies, modules) used in an embedded system. This is used for traceability, cybersecurity and regulatory compliance.

10.6. Cybersecurity

Cybersecurity is a major challenge for embedded AI systems, particularly when they are integrated into critical, autonomous or cyber-physical environments. These systems combine hardware, software and algorithmic components with sensitive data, which broadens their attack surface and exposes them to specific AI-related threats.

In the context of embedded AI, cybersecurity aims to protect models, data and inference mechanisms against attacks, whilst ensuring service continuity, decision integrity and the confidentiality of processed information. The concepts covered in this chapter provide a framework for analysing threats, designing appropriate defence mechanisms and strengthening the robustness of embedded AI systems against intentional or accidental attacks.

Cybersecurity of Embedded AI Systems

Protection of AI models, algorithms and data used in siloed sectors where critical systems predominate. Given that critical systems rely heavily on embedded AI systems for their autonomous operations, navigation and data processing, it is essential to ensure their security. Consequently, the cybersecurity of embedded AI systems involves protecting AI-based critical systems against hacking.

Embedded AI system for cybersecurity

AI systems to enhance the cybersecurity of critical systems in terms of security. Embedded AI systems can detect, prevent and respond to cyber threats in real time, which is crucial for the components of critical systems.

Threats

Potential security breach. Threats to cyber-physical systems include the destruction of information and/or other resources; the theft, deletion or loss of information and/or other resources; the corruption or alteration of information; the disclosure of information; and the disruption of services.

Mitigation Strategies

Implementation of protective measures aimed at reducing the vulnerability of assets by limiting the consequences of accidents and preventing security incidents.

Privacy by Design

An approach that integrates the protection of personal data from the earliest stages of designing an AI system. (e.g. biometric sensors).

Security by Design

A methodology that incorporates security mechanisms from the development phase of a system to prevent vulnerabilities.

AI safety layer

Mechanisms built in to monitor and correct unexpected or dangerous behaviour in an AI model.

AI Model Theft

The unauthorised extraction, reverse engineering or malicious exploitation of AI/ML models

deployed in critical systems. This can occur in real time during the operation or maintenance of critical systems.

Offensive evaluation – Red Teaming

A testing technique in which a team simulates attacks or malicious use to identify vulnerabilities in an AI system.

Adversarial Attack

A malicious action involving the slight modification of an input (for example, by adding imperceptible noise) in order to deceive a machine learning model. These perturbations are generally invisible to humans but cause the model to make a prediction error.

Adversarial Robustness

The ability of an AI model to withstand adversarial attacks, i.e. to maintain acceptable performance even when its inputs have been intentionally perturbed. Robustness is a sought-after property in critical or embedded systems exposed to potential manipulation.

Adversarial training

A defence technique involving the incorporation of misleading inputs into the training process to increase the model's robustness. The model thus learns to recognise and neutralise malicious perturbations, making it one of the most effective methods for improving resistance to adversarial attacks.

Secure Boot

A security mechanism ensuring that only authenticated and integrity-checked software can be executed when an embedded system boots up. In embedded AI, Secure Boot protects models, inference chains and critical components against unauthorised modifications from the moment power is applied.

Trusted Execution Environment (TEE)

A hardware-isolated environment within a processor that enables the secure execution of code and the storage of sensitive data. In embedded AI, the TEE is used to protect models, cryptographic keys and certain critical decisions against software or hardware attacks.

10.7. Qualification, certification, evidence

Qualification is a key step in demonstrating that an embedded AI system is fit for deployment and operation in a given context, particularly when used in critical, regulated or high-safety-stakes environments. It aims to establish, in a structured and documented manner, that the models, software and runtime environments meet the defined requirements.

Qualification

All activities aimed at demonstrating that a system, component or process is fit to fulfil its function in a given context. In embedded AI, this includes the validation of models and runtime environments.

Verifiability

The ability to assess the implementation of requirements in order to determine that they have

been met (according to ARP4754A).

Verification and Validation (V&V)

Verification that the system has been built correctly, in other words that it complies with the defined requirements, specifications and standards; validation consists of verifying that the right system has been built, i.e. that it effectively meets the needs and expectations of users.

Compliance

Demonstration that a characteristic or property of a product meets the stated requirements.

Certification

Third-party attestation regarding products, processes, systems or individuals. This may be a written guarantee that a system or component complies with its specified requirements and is acceptable for operational use. It may also be a formal demonstration that a system or component complies with the specified requirements and is acceptable for operational use. For embedded systems, this is the process of confirming that a system or component complies with its specified requirements and is acceptable for operational use.

Evidence

Documents, test results or reports that demonstrate a system's compliance with safety, robustness or regulatory requirements. "Evidence" is essential in certification processes.

Assurance Case

Argument used to justify certain claims regarding a system, based on evidence relating to both the system and the environment in which it operates. The claims may concern any property of the system, such as reliability or safety.

Accountability

The principle that those involved in the design and use of AI must be identifiable and accountable for the decisions made by the systems. This is particularly important for autonomous embedded systems, especially in the event of an incident or dispute. Accountability is a property that ensures that an entity's actions can be attributed solely to that entity.

11. Lifecycle and Governance of Embedded AI

This chapter covers the terminology used to describe the stages, processes and governance mechanisms governing the lifecycle of embedded AI systems. It focuses on the concepts necessary for the operational management of embedded AI, from design through to end of life, incorporating the traceability, validation and accountability requirements essential for the effective implementation of applicable regulatory frameworks.

The concepts presented aim to structure a coherent and controlled approach to embedded AI in constrained, often critical, systems where technical and organisational choices must be documented, justifiable and verifiable throughout the life cycle.

11.1. Lifecycle of Embedded AI

This section describes the various stages involved in designing, transforming, integrating and operating an embedded AI system on a constrained platform. It highlights the successive transitions between an offline-designed model and an operational embedded system, taking into account the adaptations required to ensure reliable, high-performance and controlled execution over time.

These concepts are essential for structuring a comprehensive and coherent view of the journey of an embedded AI model, from its design through to end-of-life, incorporating the hardware, operational and organisational realities specific to embedded systems.

Embedded AI Lifecycle

The set of successive phases covering the design, development, integration, validation, conversion, optimisation, deployment, operation, evolution and end-of-life of an embedded AI system, from the initial specifications through to the system's decommissioning.

Design Phase

The stage during which the functional objectives of the embedded AI, the hardware and operational constraints, as well as the performance, robustness, traceability and governance requirements necessary for its integration into an embedded system are defined.

Embedded Model Development

The process of designing, training and adapting an AI model intended for execution on an embedded platform, taking into account computational, memory, latency and deterministic behaviour constraints.

Model Conversion

The process of transforming a trained model into a format and representation compatible with the embedded device (runtime, inference library, hardware accelerator), with a view to its integration and local execution.

Model Optimisation

A set of techniques aimed at reducing the cost of running an embedded model (latency, power consumption, memory footprint), whilst minimising the impact on functional performance.

Deployment Artefact

A versioned set of elements required to run an embedded AI system on the final target (converted model, parameters, processing chains, configurations), constituting the traceable and versioned operational deployment unit.

System Integration

The phase during which the AI model, associated processing chains and monitoring mechanisms are integrated into the overall embedded system, interacting with the hardware, sensors and existing functions.

On-device Deployment

Commissioning of an on-device AI system on the final target, including configuration, final optimisation, verification of expected behaviour and documentation of the system's initial state.

Operation

Phase during which the embedded AI system operates under real-world conditions, producing local inferences and decisions, under supervision and in accordance with defined governance rules.

Evolution and Maintenance

All activities aimed at correcting, adapting or improving an embedded AI system during its operation, whilst ensuring service continuity, the traceability of changes and compliance with the original requirements.

End of Life

The phase during which an embedded AI system is taken out of service, including the controlled deactivation of AI functions, the archiving of elements necessary for traceability, and the management of final model versions.

11.2. Validation and lifecycle management

This section presents the concepts used to demonstrate that the embedded AI behaves in accordance with the defined requirements. It covers the activities necessary to ensure that the system operates correctly, in a reproducible manner and under conditions representative of its actual use.

These concepts form an essential foundation for ensuring that embedded AI remains under control throughout its operation and for providing objective evidence to justify the system's performance and behaviour in constrained or critical contexts.

Embedded AI validation

A process aimed at demonstrating that the embedded AI system meets the defined functional and operational requirements, under conditions representative of its actual use on the embedded platform.

Verification

All activities designed to ensure that the embedded AI system is correctly implemented in accordance with the defined specifications, particularly in terms of behaviour, performance and

time constraints.

Embedded AI Testing

Tests carried out on the embedded target or in a representative environment to assess the behaviour of embedded AI under conditions close to actual operation.

Continuous Validation

The process of verifying, during operation or following an update or recalibration, that the behaviour of the embedded AI remains compliant with the initial requirements, despite changes in usage conditions, input data or the operational context.

11.3. Governance of Embedded AI

This section covers the concepts used to guide technical and organisational decisions relating to embedded AI. It clarifies the responsibilities, rules and processes required to manage models, configurations and their evolution over time.

These concepts are essential for ensuring sustainable control of embedded AI, facilitating decision-making and guaranteeing the responsible, documented and consistent operation of embedded intelligent systems.

Governance of Embedded AI

The set of rules, processes and responsibilities enabling the management of the design, deployment, operation and evolution of embedded AI systems, ensuring the consistency of technical and organisational decisions throughout the lifecycle.

Accountability

The principle that those involved in the design and use of AI must be identifiable and accountable for the decisions made by the systems. This is particularly important for autonomous embedded systems, especially in the event of an incident or dispute. Accountability is a property that ensures that an entity's actions can be attributed solely to that entity.

Operational Accountability

Definition of responsibilities relating to the day-to-day operation, monitoring and corrective actions on an embedded AI system, particularly in the event of a performance degradation or incident.

Lifecycle Traceability

The ability to trace the history of decisions, model versions, parameters, processing chains and configurations throughout the lifecycle of an embedded AI system, in order to enable the auditing and justification of the choices made.

Version Management

A process for managing changes to models, parameters, processing chains and configurations deployed on embedded systems, ensuring clear identification of the versions in operation.

EdgeOps

The set of operational practices and processes dedicated to AI models deployed on embedded

systems, covering deployment, monitoring, maintenance, updates and version management, subject to resource and connectivity constraints.

11.4. Controlled operation and monitoring

This section addresses concepts related to observing and monitoring the behaviour of embedded AI systems in real-world operation. It helps to understand how the performance, stability and behaviour of AI can be monitored and evaluated over time.

These concepts are essential for maintaining a level of operational confidence in embedded AI, by identifying degradation scenarios and ensuring predictable behaviour in variable operating environments.

Embedded Monitoring

Mechanisms for monitoring the operation of embedded AI at runtime, to detect abnormal behaviour, performance deviations or unforeseen situations.

In-Operation Monitoring

Continuous or periodic observation of the behaviour and performance of embedded AI during actual operation, whilst adhering to the system's computational and energy constraints.

On-Device AI Telemetry

Collection and reporting, on the on-device target, of metrics relating to AI operation (inference time, load, input quality, confidence scores, degradation events), to enable analysis and supervision compatible with on-device constraints.

Model Drift

Gradual degradation of a model's performance in operation, linked to changes in input data, usage context or environment, which may require recalibration, updating or revalidation of the on-device system.

Drift Management

The set of mechanisms designed to detect, characterise and address performance drifts in an embedded AI system, including thresholds, alerts, degraded modes, recalibration and versioned updates.

Periodic Model Recalibration

Planned or triggered adjustment of an embedded AI system (parameters, thresholds, associated calibrations) to maintain compliant performance during operation, whilst ensuring the necessary traceability and revalidation.

Post-deployment monitoring

The activity of monitoring the behaviour of an embedded AI system over time after it has been commissioned, to confirm that expected performance is being maintained and to identify any need for adjustment or updating.

Graceful Degradation

The ability of an embedded AI system to adopt safe, predictable and documented behaviour in

the event of performance loss, degraded data or non-nominal operating conditions.

11.5. Updates and Operational Continuity

This section describes the mechanisms for evolving an embedded AI system after deployment, whilst maintaining service continuity. It formalises the principles required to introduce controlled and reversible changes in embedded environments.

These concepts play a key role in the sustainability of embedded AI systems, enabling their gradual adaptation without compromising stability or operational control.

Over-the-Air update (OTA)

A mechanism for remotely deploying a new version of a model, setting or configuration to an embedded target, ensuring version identification and service continuity.

Rollback

The ability to restore a previous version of a model or configuration on an embedded target, particularly in the event of a performance regression, instability or an incident detected during operation.

12. Regulations and standards for embedded AI

12.1. European regulations

AI Act

Regulatory framework proposed by the European Union to govern the use of AI according to its level of risk, with specific obligations for critical systems.

- [Regulation \(EU\) 2024/1689 - EN - EUR-Lex](#)
- [Legislation | EU Artificial Intelligence Act](#)

General Data Protection Regulation (GDPR)

European regulation harmonising the rules on the processing of personal data in the EU, which came into force on 25 May 2018. It strengthens individuals' rights and makes data controllers accountable.

Data Act

Having entered into force in January 2024, the EU Data Act (Regulation (EU) 2023/2854) has been fully applicable since September 2025 following the expiry of its transitional period. This legislation establishes a harmonised framework setting out new rules on the access, use and sharing of industrial data generated by connected products and services in the EU, across all economic sectors.

Chips Act

The European Chips Act (Regulation (EU) 2023/1781) aims to strengthen the EU's semiconductor capacity, support research and development and develop advanced production capabilities. Its objective is to achieve 20 per cent of global chip production by 2030. It also provides for a coordination mechanism to secure the supply chain in the event of a crisis. It promotes the emergence of a European ecosystem capable of designing and producing components optimised for embedded AI and cyber-physical systems.

Data Governance Act (DGA)

The Data Governance Act (Regulation (EU) 2022/868) sets out the governance and mechanisms for data sharing, in particular the reuse of certain public sector data, the framework for data intermediaries and “altruistic” data sharing.

Product Liability Directive

The new European Product Liability Directive (2024/2853) was adopted on 23 October 2024 and must be transposed into the national laws of Member States by 9 December 2026. It modernises liability rules to include digital products, AI and circular business models.

Cybersecurity Act (2019)

A European regulation adopted in 2019, which establishes the European cybersecurity certification framework for ICT products. It gave rise to future certification schemes such as EUCC, EUCS, EU5G, etc. It does not create direct obligations for manufacturers, but it structures

the trust framework and certification schemes. Its aim was to harmonise cybersecurity certification across Europe.

Cyber Resilience Act (CRA)

A European regulation that came into force on 10 December 2024, aimed at improving the cybersecurity of products containing digital elements (PDEs) and securing the Internet of Things. It imposes legal cybersecurity obligations on all products with digital elements (PDEs), including connected devices, embedded software, industrial systems, etc. It aims to ensure that all digital products sold in Europe meet a minimum level of cybersecurity throughout their lifecycle.

12.2. Cross-cutting directives with significant impact

General Product Safety Regulation (GPSR)

Regulation (EU) 2023/988 on general product safety, applicable from December 2024. It requires manufacturers to ensure the safety of consumer products, including those incorporating software or AI, and strengthens market surveillance obligations.

Digital Product Passport (DPP)

A measure introduced by the EU's Circular Economy Package, aimed at providing a digital passport containing essential information about a product (composition, reparability, durability, traceability). It is being gradually applied to several product categories, including electronic equipment.

NIS2

Directive (EU) 2022/2555 strengthening cybersecurity requirements for essential and important sectors. It imposes increased obligations regarding risk management, system security and incident reporting, directly impacting embedded systems and critical infrastructure.

Radio Equipment Directive (RED)

Directive 2014/53/EU on radio equipment now incorporates strengthened requirements regarding cybersecurity, data protection and fraud prevention for connected devices. The associated delegated acts impose specific obligations on manufacturers of radio devices incorporating software or AI functions.

European Cybersecurity Certification Scheme (EUCC)

European cybersecurity certification scheme for ICT products, developed under the Cybersecurity Act. It provides a harmonised framework for certifying the security of hardware and software components, relevant to embedded systems and products incorporating AI.

12.3. European standards supporting AI regulation

Harmonised European standards designed to support the implementation of the AI Act are currently being developed within Joint Technical Committee 21 (AI) of the CEN/CENELEC organisation. Compliance with all these standards by suppliers and deployers of AI systems will constitute a presumption of conformity with the provisions of the regulation, particularly for high-risk qualified systems.

As these standards do not yet exist, this section provides a list of all the standards expected to be published in 2026 and 2027. They will apply in full to embedded systems, even though their content is not specific to them.

Additional standards for general-purpose AI systems (including generative AI, LLMs, etc.) are expected to be developed at a later stage.

Quality Management Systems (prEN 18286)

Quality management for AI systems

Risk Management (prEN 18228)

Risk management for AI systems.

AI trustworthiness framework – Part 1

Logging, transparency and human oversight (prEN 18229-1): managing three aspects of trust in AI systems: traceability, user information and oversight by human operators.

AI trustworthiness framework – Part 2

Accuracy and robustness (prEN 18229-2): management of two aspects of trust in AI systems: accuracy and robustness.

Quality and governance of datasets (prEN 18284)

Standard on the quality and governance of datasets used in AI systems.

Managing Bias (prEN 18283)

Management (and reduction) of bias in data, in order to meet fairness requirements.

Cybersecurity specifications for AI systems (prEN 18282)

Specifications on the cybersecurity of AI systems, analysis of threats and ways to respond to them.

Conformity Assessment (prEN 18285)

Assessment of conformity with all standards, carried out by authorised bodies.

The following two standards, also currently in preparation, do not correspond to an explicit request from the European Commission.

- Evaluation methods for accurate computer vision systems (prEN 18281)
- Natural Language Processing (prEN ISO/IEC 23282)

12.4. International standards for governance and risk management

The harmonised European standards currently being developed draw inspiration from these international standards without replicating them in full, as the latter primarily concern organisations that produce AI systems, whereas European regulations focus on products. Compliance with ISO/IEC standards will therefore not constitute a presumption of conformity with

European regulations.

ISO/IEC 42001

The first international standard dedicated to artificial intelligence (AI) management systems, published in December 2023, which provides a framework for the development, use and governance of AI systems to ensure responsible, ethical and secure use.

ISO/IEC 5338

Published on 20 December 2023, it provides a standardised framework for the development and management of AI systems, covering all aspects of the lifecycle of software incorporating AI.

ISO/IEC 23894

Provides a comprehensive framework for identifying, assessing and managing risks associated with artificial intelligence, drawing on the principles of ISO 31000 to ensure effective integration into organisations' activities.

12.5. Sector-specific standards

The rise of embedded AI in critical systems necessitates reliance on established sector-specific standards. These sector-specific standards form the essential foundation of trust required for the integration of embedded AI into critical systems. Whether in the automotive, healthcare, aerospace, industrial, IoT, defence, energy or rail sectors, they provide a framework for functional safety, cybersecurity and regulatory compliance. Together, they ensure that embedded AI can be deployed reliably, interoperably and securely in demanding environments.

12.5.1. Automotive

This summary highlights that in the automotive sector, standards cover functional safety, algorithm reliability, cybersecurity and software update management: four essential pillars for embedded AI.

ISO 26262 (Functional Safety)

International standard for functional safety of electrical and electronic systems in vehicles. It defines development, validation and verification processes to reduce risks associated with hardware or software failures. Essential for any critical embedded system, including those incorporating AI.

ISO/PAS 21448 (SOTIF – Safety Of The Intended Functionality)

A supplement to ISO 26262, covering risks associated not with technical failures, but with the limitations of design and algorithms. Crucial for in-vehicle AI, as it addresses the unexpected or unanticipated behaviour of intelligent systems.

ISO/SAE 21434 (Cybersecurity for Road Vehicles)

A joint ISO/SAE standard defining cybersecurity requirements for connected vehicles. It covers the entire lifecycle, from design to maintenance, to protect against attacks and intrusions. Essential for embedded architectures incorporating AI and connectivity.

UNECE R155/R156 (Cybersecurity and Software Updates)

Regulations of the United Nations Economic Commission for Europe (UNECE) requiring car

manufacturers to implement cybersecurity management (R155) and secure software update processes (R156). They mandate protection and continuous maintenance practices directly linked to intelligent in-vehicle systems.

12.5.2. Healthcare

This summary shows that in the healthcare sector, standards cover device safety, software reliability, cybersecurity and regulatory compliance, all of which are essential for the integration of embedded AI.

MDR (Medical Device Regulation)

A European regulation governing the placing on the market of medical devices. It imposes strict requirements regarding safety, performance and traceability, which are essential for embedded systems integrating AI into medical devices.

IVDR (In Vitro Diagnostic Regulation)

Complementary to the MDR, it regulates in vitro diagnostic devices. It ensures the reliability and safety of embedded systems used in biological testing, particularly where AI is involved in the analysis and interpretation of results.

IEC 62304

An international standard for the lifecycle of medical device software. It defines the processes for development, maintenance and risk management, which are crucial for embedded software incorporating AI algorithms.

IEC 60601

Safety and performance standard for electromedical equipment. It covers electrical, mechanical and software aspects, ensuring that embedded devices, including those enhanced by AI, do not pose a risk to patients.

IEC 81001-5-1

A recent standard dedicated to the cybersecurity of healthcare software. It sets out requirements to protect embedded systems against digital threats, a major challenge for connected and distributed medical AI.

12.5.3. Aerospace

In summary, in aeronautics, standards cover three essential pillars for embedded AI: software safety, hardware reliability and cybersecurity.

DO-178C (Software Considerations in Airborne Systems and Equipment Certification)

The benchmark standard for the certification of embedded software in aeronautics. It defines the levels of assurance required according to the criticality of functions, with strict development, verification and validation processes. Essential for all critical embedded software, including those incorporating AI.

DO-254 (Design Assurance Guidance for Airborne Electronic Hardware)

This standard complements DO-178C: it applies to airborne electronic hardware (circuits, FPGAs,

ASICs). It imposes rigorous design and validation methods to ensure the reliability of critical components. It is relevant for embedded AI when it relies on specialised hardware.

DO-326A / DO-356A (Airworthiness Security Process & Information Security Guidance)

Standards dedicated to aviation cybersecurity. They define processes for managing threats and vulnerabilities, as well as requirements for protecting onboard systems against attacks. They are crucial for connected onboard AI, which must remain secure in the face of cyber risks.

12.5.4. Industry

In industry, standards focus on two fundamental areas for embedded AI: functional safety and the cybersecurity of industrial systems.

IEC 61508 (Functional Safety of Electrical/Electronic/Programmable Systems)

An international standard for functional safety covering electrical, electronic and programmable systems. It defines the requirements for design, validation and risk management to ensure that critical systems operate safely. It is relevant to embedded AI when integrated into automated industrial processes.

IEC 62443 (Industrial Communication Networks – IT Security for Industrial Automation Systems)

A standard dedicated to the cybersecurity of industrial systems and OT (Operational Technology) environments. It covers the entire lifecycle, from design to operation, and sets out requirements to protect embedded architectures against cyber threats. It is essential for embedded AI in connected factories and Industry 4.0.

12.5.5. Internet of Things (IoT)

In the IoT, this standard serves as the central framework for ensuring the security and trustworthiness of smart embedded systems.

ETSI EN 303 645 (Cybersecurity for Consumer IoT)

A European standard that defines cybersecurity requirements for consumer IoT devices. It sets out minimum protection rules (strong passwords, secure updates, personal data management) to reduce the vulnerabilities of IoT devices. It is relevant to embedded AI, as it provides a framework for the security of sensors, gateways and smart devices used in domestic and urban environments.

12.5.6. Defence

STANAG (Standardisation Agreement – NATO)

STANAGs are NATO standardisation agreements that define technical and operational standards to ensure interoperability between allied forces. In the field of embedded AI, they relate in particular to system security, communication reliability and the compatibility of critical equipment. They ensure that embedded solutions comply with common safety and cybersecurity requirements in military environments.

12.5.7. Energy / Smart Grid

In the energy sector, these standards ensure the interoperability, reliability and security of communications in smart grids, enabling embedded AI to play a key role in optimising and enhancing the resilience of infrastructure.

IEC 61850 (Communication Networks and Systems for Power Utility Automation)

An international standard defining the architecture and communication protocols for smart power systems. It enables interoperability between equipment from different manufacturers and ensures the reliability of real-time data exchange within power networks. It is relevant to embedded AI as it provides a framework for local and distributed data management in smart grids.

IEEE 2030.5 (Smart Energy Profile 2.0)

An American standard specifying communication protocols for the interoperability of distributed energy networks, particularly in the context of renewable energy and demand response. It promotes the integration of embedded AI to optimise the dynamic management of energy flows and the coordination of microgrids.

12.5.8. Rail/Railway

In the railway sector, these standards ensure that embedded AI meets the same software and electronic safety requirements as traditional systems, guaranteeing reliability and compliance in a critical environment.

EN 50128 (Software for Railway Control and Protection Systems)

A European standard defining the requirements for the development and validation of software used in railway control and signalling systems. It is based on IEC 61508 but adapts it to the specificities of the railway sector. It is relevant to embedded AI where AI is involved in traffic management, signalling or safety systems.

EN 50129 (Safety-Related Electronic Systems for Signalling)

A complementary standard governing the safety of electronic systems related to railway signalling. It sets out the design, assessment and certification requirements to ensure the safe operation of embedded architectures. Embedded AI is covered by this standard where it is involved in decision-making within critical systems.

12.5.9. Sector-specific supplements for CE marking

Automotive

Already covered by ISO 26262, SOTIF, ISO/SAE 21434, UNECE R155/156. To this must be added the General Regulation on Vehicle Safety (EN 2019/2144), which mandates certain automated functions and electronic safety features.

Health

ISO 14971 (Risk Management for Medical Devices) must be added; this harmonised standard is essential for the CE marking of medical devices.

Aeronautics

No direct CE marking (the aeronautics sector falls under EASA/FAA certification), but EN 9100 (aeronautical quality) may be cited as a harmonised standard.

Industry

IEC 61508 and IEC 62443 are key; to these must be added the **Machinery Directive (2006/42/EC / future Machinery Regulation)** and the **ATEX Directive (2014/34/EU)** in the case of systems installed in explosive atmospheres.

Internet of Things (IoT)

ETSI EN 303 645 is central. To this must be added **the Radio Equipment Directive (2014/53/EU)**, already mentioned in section 12.2; this is mandatory for all connected radio devices (Wi-Fi, Bluetooth, LTE, and others). It determines the CE marking of IoT equipment by imposing requirements for electrical safety, EMC and cybersecurity. It is relevant and essential for embedded AI whenever radio connectivity is involved.

Defence

STANAG remains specific to NATO. No direct CE marking (military products excluded); however, for dual-use equipment (civilian + military), the cross-cutting CE directives (EMC, Low Voltage, RoHS, Machinery, RED) apply for CE marking.

Energy / Smart Grid

IEC 61850 and IEEE 2030.5 are sector-specific. The Low Voltage Directive and the Electromagnetic Compatibility Directive must also be taken into account, as they are always required for the CE marking of electrical equipment.

Rail/Railway

EN 50128 and EN 50129 are sector-specific. The **Railway Interoperability Directive (EU 2016/797)** must also be included, as it governs the CE conformity of railway subsystems.

13. Applications and Use Cases

This chapter brings together the terminology used to describe, structure and classify the applications and use cases of embedded AI. It focuses on artificial intelligence functions executed directly on embedded systems incorporating sensors, local computing capabilities and dedicated processing chains, in order to generate decisions, estimates or commands in interaction with the physical world. The concepts presented aim to provide a common framework for linking the operational objectives of embedded AI to on-device execution constraints, architectural choices and operational requirements in real-world environments.

13.1. General concepts

The concepts outlined here define what an embedded AI application is and how it fits into a real-world system. They clarify the relationship between operational objectives, AI functions deployed on the target and operational scenarios, considering the specific constraints of embedded platforms. This conceptual framework serves to coherently describe the uses of AI integrated into physical systems.

Application

A coherent set of functions provided by an embedded system, some of which are based on AI models running on the target platform. An embedded AI application combines sensors, processing and decision-making logic to produce a behaviour, capability or operational service integrated into the system.

Use case

A structured description of an operational scenario in which one or more embedded AI functions contribute to achieving an identified operational objective. A use case specifies the context, sensor inputs, expected outputs and associated execution constraints.

Embedded AI function

A software function deployed on an embedded system, implementing a processing chain and one or more AI models to produce an actionable output (information, alert, estimate, decision or command) in interaction with the environment.

Operational scenario

A description of a real-world usage context characterised by an environment, variable operating conditions, time constraints, and nominal or degraded situations to which the embedded AI must respond reliably.

Benefit of local inference

Advantage gained by running AI models directly on the embedded target, enabling compliance with requirements for responsiveness, autonomy, service continuity, resilience or control of data flows.

13.2. Major families of sector-specific applications

Organising by sector highlights the diversity of contexts in which embedded AI is deployed. Each sectoral category illustrates concrete use cases based on the local execution of AI functions, enabling improved responsiveness, autonomy and robustness of systems in industrial, critical or distributed environments.

Industry, manufacturing and processes

Applications in which embedded AI analyses data from industrial sensors to carry out inspections, monitor processes and detect anomalies directly on the equipment. Embedded processing enables immediate responsiveness, robustness in industrial environments and a reduction in data flows to central systems.

Autonomous and collaborative robotics

Applications where embedded AI supports perception, navigation, manipulation and interaction between robots and operators. Embedded processing ensures latency compatible with movement and physical interaction, as well as autonomous operation in dynamic environments.

Mobility, transport and smart vehicles

Applications incorporating embedded AI to interpret the environment, assist with driving, manage trajectories and support autonomous functions. Embedded systems enable fast and reliable local decision-making, independent of network connectivity.

Aeronautics and aerospace

On-board applications in aircraft, drones or space systems using AI for perception, surveillance and decision support. On-board AI provides autonomy, resilience and service continuity in isolated or highly constrained environments.

Defence and critical systems

Applications in which embedded AI contributes to surveillance, situational analysis and decision support. Embedded processing enables robust, offline and controlled operation, which is essential in critical contexts.

Healthcare, medical devices and pharmaceuticals

Applications integrating embedded AI into medical devices, diagnostic equipment or monitoring systems. Local processing promotes responsiveness, operational continuity and better control over sensitive data.

Energy, networks and critical infrastructure

Applications utilising embedded AI to monitor, diagnose and optimise energy equipment or distributed infrastructure. Embedded processing enables efficient local management and large-scale operation of autonomous devices.

Telecommunications and network equipment

Applications in which embedded AI is used to optimise the use of network resources and equipment, detect anomalies and improve service quality on edge computing. Embedded systems enable rapid adjustments as close as possible to the traffic and equipment.

Smart construction, building and infrastructure

Applications using embedded AI for energy management, equipment monitoring and the security of buildings or structures. Embedded processing facilitates local decisions tailored to actual usage and varying conditions.

Agriculture and the environment

Applications utilising sensors and on-board vision to monitor the environment, analyse crops or detect natural events. On-board AI enables autonomous, energy-efficient devices capable of operating for long periods without continuous connectivity.

Connected devices, wearables and consumer electronics

Applications incorporating compact AI models for interaction, personalisation and activity recognition. On-device processing ensures a responsive experience, low energy consumption and independence from remote infrastructure.

Culture, media and interactive content

Applications in which embedded AI supports audiovisual analysis, real-time interaction, augmented reality or immersive experiences. Embedded processing enables fluid and immediate interactions on autonomous devices.

13.3. Typical application functions

The functions described correspond to the operational capabilities provided by embedded AI, such as environmental perception, situation analysis, state estimation or user interaction. They constitute reusable functional building blocks, implemented directly on the embedded platform to produce decisions or commands adapted to the system's constraints.

On-board perception

The ability of an on-board system to interpret its physical environment using data from sensors (image, radar, LiDAR, audio, inertial, biometric, etc.), to produce a representation that can be used for decision-making or to control a physical system.

Detection

The function of identifying the presence or occurrence of objects, events or situations of interest in input data, for example the detection of a fault, an obstacle, an incident or a change of state, with an output that is generally binary or localised.

Classification

A function aimed at associating a sensor observation with a predefined category, such as object type, equipment status, environmental condition or activity, to guide a decision or local action.

Segmentation

A function enabling data, particularly an image or a signal, to be divided into homogeneous regions or segments corresponding to areas of interest, in order to precisely locate objects, defects or phenomena relevant to the embedded application.

Estimation

A function that produces a continuous value representing a state or a physical quantity, such as a position, velocity, orientation, level or intensity, based on sensor measurements, to feed into a control or supervision logic.

Prediction

A function consisting of anticipating the future evolution of a state, behaviour or observed phenomenon, for example equipment wear, a probable trajectory or a performance drift, to enable proactive action by the embedded system.

Anomaly detection

A function designed to identify significant deviations from a learned or defined nominal operating state, to detect faults, incidents or unusual situations early on in real-world environments.

Activity recognition

A function enabling the identification of an activity, gesture or behaviour based on time series or multimodal signals, such as data from inertial, biometric or acoustic sensors, particularly in wearables and interactive systems.

Embedded intelligent interaction

A function enabling direct and responsive interaction between a user and an embedded system, via voice, gestures, context or sensory interfaces, with local processing ensuring low latency and service continuity.

Perception–decision–action loop

A functional organisation in which information derived from perception feeds into a local decision, which then triggers an action or command on the system or its environment, forming a closed loop executed under real-time constraints.

13.4. Structural cross-cutting constraints

The constraints presented characterise the conditions under which embedded AI applications must be designed and operated. They relate to the need for local processing, control of response times, energy autonomy, robustness in real-world conditions and the management of device fleets. These constraints directly influence the technical choices and architecture of embedded solutions.

On-target processing

The principle whereby AI functions are executed directly on the embedded platform, without systematic reliance on a remote infrastructure, to guarantee responsiveness, operational autonomy and service continuity in constrained environments.

Application real-time

A requirement stipulating that the results produced by AI must be available within a timeframe compatible with the application, for example to trigger an action, assist an operator or control a physical system, with limited and controlled response times.

Energy autonomy

The ability of an AI-integrated embedded system to operate sustainably within a limited energy budget, whether powered by a battery, energy recovery, or subject to strict consumption constraints.

Operational robustness

The ability of the embedded AI system to maintain useful and predictable behaviour despite the variability of real-world conditions, sensor noise, environmental disturbances or non-nominal situations.

Limited or intermittent connectivity

A context in which the embedded system cannot rely on continuous or high-speed network access, requiring autonomous decision-making and processing capabilities directly on the target.

Large-scale deployment

Deployment of the same embedded AI function across a large number of devices, requiring consistent behaviour, version control and industrial-scale operation of the embedded fleet.

13.5. Representative use cases

The selected use cases illustrate real-world situations in which embedded AI delivers measurable operational value. They demonstrate how analysis, detection or prediction functions are integrated into embedded systems to meet needs such as inspection, maintenance, monitoring or decision support, without permanent reliance on remote infrastructure.

On-board visual inspection

A use case in which an embedded system analyses images or video streams from cameras to detect defects, assess compliance or sort objects, with local execution ensuring responsiveness and robustness in industrial or operational environments.

On-board predictive maintenance

A use case where signals from sensors (vibration, acoustic, current, temperature, pressure) are analysed by on-board AI models to anticipate failures, identify operational deviations and plan appropriate maintenance actions.

Monitoring of critical events

Use cases involving the timely detection of rare or abnormal events, such as an incident, dangerous behaviour or a threshold being exceeded, to trigger an alert or immediate action by the embedded system.

Driving and mobility assistance

Use case in which multi-sensor perception and local decision-making support driving assistance, navigation or partial autonomy functions, with high real-time and reliability requirements.

Local energy optimisation

Use case where an on-board system dynamically adjusts energy management strategies (charging, equipment control, prioritisation) based on local measurements and estimates, in

order to improve energy efficiency without continuous reliance on a remote infrastructure.

Decision support in isolated environments

Use cases in which embedded AI provides recommendations, alerts or diagnostics directly in the field, in contexts with limited or no connectivity, to support human or automated action.

Activity recognition on wearables

Use cases where a wearable device locally analyses data from inertial, biometric or physiological sensors to identify activities, gestures or states, whilst meeting strict energy efficiency requirements.

Local infrastructure monitoring

Use cases in which embedded devices deployed on infrastructure carry out continuous or periodic diagnostics of the condition of equipment or structures, producing actionable indicators even under conditions of intermittent connectivity.

13.6. Emerging use cases

Emerging use cases reflect the recent evolution of embedded AI towards more adaptive, cooperative and responsible systems. They highlight approaches aimed at local model adaptation, cooperation between devices, resource efficiency and consideration of environmental and social issues, whilst maintaining predominantly embedded execution.

Local adaptation and on-device learning

The limited ability of an on-device AI system to adjust certain parameters or thresholds during operation to maintain its performance in the face of changing environments, subject to resource and safety constraints.

Distributed embedded AI

An organisation in which several embedded systems cooperate to produce a collective perception or decision, whilst retaining predominantly local execution.

Frugal and responsible embedded AI

Applications in which embedded AI is used as a lever for digital frugality, minimising the energy footprint, the volumes of data processed and dependence on remote infrastructure, whilst maintaining operational utility.

Embedded AI for environmental sustainability

Applications utilising embedded AI to observe, analyse and act locally on environmental phenomena, using autonomous, distributed and resource-efficient devices.

Embedded AI for trust and social acceptance

Applications in which embedded AI helps to build trust among users and stakeholders through controlled local decision-making, increased operational transparency and integration that respects human and social contexts.