2

Taxonomy and Terminology

The objective of this section is to provide an overview of taxonomy, terminology and definitions of AI and edge AI terms and concepts, which are essential for clear communication, systematic analysis, development of a strategic research agenda, innovation, regulation, business strategy, interdisciplinary collaboration, effective deployment and standardisation. It lays the foundation for the coherent and unified progress of the field.

Having standard definitions and terminology for AI and edge AI terms and concepts prevents miscommunications and misinterpretations arising from varying interpretations of terms, establishing a common baseline for discussions and ensuring that the stakeholders involved have a clear understanding of the key concepts.

Taxonomy allows the classification of different AI and edge AI methods, algorithms, and systems, making it easier to address and analyse them systematically by helping in effectively evaluating, comparing, and benchmarking different approaches, technologies, solutions, and applications.

A clear taxonomy can support identifying gaps in current research and areas that need further exploration and provide a solid foundation upon which new ideas, models, and technologies can be built.

Using a common taxonomy, terminology and definitions of AI and edge AI terms and concepts helps regulators and policymakers create appropriate frameworks, guidelines, and standards to ensure the ethical and safe development and deployment of AI and edge AI technologies, ensuring compliance and responsible AI deployment.

Edge AI applied to different industrial domains requires interdisciplinary collaboration, as the technology's development intersects with various fields, such as the Internet of Things (IoT), sensing/actuating, edge computing, cybersecurity, and intelligent connectivity.

Understanding the definitions and taxonomy can help identify the most suitable AI or edge AI techniques for specific real-world applications, optimise performance, and ensure practical feasibility.

2.1 Definitions

In this section, we propose definitions for the main concepts related to AI and edge AI considering that the challenge with a definition is that it specifies the meaning or significance of a word or phrase and can concern either its usage or the content of a concept expressed by a word.

Definition challenges often arise due to the inherent complexity and variability of the scientific and technological domain, as well as the different terminologies used in other domains.

Moreover, as technology and science evolve, so do their terminology and definitions, resulting in shifts in meaning that can render previously clear definitions outdated or incomplete. This dynamic nature of terminology requires continuous adaptation and consensus, which can be challenging to achieve across the scientific and industrial domains, especially in rapidly advancing or inherently subjective fields, such as AI technology.

2.1.1 Artificial Intelligence

Many AI researchers acknowledge that defining intelligence in a universally satisfactory manner is challenging. However, for the field to advance coherently, it is essential to reason from a clear and generally accepted statement of its subject matter. Despite the lack of a standardised definition of intelligence in AI today, establishing a common framework is crucial for guiding research, fostering communication, and ensuring consistent progress in the discipline.

In this context, several definitions of artificial intelligence and machine intelligence exist as listed in Table 2.1.

AI system: An AI system is a machine-based system that, for explicit or implicit objectives, infers from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments. Different AI systems vary in their levels of autonomy and adaptiveness after deployment [81].

ISO/IEC DIS 22989 standard defines an AI system as an engineered system featuring AI. The AI systems can be designed to generate outputs such as predictions, recommendations and classifications for a given set of human defined objectives. The AI systems can be designed to operate with varying levels of automation [21].

AIA states that an AI system means a machine-based system that is designed to operate with varying levels of autonomy and that may exhibit adaptiveness after deployment, and as mentioned above, for explicit or

Table 2.1 AI definitions

Definition
• Artificial intelligence is that activity devoted to making machines intelligent, and intelligence is that quality that enables an entity to function appropriately and with foresight in its environment [67].
• The study of agents that receive percepts from the environment and perform actions [68].
• The science of making machines do things that would require intelligence if done by humans [61].
• The use of computer programs and programming techniques to cast light on the principles of intelligence in general and human thought in particular [61].
• Artificial Intelligence is a science and a set of computational technologies that are by inspired by-but typically operate quite differently from-the ways people use their nervous systems and bodies to sense, learn, reason, and take action [64].
• Intelligence is the capacity of an information-processing system to adapt to its environment while operating with insufficient knowledge and resources [74].
• Artificial Intelligence is a fully controlled agent with a capacity of an information-processing system to adapt to its environment while operating with insufficient knowledge and resources [73].
• Artificial intelligence may be considered a computational construct, as it is inferred from the outcomes of simulated aspects of human thought and decision-making, which are facilitated by data processing, machine learning techniques, and algorithmic principles [62].
• Intelligence is the computational part of the ability to achieve goals. A goal achieving system is one that is more usefully understood in terms of outcomes than in terms of mechanisms [65].
• Artificial intelligence is a machine's ability to perform logical analysis, acquire knowledge, and adapt to an industrial environment that varies over time or in context. These abilities include the collective attributes of a machine (i.e., computer, robot, or intelligent IoT device) to perform functions such as perception, understanding, reason, prediction, learning, decision making and action [2].
• Artificial intelligence is the discipline of research and development of mechanisms and applications of AI systems. Research and development can take place across any number of fields such as computer science, data science, humanities, mathematics and natural sciences [21].
• Artificial intelligence refers to systems that display intelligent behaviour by analysing their environment and taking actions – with some degree of autonomy – to achieve specific goals. AI-based systems can be purely software-based, acting in the virtual world (e.g., voice assistants, image analysis software, search engines, speech and face recognition systems), or AI can be embedded in hardware devices (e.g., advanced robots, autonomous vehicles, drones or Internet of Things (IoT) applications) [10, 16, 17, 18].
• Engineered system set of methods or automated entities that together build, optimize and apply a model so that the system can, for a given set of predefined tasks, compute predictions, recommendations, or decisions [21].
• Discipline study of theories, mechanisms, developments and applications related to artificial intelligence engineered system [21].

Table 2.1 *Continued.*

Definition
• AI is a fast-evolving family of technologies that contributes to a wide array of economic, environmental and societal benefits across the entire spectrum of industries and social activities. By improving prediction, optimising operations and resource allocation, and personalising digital solutions available for individuals and organisations, the use of AI can provide key competitive advantages to undertakings and support socially and environmentally beneficial outcomes, for example in healthcare, agriculture, food safety, education and training, media, sports, culture, infrastructure management, energy, transport and logistics, public services, security, justice, resource and energy efficiency, environmental monitoring, the conservation and restoration of biodiversity and ecosystems and climate change mitigation and adaptation [116].

implicit objectives, infers from the input it receives, how to generate outputs such as predictions, content, recommendations, or decisions that can influence physical or virtual environments [116].

AI training: Is defined as the process to determine or to improve the parameters of a machine learning model, based on a machine learning algorithm, by using training data [21].

AI inference: Reasoning by which conclusions are derived from known premises. A premise is either a fact, a rule, a model, a feature or raw data [21].

The edge AI definition builds on the converge of AI, IoT and edge computing technologies and is defined later in this section. It is useful to note that in continual learning, training and inference stages are no longer purely subsequent, but more heavily intertwined.

2.1.2 Open-Source

Open source is a paradigm and practice that promotes the free access, use, and modification of software, hardware, or other resources. It emphasises collaboration, transparency, and community-driven development. Open source typically involves distributing the source code or design files, allowing users to study, modify, and improve the product. This approach fosters innovation and inclusivity, enabling diverse contributions and widespread adoption.

Open-source hardware centres on making physical design files and documentation available for replication and modification. Open-source software focuses on freely accessible source code and the ability to modify and redistribute software. Open-source AI involves sharing code, models, and datasets for collaborative advancement, emphasising ethical considerations and transparency.

The most widely recognised definitions of open-source are the ones provided by the Open Source Initiative (OSI) [87], which maintains a set of criteria for what constitutes open-source software and now open-source, and the Open Source Hardware Association (OSHWA) [88] which maintains the definition and set of standards for open-source hardware.

The AI and edge AI developers are supporting open-source developments, and the European Commission is increasingly reinforcing the open accessibility of research outputs, including data, code, hardware and publications [83]. This push is part of the broader Open Science policy, aiming to make research more transparent, efficient, and impactful. Open access promotes better science and innovation in both public and private sectors. The Open Research Europe platform provides guidelines for authors on managing and sharing their research data [84]. These guidelines emphasise the importance of making data openly accessible and reusable. This European platform endorses the **FAIR** principles that emphasise making data **F**indable, **A**ccessible, **I**nteroperable and **R**eusable [85].

Open-source Hardware:

The OSHWA provides a widely recognized definition and set of standards for open-source hardware [88]. The definition is based on the Open Source Definition for Open Source Software [87].

Open Source Hardware (OSHW) refers to tangible artifacts, such as machines, devices, or other physical things, whose design has been released to the public so that anyone can make, modify, distribute, and use it. This definition aims to provide guidelines for developing and evaluating licenses for OSHW.

Hardware differs from software in that physical resources must always be committed to creating physical goods. Therefore, individuals or companies producing items ("products") under an OSHW license have an obligation to make it clear that such products are not manufactured, sold, warrantied, or otherwise sanctioned by the original designer and not to make use of any trademarks owned by the original designer.

The distribution terms of OSHW must comply with the following criteria [88]:

- **Documentation -** The hardware must be released with documentation including design files and must allow modification and distribution of the design files. Where documentation is not furnished with the physical product, there must be a well-publicized means of obtaining this

documentation for no more than a reasonable reproduction cost, preferably downloading via the Internet without charge. The documentation must include design files in the preferred format for making changes, such as the native file format of a CAD program. Deliberately obfuscated design files are not allowed. The license may require that the design files are provided in fully documented, open format(s).

- **Scope -** The hardware documentation must clearly specify what portion of the design, if not all, is being released under the license.
- **Necessary Software -** If the licensed design requires software, embedded or otherwise, to operate properly and fulfil its essential functions, then the license may require that one of the following conditions be met:
    - The interfaces are sufficiently documented that it could reasonably be considered straightforward to write open-source software that allows the device to operate properly and fulfil its essential functions. For example, this may include using detailed signal timing diagrams or pseudocode to clearly illustrate the interface in operation.
    - The necessary software is released under an OSI-approved open-source license.
- **Derived Works -** The license shall allow modifications and derived works to be distributed under the same terms as the license of the original work. The license shall also allow for the manufacture, sale, distribution, and use of products created from the design files, the design files themselves, and derivatives thereof.
- **Free redistribution -** The license shall not restrict any party from selling or giving away the project documentation. It shall not require a royalty or other fee for such sale or for the sale of derived works.
- **Attribution -** The license may require derived documents and copyright notices associated with devices to provide attribution to the licensors when distributing design files, manufactured products, and/or derivatives thereof. The license may require that this information be accessible to the end-user using the device typically but shall not specify a specific format of display. The license may require derived works to carry a different name or version number from the original design.
- **No Discrimination Against Persons or Groups -** The license must not discriminate against any person or group of persons.

- **No Discrimination Against Fields of Endeavor -** The license must not restrict anyone from using the work (including manufactured hardware) in a specific field of endeavour. For example, it must not restrict the hardware from being used in a business or from being used in nuclear research.
- **Distribution of License -** The rights granted by the license must apply to all to whom the work is redistributed without the need for execution of an additional license by those parties.
- **License Must Not Be Specific to a Product -** The rights granted by the license must not depend on the licensed work being part of a particular product. If a portion is extracted from a work and used or distributed within the terms of the license, all parties to whom that work is redistributed should have the same rights as those that are granted for the original work.
- **License Must Not Restrict Other Hardware or Software -** License Must Not Restrict Other Hardware or Software: The license must not place restrictions on other items that are aggregated with the licensed work but not derivative of it. For example, the license must not insist that all other hardware sold with the licensed item be open-source, nor that only open-source software be used external to the device.
- **License Must Be Technology-Neutral -** License Must Be Technology-Neutral: No license provision may be predicated on any individual technology, specific part, or component.

On the hardware side, RISC-V plays a crucial role in the open-source movement [13, 103]. RISC-V is an open standard Instruction Set Architecture (ISA) which allows anyone to design and sell RISC-V-based chips without licensing fees, fostering innovation and competition. Its flexibility and customizability make it suitable for various applications, from embedded systems to high-performance computing and AI. This inclusivity makes it easier for smaller companies to enter the market and encourages the use of open-source hardware, allowing high-performance computing to become available to a larger audience. Other open-source hardware platforms for AI are:

- **BeagleBone** [104]: An open-source development board that offers a powerful platform for AI and machine learning applications.
- **Raspberry Pi** [105]: The Raspberry Pi is a low-cost, credit-card-sized computer that can be used for a variety of AI projects. Its affordability and versatility make it an excellent choice for hobbyists and educators

looking to explore AI. Although the Raspberry Pi itself is not entirely open-source, its schematics are regularly released as documentation, and it does support and contribute to the open-source community.
- **ESP32** [106]: The core ESP32 hardware, developed by Espressif Systems, is proprietary. However, there are open-source hardware versions available, such as the ESP32-EVB by Olimex.
- **Antmicro's Open-Source Jetson Baseboard** [107]: There are open-source projects and community contributions that enhance the Jetson ecosystem. Antmicro has developed an open-source Jetson baseboard to allow customers to get full control of the solutions that we build based on it, along with unmatched flexibility, transparency and usability.
- **Eyes of Things (EoT) Project** [102]: The schematics of the EoT board were made public on the project's website. However, the main blocks, like the Myriad processor, were not disclosed.

Open-source Software:

According to the Open-Source Initiative, open-source software must comply with the following criteria:

- **Free Redistribution:** The software can be freely distributed to anyone without restrictions, including both source code and compiled binaries.
- **Source Code:** The source code must be included or made available, allowing users to modify and improve the software.
- **Modification:** Users must be able to modify the software to suit their needs. This includes the ability to create derivative works.
- **Integrity of the Author's Source Code:** Distributions of modified versions may be allowed only if they are clearly marked and not misrepresented as the original software.
- **Non-Discrimination Against Persons or Groups:** The license must not discriminate against any person or group of persons.
- **Non-Discrimination Against Fields of Endeavor:** The license must not restrict anyone from using the software in a specific field of endeavour, such as business or research.
- **Distribution of License:** The rights attached to the software must apply to all who receive it, making it unnecessary to sign further agreements.
- **License Must Not Be Specific to a Product:** The rights attached to the software must not depend on the software being part of a particular product.
- **The License Must Not Restrict Other Software:** The license should not impose restrictions on other software distributed along with the open-source software.

- **Technology-Neutral:** No provision of the license may be predicated on any individual technology or style of interface.

Open-source software is built on several key principles:

- **Collaboration:** Open source encourages collaborative development, allowing developers from different backgrounds to contribute to and improve the software.
- **Transparency:** With access to source code, users can understand how the software operates, which fosters trust and accountability.
- **Community-Driven:** Open-source projects often rely on communities of users and developers who advocate for improvements and support one another.
- **Innovation:** Open-source promotes innovation by allowing anyone to build upon existing software, leading to rapid development cycles and diverse applications.
- **Access and Inclusion:** By removing cost barriers and restrictions, open-source software is accessible to a wide range of users, including those in developing countries.

These principles help to create a dynamic ecosystem where software can evolve and adapt to meet the needs of its users while promoting inclusivity and collaboration.

Open-source AI:

The potential and the critical challenges of open-source AI are highlighted in several references [86, 89, 92, 93]. On the positive side, open-source AI can enhance transparency and facilitate the auditing of AI systems, which in turn can build citizen trust. It also stimulates economic activities and fosters domain-specific expertise by allowing both the public and private sectors to innovate more freely. However, legal challenges arise from the need to navigate complex intellectual property rights and licensing issues. Another drawback is that maintaining and updating AI projects can be demanding, requiring substantial resources and expertise. Data-related challenges include ensuring the availability of high-quality datasets and addressing privacy concerns. Risk management is another critical area, as open-source AI systems can be more vulnerable to security threats. Additionally, societal and ethical challenges must be considered, such as the potential for bias in AI systems and the broader implications of AI deployment on society.

Open-source hardware and software have revolutionized the field of AI, making advanced technologies more accessible and fostering innovation through community collaboration. Thus, we can claim that it is not only the

advent of deep learning and powerful GPUs what have caused the AI explosion, but also the widespread access to research results, code and datasets. An open-source AI ecosystem has helped lower the barriers for researchers, students and practitioners to acquire AI skills and develop prototypes of new exciting ideas. Some well-known examples of open-source software for AI include:

- **TensorFlow and PyTorch** [94, 95]: Developed by Google and Facebook's AI Research lab respectively, they are two of the most prominent open-source frameworks for AI. They offer a comprehensive ecosystem of tools, libraries, and community resources that support a large variety of AI applications.
- **Keras** [96]: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. It allows for easy and fast prototyping, making it a popular choice for beginners and experts alike.
- **Hugging Face** [97]: It is known for its open-source contributions. The company provides a suite of tools, including the popular Transformers library, which offers pre-trained models for tasks such as text classification, translation, and summarization. Hugging Face's commitment to open-source AI is evident in its extensive model hub, where developers can share and access a wide range of models and datasets.
- **OpenCV** [98]: It is one of the most well-known and widely used computer vision libraries in the world. Its extensive collection of algorithms and tools makes it a go-to choose for both beginners and experts in the field of computer vision. Since OpenCV 3.1 there is DNN module in the library that implements inferencing with deep AI networks, facilitating the integration of AI solutions in computer vision applications.
- **Gymnasium** [99]: Gymnasium is an open-source Python library for developing and comparing reinforcement learning algorithms by providing a standard API to communicate between learning algorithms and environments, as well as a standard set of environments compliant with that API. Gymnasium is a fork of OpenAI's Gym library.
- **NVIDIA Jetson open-source components** [100]: Although the hardware is proprietary, some tools and libraries associated with Jetson are open-source. For example, jetson-stats is an open-source package for monitoring and controlling Jetson devices.
- **PYNQ** [101]: An open-source project from Xilinx that makes it easy to design embedded systems with Python and programmable logic.

- **Eyes of Things (EoT) Project** [102]: EoT was an Innovation Action funded under the European Union's H2020 Framework Programme that built an open embedded computer vision platform with AI capabilities. The generated code was made public at the end of the project.
- **Lightweight Kubernetes (K3s):** K3s is a certified lightweight Kubernetes distribution that automates application software deployment to edge AI devices.

Within the ongoing AI arms race (present almost every day in mass media), big companies are eager to get traction and publicity via open sourcing part of their work. Independent of this, the open-source paradigm can indeed be profitable within the field of AI. Take for example the case of company OpenAI, which started with the explicit aim to develop open and safe artificial intelligence for humanity. A few years later, however, OpenAI transitioned to a capped-profit model, citing the need for substantial funding and competitive advantage [108]. This shift highlights the tension between open-source ideals and the practical demands of sustaining advanced AI research and development. Balancing openness with financial viability remains a significant challenge in the AI industry.

However, what truly constitutes "open-source" in the context of AI remains an open question. Addressing this question is crucial because it impacts how AI technologies are developed, shared, and used globally. Key points of the discussion to define open-source AI are [86]:

- **Accessibility:** Open-source AI tools allow individuals and organizations with limited resources to experiment with and develop AI technologies.
- **Collaboration:** Open-source projects foster a collaborative environment where developers can share ideas, improve existing tools, and create innovative solutions.
- **Transparency:** Open-source software and hardware promote transparency, enabling users to understand how AI models work and ensuring that they can be audited for fairness and bias.
- **Customization:** Users can modify open-source tools to suit their specific needs, leading to more appropriate and effective AI solutions.
- **Cost-Effectiveness:** Open-source projects reduce the cost of development by providing free access to high-quality tools and resources.
- **Ethical and Legal Considerations:** This includes concerns about the misuse of AI technologies and the need for regulations to ensure responsible use.

The need for a specific definition of open-source AI arises from the unique characteristics of AI systems compared to traditional software. AI systems involve not just code, but also models, training data, and weights. Without a clear definition, some entities might misuse the term "open-source" for marketing purposes, leading to confusion and potentially undermining trust in AI technologies.

The Open source Initiative (OSI) has been working since mid-2023 to create a clear definition of open-source AI [89], based on OECD's definition of AI [81]. This involves ensuring that AI systems are transparent, modifiable, and shareable, adhering to the principles of open-source software. OSI has brought together researchers, lawyers, policymakers, activists, and representatives from major tech companies like Meta, Google, and Amazon [86]. Having a diverse group is essential to ensure that the definition is robust and widely accepted, helping to find a balance between different perspectives and interests [92].

The OSI's "Open-Source AI Definition" emphasises four fundamental freedoms [89]:

- **Freely Usable:** The AI components should be freely usable by anyone for any purpose.
- **Understandable:** Users need to know how AI systems were created and work.
- **Modifiable:** Users should be able to modify AI systems to suit their needs.
- **Shareable:** AI systems, with or without modifications, should be shareable with others, promoting further innovation and collaboration.

Additionally, AI systems require transparency regarding their training data, source code, and models, ensuring that researchers can inspect and understand the systems. This level of transparency is crucial for building trust in AI systems to reuse and modify them. To modify an AI system, the following needs to be considered [89]:

- **Data Information:** Detailed data information is required to recreate an AI system. This includes training methodologies, data sets, gathering process, scope, characteristics, selection process, labelling, and curating methods. Data must be available under licenses that comply with the Open-Source Definition.
- **Code:** Source code for training and running the AI system must be available with OSI-approved licenses.

- **Weights:** Model weights and parameters must be available under OSI-approved terms.

The current draft also makes a distinction between open-source AI models and weights. An AI model includes the architecture, parameters (weights), and inference code, while AI weights are the learned parameters that overlay the model architecture.

The convergence of open-source software and AI is driving rapid advancements in several different sectors. The open-source AI approach comes with high innovation potential, in both the public and private sector, thanks to the capacity and uptake of individuals and organisations to freely reuse the software under open-source licences. Advantages include the ability to enhance transparency, facilitate the auditing of AI and thereby enhance citizen trust, while stimulating economic activities and domain-specific expertise. Disadvantages and limits include legal, technical, data, risk management, societal and ethical challenges. Several open-source AI pro and cons are identified and seven recommendations to boost its uptake are proposed in [66].

2.1.3 Edge Computing

Edge computing is a distributed computing paradigm that brings computation and data storage closer to the location where they are needed to improve response times and save bandwidth. Edge computing processes data locally on or near the device that generated it, such as IoT devices, sensors, or local on-premises servers.

The key features of edge computing are the proximity to where the data is generated reducing latency and enabling faster processing, while providing bandwidth efficiency by reducing the amount of data that needs to be transmitted to a central data processing. This allows for real-time processing with minimal delays, scalable growth as devices and data volumes increase and enhanced security and privacy by keeping sensitive data closer to its source and controlling data flow to centralised systems. The following sub-sections provide the definitions and the descriptions of the concepts of micro-, deep- and meta-edge developed in [75, 76].

2.1.4 Micro-Edge

Micro-edge is defined as the miniaturised version of edge computing, implemented where the sensing/actuating computational resources and capabilities

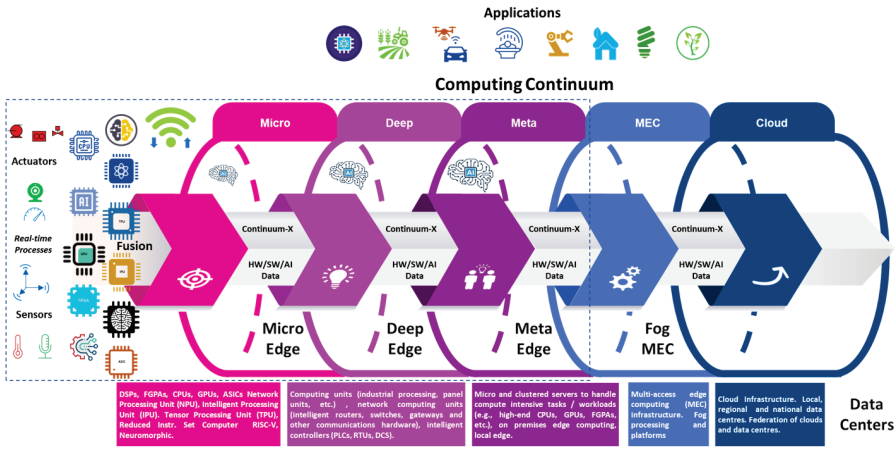

Figure 2.1 Edge granularity [75].

are deployed into microdevices. The concept pushes processing and analytics to be performed on micro, lightweight, and outer resource-constrained devices compared to single-board computers and gateways. These devices may distil information from the data collected by integrating AI-based algorithms for inference and training. Intelligent micro-edge allows real-time applications to become omnipresent and react quickly and intelligently to inferred situations while consuming minimal power.

The micro-edge includes intelligent sensors (physical, chemical, environmental parameters, perception, etc.) and actuators (audio, motion, etc.) systems with processing and connectivity capabilities that generate insight data and analytics. Micro-edge devices are implemented using microcontrollers built around ARM Cortex M0, M0+, M3, M4, M7X, ASICs and RISC-V. The distance from the data source (sensors/actuators) is minimised, and the micro-edge devices have extreme constraints on cost and power consumption [75, 76].

2.1.5 Deep-Edge

Deep-edge involves deploying advanced computational resources and AI capabilities at the edges of networks. The deep-edge model emphasises processing data in real-time, close to the data source, involving more complex computationally intensive applications than the micro-edge. Deep-edge

devices have a higher power budget since they are shared across multiple endpoints and generally have a more capable processor which can be multiplexed to support those endpoints.

The deep-edge comprises intelligent controllers, PLCs, SCADA elements, connected machine vision systems, networking equipment, gateways, and computing units that aggregate data from the sensors/actuators and IoT devices. Deep-edge processing capabilities are implemented with performant processors and microcontrollers, such as Intel i-series, Atom, ARM Cortex M7+, etc., including CPUs, GPUs, TPUs, FPGAs and ASICs. The system architecture, including the deep-edge, relies on foreseen functionality and deployment options. These functions include cognitive capabilities that can acquire, aggregate, understand, react to data, exchange, and distribute information.

Deep-edge computing enables a new level of AI-driven analysis and autonomy at the extreme edges of networks, supporting the growing demand for responsive, intelligent, and decentralised data processing across various sectors [75, 76].

2.1.6 Meta-Edge

Meta-edge computing enables a high level of edge processing and AI-driven analysis on high performance processors and on-premises servers emphasising enhanced integration, scalability, and coordination across edge computing continuum. Meta-edge offers adaptability to changing conditions and requirements, such as fluctuating workloads, network conditions, or data flows, adjusting resource deployment and task execution accordingly while generating higher-level insights or decisions that require understanding or context from broader data sets.

The meta-edge integrates processing units, typically on-premises, implemented with high-performance embedded computing units, edge machine vision systems, and edge servers (e.g., high-performance CPUs, GPUs, FPGAs, etc.), which are designed to handle compute-intensive tasks (e.g., data series, image, and video processing), advanced analytics, AI-based functions, networking, and data storage.

Meta-edge aggregates data and insights from multiple edge nodes, creating a meta-level understanding or intelligence that can inform broader network or enterprise-level decisions [75, 76].

2.1.7 Edge AI

Edge AI combines AI, IoT and edge computing technologies and provides real-time collection, computing, and analytics, allowing data processing and execution of AI algorithms to occur directly on devices at the network's edge. By bringing AI closer to the source of data generation, edge AI enables more efficient and responsive decision-making in various applications.

Edge AI uses AI-based algorithms and techniques on edge devices, enabling more rapid and efficient data processing and improved privacy and security. Edge AI provides computational intelligence to develop intelligent systems that perform real-time tasks that typically require human-level intelligence, such as decision-making, problem-solving, pattern recognition, and learning.

By combining AI's analytical capabilities with the IoT's sensing/actuating capabilities, intelligent connectivity, and the distributed architecture of edge computing, edge AI offers robust, efficient, and secure solutions across diverse industries, supporting the growing demand for intelligent and autonomous systems.

2.1.8 Edge AI System Dependability

The International Electrotechnical Commission (IEC) addresses the dependability concept through the Technical Committee TC 56, which develops and maintains international standards. These standards provide systematic methods and tools for assessing and managing the dependability of equipment, services, and systems throughout their life cycles. Based on the vocabulary provided by IEC in this work, edge AI dependability is defined as the ability of the edge AI system to perform as and when required, operate as desired, and satisfy the defined functional and non-functional requirements. The term edge AI system dependability includes the core system's attributes and characteristics such as availability, connectability, maintainability, privacy, reliability, resilience, safety, security, to which AI specific attributes and characteristics are added, e.g., accountability, accuracy, authenticity, credibility, controllability, compliance, durability, efficiency, explainability fairness, inclusiveness, integrity, interpretability, interoperability, learnability, performance, robustness, transparency, understandability, usability.

The attributes and characteristics of edge AI system dependability can be expressed qualitatively or quantitatively through expressing the ability to perform by defining the functional and non-functional requirements in terms of features, functions and qualities to be performed, when the performance

is to be achieved, considering operational conditions, the KPIs and measures specified.

Following the concept in [2, 5] a systematic structure of the concepts of edge AI dependability consists of three elements: attributes, threats, and means [4].

The attributes or characteristics are represented by the ways to assess the dependability of an edge AI system. These attributes or characteristics are qualities of an edge AI system. These can be assessed to determine its overall dependability by defined KPIs using qualitative or quantitative measures.

The threats are represented by the elements that can affect the dependability of an edge AI system (errors, failures, faults). An error is a discrepancy between an edge AI system's intended behaviour and actual behaviour inside the edge AI system boundary. Errors occur at runtime when some part of the edge AI system enters an unexpected state due to the activation of a fault. Errors are generated from invalid states and algorithms and are challenging to find without unique mechanisms. A failure is an instance in time when an edge AI system displays behaviour that is errant from its specification. An error may not necessarily cause a failure, but the edge AI system's performance is influenced. A fault is a defect in an edge AI system, and its presence may or may not lead to a failure due to input and state conditions that may never cause the fault to be executed so that an error occurs.

The means are represented by the ways to increase an edge AI system's dependability (forecasting, prevention, removal, tolerance) and are intended to reduce the number of failures made visible to the users of an edge AI system.

2.1.9 Edge AI System Trustworthiness

Trustworthy edge AI systems are essential for user confidence, the safe deployment of edge AI technologies including generative AI, and ensuring that these systems enhance technological, economic and societal progress.

According to [17], trustworthy AI should be lawful by respecting all applicable laws and regulations, ethical by respecting ethical principles and values, and robust from a technical perspective while considering its social environment. Trustworthy AI is dependent upon consistent and veracious application of these three underpinning principles from conception through to application [151].

The Dictionary.com defines "trustworthiness" as the quality of being deserving of trust or confidence, dependability. The National Institute of Standards and Technology (NIST) [19] defines "trustworthiness" as the following: "trustworthiness is the demonstrable likelihood that the system performs according to designed behaviour under any set of conditions as evidenced by characteristics including, but not limited to, safety, security, privacy, reliability and resilience. In computer security, a chain of trust is established by validating each component of hardware and software from the bottom up. It is intended to ensure that only trusted software and hardware can be used while still retaining some level of flexibility".

Based on the definition of trustworthiness in ISO/IEC TS 5723:2022 [42], edge AI trustworthiness can be defined as the ability of the edge AI system to meet the functional and non-functional requirements and user expectations in a verifiable way, which includes measurability and demonstrability by means of objective evidence that needs verification to ensure that user expectations are met. The verification process considers the definition of KPIs, measures, monitoring, ongoing assessment, transparency, and alignment with societal values and needs.

ISO/IEC TS 5723:2022 highlights that trustworthiness characteristics include accountability, accuracy, authenticity, availability, controllability, integrity, quality, reliability, privacy, resilience, robustness, safety, security, transparency, and usability.

Trustworthiness is an attribute that can be applied to edge AI systems and related services, products, technology, data and information.

ISO/IEC TR 24028:2020 [43] surveys topics related to trustworthiness in AI systems, including approaches to establish trust in AI systems through transparency, explainability, controllability, etc.; engineering pitfalls and typical associated threats and risks to AI systems, along with possible mitigation techniques and methods; and approaches to assess and achieve availability, resiliency, reliability, accuracy, safety, security and privacy of AI systems that can be applied as well to edge AI systems. The specification of levels of trustworthiness for AI systems is out of the scope of the document.

2.2 AI and Edge AI Taxonomy

The AI technology foundation has evolved over the years with the development of new technologies such as machine learning, deep learning, and generative AI, as illustrated in the Figure 2.2.

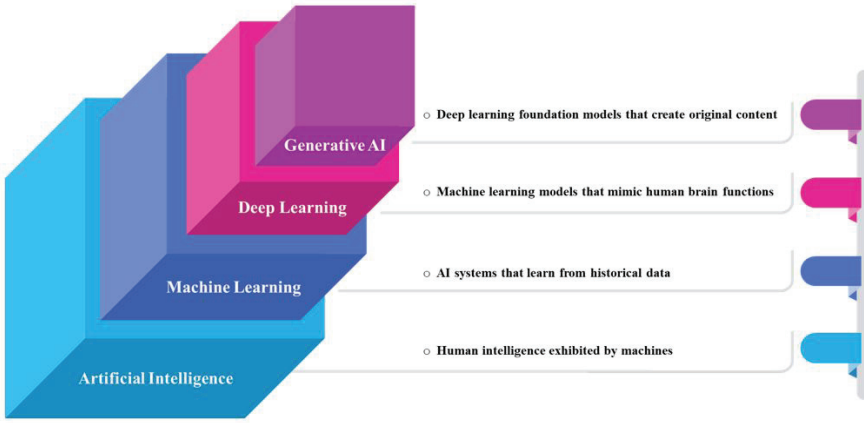

Figure 2.2 AI technology evolution.

Presenting a taxonomy and classification of AI and edge AI systems concepts is essential for understanding the landscape of edge deployments and defining the functional and non-functional requirements of such systems. These classifications guide the design of hardware, software, edge AI technology stacks, and datasets, and inform the strategic deployment of edge AI solutions across various industry domains. This helps maximise the potential of AI at the edge while addressing specific challenges and requirements of localised processing.

Creating a taxonomy and classification for edge AI systems involves categorising them based on various dimensions, including the specific AI and edge AI domains they serve. In this work, the concepts used are defined as follows:

Edge AI: is a distributed computing paradigm that integrates AI algorithms with IoT sensing and edge computing infrastructure to process data locally at the source where data is generated. Edge AI enables real-time analytics, intelligent decision-making, and enhanced privacy by executing computational tasks directly on edge devices rather than relying on centralised cloud connectivity. Edge AI can leverage generative and agentic AI to enable autonomous systems at the edge.

Machine Learning (ML): Edge AI domain that addresses the deployment of ML models optimised to fit edge devices with limited computational power, memory, and energy constraints. To maintain performance, this involves model compression techniques like quantisation, pruning, and distillation.

ML is defined in [21] as the process of optimising model parameters using computational techniques, such that the model's behaviour reflects the data or experience.

Distributed Machine Learning (DML): Execution of training and inference processes across multiple machines. Parallelisation can be applied to datasets and models [90, 91]. In dataset parallelism, each node computes gradients and local statistics on a subset of the original dataset. The resulting updates are subsequently aggregated by a parameter server. In model parallelism, each node trains a specific part of the model, with the central server coordinating the integration of the different components. By combining the two approaches, distributed machine learning facilitates the training of complex models on large datasets that would be infeasible on a single machine.

Deep Learning (DL): An edge AI domain that involves deploying deep learning models on edge devices. DL, a subset of ML, utilises neural networks with multiple layers (deep neural networks) to model complex patterns and representations in data. DL models perform inference directly on edge devices, enabling real-time data processing and decision-making. DL is an approach to creating rich hierarchical representations by training neural networks with many hidden layers. This process allows the neural network to progressively refine its final output. DL can reduce or eliminate the need for feature engineering, as the most relevant features are identified automatically. However, DL can require significant time and computing resources [21].

Natural Language Processing (NLP): Edge AI domain that handles human language data for applications such as voice recognition, language translation, and sentiment analysis on edge devices. NLP is defined in [21] as system information processing based upon natural language understanding defined in ISO/IEC 2382:2015 [170] as natural language comprehension extraction of information, by a functional unit, from text or speech communicated to it in a natural language, and the production of a description for both the given text or speech and what it represents or natural language generation defined as a task of converting data carrying semantics into natural language.

Supervised Learning: Edge AI domain that focuses on implementing supervised learning techniques directly on edge devices. Supervised learning is a type of ML where models are trained on labelled datasets, meaning the input data is paired with the correct output. This training enables models to make predictions or decisions when encountering new, unlabelled data. Supervised learning is defined in [21] and [41] as "machine learning that

makes use of labelled data during training". In this context, ML models are trained with training data that include a known or determined output or target variable (the label). The value of the target variable for a given sample is also known as the ground truth. Depending on the task, labels can be of any type, including categorical, binary, or numeric values or structured objects (e.g., sequences, images, trees, or graphs). Labels can be part of the original dataset, but in many cases, they are determined manually or through other processes. Supervised learning can be used for classification and regression tasks, as well as for more complex structured prediction tasks [21].

Unsupervised Learning: Edge AI domain that focuses on implementing unsupervised learning techniques directly on edge devices. Unsupervised learning is a type of ML that infers patterns from unlabelled data, where the edge AI system learns the underlying structure without explicit output guidance. These systems can adapt to input data in real time, updating models as they encounter new patterns. Unsupervised ML is defined in [21, 41] as "machine learning that makes use of unlabelled data during training". Unsupervised ML can be useful in cases such as clustering, where the objective is to identify commonalities among the input data samples. Reducing the dimensionality of a training dataset is another application of unsupervised machine learning, in which the most statistically relevant features are determined regardless of any labels.

Semi-Supervised Learning: focuses on implementing semi-supervised learning techniques directly on edge devices. Semi-supervised ML is defined as "machine learning that makes use of both labelled and unlabelled data during training." It is a hybrid of supervised and unsupervised machine learning [21]. Semi-supervised machine learning is helpful when labelling all samples in an extensive training dataset would be prohibitive from a time or cost perspective.

Self-Supervised Learning: focuses on implementing self-supervised learning techniques that generate implicit labels from unstructured data, rather than relying on labelled datasets for supervisory signals.

Reinforcement Learning: uses AI models to improve decision-making through trial-and-error. Reinforcement learning is the process of training an agent(s) to interact with its environment to achieve a predefined goal [21]. In reinforcement learning, a machine learning agent(s) learns through an iterative process of trial and error. The goal of agent(s) is to find the strategy (i.e. build a model) for obtaining the best rewards from the environment. For

each trial (successful or not), the environment provides indirect feedback. Based on this feedback, the agent(s) then adjust their behaviour (i.e., their model) [21].

Federated Learning (FL): Federated learning is a decentralised machine learning approach in which multiple devices collaboratively train a model while each device sees only part of the data. Instead of sending data to a central server, each device trains the model locally using only its own data. The results are then sent to a central server, where they are aggregated to improve the global model. The rationales for federated learning often centre on privacy and/or certainty.

Transfer Learning: Refers to a series of methods where data intended for solving one problem is leveraged to apply the knowledge gained from it to a different problem [21]. For example, information gained from recognising house numbers in a street view can be used to recognise handwritten numbers.

Continuous Learning: Edge AI domain used for continuous AI model training. It is also known as continual learning or lifelong learning and is incremental training of a model that occurs continuously while the system is running in production. This is a particular case of retraining, where model updates are repeated, occur frequently, and do not interrupt operations [21]. In many AI systems, training occurs during development before deployment to production. This is like standard software development, where the system is built and tested fully before it is put into production. The behaviour of such systems is assessed during the verification process and is expected to remain unchanged during the operational phase. AI systems that embody continuous learning involve incremental updates to the model as it operates in production. The data input to the system during operation is analysed to produce an output and simultaneously used to adjust the model, improving it based on the production data. Continuous learning can help overcome the limitations of the original training data and address data and concept drift. However, it also presents significant challenges in ensuring that the AI system continues to operate correctly as it learns.

On-Device Training: Training a model directly on an edge device, like mobile phones, embedded systems, gaming consoles, web browsers, and more. This approach differs from training on a centralised server or in the cloud. On-device training is particularly valuable when data privacy is paramount and sharing it with external servers or clouds is not feasible. Additionally, it is advantageous for personalisation tasks, as the model can be

tailored and trained directly on the user's device. Existing on-device training approaches and solutions cover: (i) last-layer or bias-only backpropagation; (ii) sparse backpropagation; (iii) complete backpropagation for fine-tuning; and (iv) complete backpropagation with optimised memory consumption.

Training Data: Consists of data samples used to train an ML algorithm. Typically, the data samples relate to a topic of concern and can consist of structured or unstructured data, both unlabelled and labelled. In the latter case, the label guides the machine learning model's training process [21].

Trained model: A trained model is the result of model training, which, in turn, is defined as the process of establishing or improving the parameters of a machine learning model using a machine learning algorithm and training data [21]. An ML model is a mathematical construct that generates an inference or prediction from input data. An AI system should use the trained model to make predictions based on production data from the area of interest. Various standardised formats exist for storing and transmitting the trained model as a set of numbers.

Generative AI (GenAI): AI that can create original content such as text, images, video, audio or software code in response to a prompt or request. GenAI uses advanced DL models, algorithms and transformers that simulate the learning and decision-making processes of the human brain. These models work by identifying and encoding patterns and relationships in large amounts of data, then using that information to understand users' natural language requests or questions and respond by creating new content based on learned patterns.

Diffusion Models: Generative models used primarily for image generation and other computer vision tasks. Diffusion-based neural networks are trained through deep learning to progressively "diffuse" samples with random noise, then reverse that diffusion process to generate high-quality images.

Variational Autoencoders (VAEs): Generative models used in ML to generate new data in the form of variations of the input data they're trained on. VAEs can perform tasks common to other autoencoders, such as denoising.

Transformers: Neural network architectures that transform or change an input sequence into an output sequence. The transformer models are trained on sequential data to generate extended sequences of content, such as shapes in an image, words in a sentence, frames of a video, commands in software code, etc. The transformer model uses an internal mathematical

representation to identify the relevance and relationships between the content, and it uses that knowledge to generate the output.

Multimodal AI: Refers to machine learning models capable of processing and integrating information from multiple modalities or types of data. These modalities can include text, images, audio, video and other forms of sensory input. Multimodal AI combines and analyses different forms of data inputs to achieve a more comprehensive understanding and generate more robust outputs.

Retrieval Augmented Generation (RAG): Architecture for optimising the performance of an AI model by connecting it with external knowledge bases and supporting LLMs and SLMs in generating more relevant responses of higher quality. RAG enhances AI applications by retrieving relevant information from databases, documents, or the web and supplying it as real-time context to LLMs, improving accuracy and relevance without requiring retraining.

Generative Adversarial Network (GAN): ML model designed to generate realistic data by learning patterns from existing training datasets. It operates within an unsupervised learning framework using DL techniques, where two neural networks work in opposition: one generates data, and the other evaluates whether it is real or generated.

Large Language Models (LLMs): A category of deep learning models trained on very large amounts of data, making them capable of understanding and generating natural language and other types of content to perform a wide range of tasks by including processing and analysing long text sequences. LLMs are built on neural network architectures such as transformers, which excel at handling sequences of words and capturing patterns in text. LLMs can understand context, generate human-like text, translate languages, and even write various types of creative content.

Small Language Models (SLMs): Smaller AI models capable of processing, understanding and generating natural language content and can be deployed at the edge. SLM parameters, which are internal variables, such as weights and biases, that the model learns during training, range from a few million to a few billion.

Vision Language Models (VLMs): VLMs are AI models that combine and mix computer vision and NLP capabilities and learn to map the relationships between text data and visual data, such as images or videos and generate text

from visual inputs or understand natural language prompts in the context of visual information. The VLMs combine LLMs and SLMs with vision models or ML algorithms. VLMs are multimodal AI systems that use text and images or videos as input and produce text as output, in the form of image or video descriptions, answering questions about an image or identifying parts of an image or objects in a video.

Agentic AI: Refers to the broader class of AI and edge systems designed with intrinsic agency that can accomplish a specific goal with limited supervision. Agentic AI possesses the capacity to initiate actions, maintain persistent memory of its state, and pursue abstract goals over extended horizons. It consists of AI agents, ML models that mimic human decision-making to solve problems in real time. In a MAS, each agent performs a specific subtask required to reach the goal and their efforts are coordinated through AI orchestration. Agentic architecture supports and regulates the behaviour of AI-powered agents operating within a GenAI system. Agentic AI systems require their agents to be adaptive and navigate dynamic environments to achieve desired outcomes. Agentic AI exhibits autonomy, goal-driven behaviour and adaptability. The term "agentic" refers to these models' agency, or the capacity to act independently and goal-oriented by using generated content from LLMs, SLMs, and VLMs to complete complex tasks autonomously by calling external tools. The functions used in implementing agentic AI systems are perception and input handling, planning and task decomposition, reasoning and decision-making, action and tool calling, communication, learning and adaptation and memory.

AI Agents: The discrete software implementations of agentic AI. Agents are characterised by their autonomy loop: perceive, reason, decide, act and memorise. is a system that autonomously performs tasks by designing workflows with available tools and can encompass a wide range of functions beyond natural language processing, including decision-making, problem-solving, interacting with external environments and performing actions. AI agents solve complex tasks across applications, including software design, IT automation, code generation and conversational assistance. AI agents use advanced natural language processing techniques of LLMs, SLMs, and VLMs to comprehend and respond to user inputs step-by-step and determine when to call on external tools. AI agents are classified based on their level of intelligence, decision-making processes, and how they interact with their surroundings to achieve desired outcomes, in categories such as simple reflex agents, model-based reflex agents, goal-based agents, utility-based agents,

and learning agents. Some agents operate purely on predefined rules, while others use learning algorithms to refine their behaviour.

Multi-Agent Systems (MAS): MAS consists of multiple artificial intelligence (AI) agents working collectively to perform tasks on behalf of a user or another system. Each agent in a MAS has individual properties, but all agents collaborate to achieve desired global properties. Multi-agent systems are effective at completing large-scale, complex tasks that involve many agents.

Multi-Agentic AI: Involves the interaction of multiple autonomous agents sharing a common environment. This field studies the dynamics of cooperation, competition, and coordination. On the road, traffic is inherently a multi-agent system where every vehicle, pedestrian, and cyclist is an independent agent with its own objectives. Multi-Agentic AI models aim to predict the collective behaviour of these entities to enable safe and efficient navigation, moving beyond simple obstacle avoidance to sophisticated social negotiation.

Agentic RAG: Defined as the use of AI agents to support RAG. Agentic RAG systems add AI agents to the RAG pipeline to increase adaptability and accuracy, enabling LLMs and SLMs to retrieve information from multiple sources and handle extended, complex workflows.

Distributed Intelligence: Refers to systems where multiple agents or entities collaborate and share information to achieve a common goal or solve complex problems. It leverages the collective capabilities of these agents, for example, through distributed computing, to enhance overall performance, adaptability, and scalability. The concept is often applied in Multi-Agent Systems (MAS), enabling them to learn from each other's experiences and improve their collective perception and decision-making. It allows for more adaptive, scalable, and complex problem-solving with minimal human intervention.

First Principles: Edge AI domain that arises from the fundamental properties and behaviours of persisting physical systems, such as energy, entropy, and the ability to maintain and propagate information over time. By deducing intelligence from these first principles, we understand it as the optimisation of actions and decisions that align with the natural laws governing the system's environment and as an emergent capacity of a system to process information, adapt, and achieve goals within the constraints of a physical universe [77].

Computer Vision: Edge AI domain that processes visual information from cameras, sensors, or is stored on a device at the edge, enabling applications like facial recognition, object detection, and video analytics.

Machine Vision: A branch of engineering that designs and implements systems that allow machines to interpret visual information locally on the device. Machine Vision systems include optical sensors, hardware, algorithms, data storage and processing. Enables instant analysis and action based on visual input, which is essential for applications such as autonomous vehicles, manufacturing quality inspection, and live security monitoring.

Speech Systems: Edge AI domain that addresses the deployment of speech recognition, processing, and generation technologies using AI directly on edge audio devices. The approach facilitates real-time processing and analysis of audio data at the point of capture, leveraging the benefits of edge computing.

Predictive Analytics: Edge AI domain that utilises historical data to forecast future outcomes and provide predictive maintenance and anomaly detection.

Robotics and Control Systems: Edge AI domain that implements real-time AI-driven decision-making for robotic movements and industrial automation.

Planning Optimisation and Scheduling Systems: An edge AI domain that focuses on deploying sophisticated algorithms to manage and optimise tasks, resources, and processes in real-time directly on edge devices. These systems aim to enhance operational efficiency, improve decision-making, and enable autonomous functionality across settings from industrial automation to smart home management.

2.3 Edge AI System Elements

The development of edge AI solutions leads to the emergence of multimodal edge AI implementations that yield real-time performance at the edge for various industrial sectors. These require the integration of edge AI hardware, software, AI technology stack building blocks and data in an edge AI design framework for the whole system, as illustrated in Figure 2.3.

Defining the non-functional and functional requirements, aligned with quality properties, supports to address holistically all the edge AI system components including hardware, software, AI techniques/methods and data and treat edge AI systems as a set of systems and system elements that interact to provide a unique intelligent capabilities that none of the constituent systems can accomplish on its own facilitating interaction of the constituent systems in this system of systems concept.

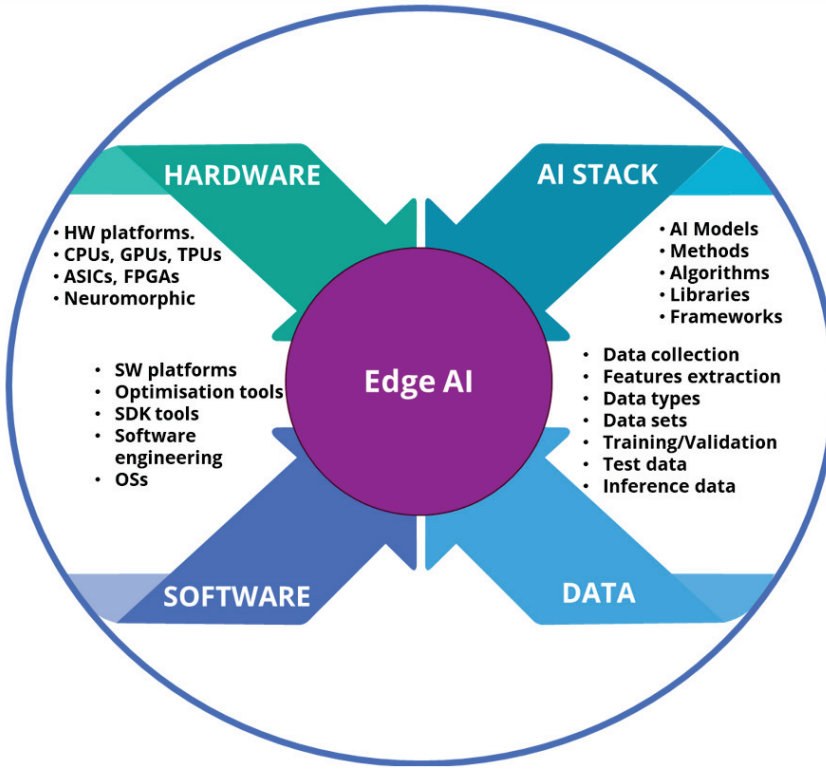

Figure 2.3 Edge AI system components.

Developing a conceptual framework for edge AI system of systems design and implementation is critical to be used as a reference for defining the functional and non-functional requirements of edge AI systems across the computing continuum and various industrial applications.

2.3.1 Edge AI Technology Stack

Edge AI technology encompasses a multi-layered approach that facilitates the deployment of AI capabilities directly at the edge. This architecture allows for efficient data processing, analysis, and decision-making closer to where data is generated, significantly reducing latency and bandwidth usage. A five-layer edge AI technology stack is presented in Figure 2.4 [69].

Edge AI hardware is the foundational layer, which contains at least three components reflecting the processing units responsible for performing

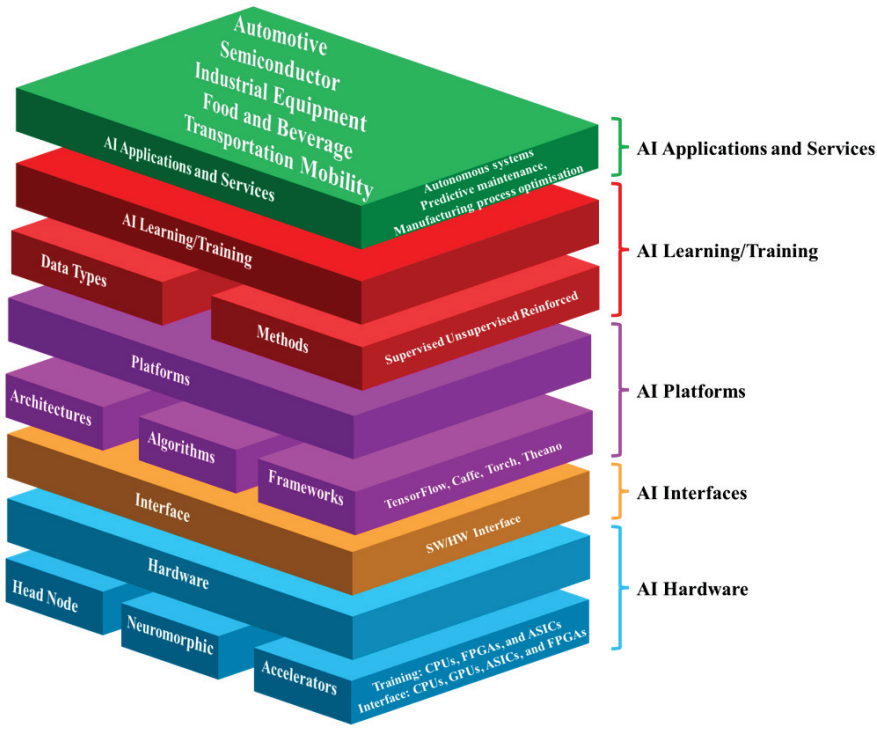

Figure 2.4 Edge AI technology stack layers [69].

specialised AI operations. The neuromorphic hardware components consist of new ultra-low-power silicon chip architectures (e.g., neuromorphic modules and chips, digital, analogue, spike NN) incorporating different chip designs and algorithms to mimic how the human brain works. The accelerator set of components consists of silicon chips designed to perform highly parallel functions and accelerate AI workloads required during training and inference, such as GPUs, NPUs, VPUs, TPUs, FPGAs, or ASICs. The HW layer includes in-memory computing architectures to support high-speed data processing within the chip, essential for handling real-time AI applications. Energy efficiency is critical for HW designs that enable AI calculations on remote edge devices without excessive energy consumption.

Edge AI interfaces layer covers the mechanisms through which edge devices communicate with each other and with other parts of the system, ensuring seamless data exchange and control. The head node components coordinate computations among accelerators and provides interfaces for

developers to integrate AI functionalities into devices and applications. Interfaces can include technology for preprocessing data locally.

Edge AI platforms provide the infrastructure for deploying and managing AI models on edge devices, incorporating tools for model development, optimisation, deployment, and monitoring. The platforms layer is used for AI and ML/DL deployment and consists of three sublayers, which aim to abstract firmware from the underlying hardware. The framework sublayer comprises packages that trigger AI frameworks, such as TensorFlow, TensorFlow Lite, Caffe, PyTorch, Theano, etc., via the interface layer, which connects the hardware and platform layers and facilitates communication. The algorithms sublayer consists of rules to achieve optimal inference according to the training method employed, such as backpropagation, evolutionary, and contrasted divergence. The architectures sublayer consists of many continuously evolving neural network architectures, such as CNN, RNN, etc.

The edge AI learning/training layer involves the methods and processes used to train AI models specifically for edge device deployment, emphasising the need for lightweight, adaptable models. The layer consists of two sublayers. The methods sublayer involves techniques for optimising the model for specific domain data, such as supervised, unsupervised, reinforcement, and first-principles learning. Techniques like pruning, quantisation, and knowledge distillation aimed at reducing model complexity and size while maintaining accuracy are part of this sublayer. Different learning approaches, such as federated learning, are included, where models are trained across numerous decentralised devices without sharing raw data, enhancing privacy and reducing the need for data transfer. The data type sublayer consists of categories of domain input data, such as labelled and unlabelled data.

The edge AI applications and services layer consists of end-user applications and services that leverage AI at the edge to deliver value and insights. Solutions can be customised based on generic data or customer-specific training data. The AI technology stack provides a common understanding of the AI layers and components when implementing and benchmarking various AI technologies and applications.

By integrating these layers, edge AI technology delivers intelligent solutions directly at the data generation and use site. This layered approach facilitates the development of edge AI applications tailored for specific tasks and the enhancement of existing systems with real-time learning, adaptability, and efficiency.

2.3.2 Hardware

Edge AI hardware refers to the computing devices designed to perform AI computations directly at the edge of networks, near the source of data generation across the micro-, deep- and meta-edge continuum. The primary goal of edge AI hardware is to enable real-time data processing, reduce latency, and enhance privacy by optimising and reducing data transmission.

Edge AI systems can run on various hardware platforms and take different forms, from central processing units (CPUs), and microcontroller units (MCUs) to advanced neural processing units (NPUs). The edge AI hardware is designed as compact system on a chip (SoC), system on modules (SoMs), and Application-Specific Integrated Circuits (ASICs) that consume minimal power, making it suitable for deployment in mobile, portable, or remote edge devices.

An ASIC is an integrated circuit customised for a particular use. ASICs are an option for providing AI-specific functionality. An ASIC can be customised as an accelerator to speed up the AI process by offering functions such as dedicated parallel multiplying-accumulating blocks, optimised memory allocation and lower precision arithmetic. ASICs provide a higher computational capability for AI with lower spatial volumes, cost and energy consumption. ASICs enable AI to be implemented in space- and power-constrained edge IoT devices [21].

An SoC is an integrated circuit integrating on a single substrate/chip or microchip multiple cores, including a mix of CPUs, GPUs, TPUs, and other types of functional units, e.g., digital signal processors (DSPs), neural processing units (NPUs), image signal processors, (ISPs), memory, input/output ports, secondary storage, and sometimes Wi-Fi and cellular modems. SoC organisation enhances performance and reduces energy consumption and semiconductor die area compared with motherboard-based architecture with equivalent functionality but discrete components.

A SoM is a board-level circuit that integrates an embedded processing system function in a single module that includes microprocessor cores, SoCs, memory blocks, communication interfaces, and hardware peripherals on a single production-ready substrate. SoMs offer a flexible and modular approach to system design.

The edge AI HW includes specialised processing units, such as AI accelerators implemented as GPUs (Graphics Processing Units), TPUs (Tensor Processing Units), NPUs, Vision Processing Units (VPUs) and neuromorphic

units, that are optimised for the parallel processing requirements of AI workloads. Field-Programmable Gate Arrays (FPGAs) and ASICs are also used to deliver high efficiency for specific AI tasks, providing customisable solutions tailored to various applications.

Increased performance edge AI HW is necessary to deal with the aggregation of multiple endpoints, multimodality of data and cross-inference from many sources. This increased functionality and breadth of scope also increases the consequence of faults, so the requirements for security and reliable operation also increase as one moves away from the micro-edge. New edge processing circuits and devices with novel processing architectures are emerging for edge AI implementations, including integrating several accelerators and processing units (CPUs, GPUs, TPUs, NPUs, ISPs, VPUs) into one chip or SoM circuit.

Edge AI implementations require low-power AI accelerator architectures (e.g., VPU, GPU, TPU) suitable for edge devices and the given application use case. They must also establish an appropriate deployment platform, learning frameworks (e.g., TensorFlow, PyTorch, TensorFlow Lite, ONNXruntime, TensorRT) that meets the required needs and a robust programming language (Python, C++) suitable for edge AI development and the platform/framework.

Edge AI hardware is equipped with the necessary computational power to run inference tasks locally, allowing devices to make predictions and decisions in real-time based on new data inputs. The HW supports various connectivity options, such as Wi-Fi, Bluetooth, and cellular networks, to communicate efficiently with other devices and systems. Deep- and meta-edge AI hardware can have the necessary computational power to partially and fully train AI models.

The HW provides the computational power necessary for real-time data analysis and decision-making based on AI algorithms and is optimised to reduce latency and increase bandwidth efficiency, which is critical for real-time applications. The HW facilitates easier scaling of AI applications, as deploying additional edge devices can enhance overall processing without overwhelming centralised resources.

As computing performance increases, edge AI hardware is advancing to enable the local processing of more complex AI models and larger datasets. This is driven by chip design and architecture innovations that optimise performance and energy consumption.

2.3.3 Software

Edge AI software and platforms provide the tools and frameworks to develop, deploy, and manage AI models directly on edge devices. These solutions effectively harness edge hardware's computational power, offering flexibility and efficiency in implementing AI applications across various environments.

Edge AI software platforms often include tools for model optimisation, such as pruning, quantisation, and knowledge distillation. These techniques reduce the size of AI models and improve their performance on resource-constrained devices without significantly compromising accuracy.

Edge AI software supports AI frameworks like TensorFlow, PyTorch, Keras, ONNX, and Caffe, enabling developers to use familiar tools and libraries while ensuring models can be executed efficiently on edge devices.

Software Development Kits (SDKs) and Application Programming Interfaces (APIs) are tailored for edge environments, simplifying integrating AI capabilities into applications.

Edge AI software can be categorised into various types based on functionality and use cases. Some common categories of edge AI software include:

- **Machine and Deep Learning Frameworks:** These libraries and tools allow developers to build, train, and deploy machine learning models.
- **Natural Language Processing (NLP) Software:** NLP software enables machines to understand, interpret, and generate human language.
- **Computer and Machine Vision Software:** These tools focus on image and video analysis, enabling AI systems to understand and interpret visual data.
- **Speech Recognition and Synthesis Software:** This category includes software that can transcribe spoken language into text and generate speech from written text.
- **Edge AI Chatbots and Virtual Assistants:** These applications use natural language processing and machine learning to interact with users and provide relevant information or services.
- **Reinforcement Learning Frameworks:** These tools are designed to develop edge AI systems that learn through trial and error, often used in robotics and autonomous applications.
- **Active inference frameworks:** These tools allow to formulate agents with adaptive behaviour to the deployment scenario, which inherits from

the physics of self-organization, i.e., first-principles. These tools enable maximizing (Bayesian) model evidence, via inference, learning, and model selection. Edge AI Development Platforms: These platforms offer end-to-end solutions for building, training, and deploying AI models, often with pre-built models and drag-and-drop interfaces.

- **Edge AI Data Annotation Tools:** These tools help label and annotate data to train edge AI models.
- **Edge AI-based Business Solutions:** These are edge AI applications designed to solve specific business problems.

Edge AI platforms often support deployment across heterogeneous devices, ensuring that AI models work cohesively within diverse hardware ecosystems.

Edge AI software and platforms support edge AI system developers in creating robust, scalable, and efficient AI solutions that leverage the full potential of edge computing. By focusing on optimisation, compatibility, and ease of deployment, these tools make it possible to meet the varied demands of real-world applications that require speed, reliability, and autonomy.

Example of edge AI software platforms:

ST Edge AI Suite: This suite offers a set of tools and software solutions to simplify the development and deployment of AI applications on edge devices, particularly those powered by STM's microcontrollers and microprocessors. The suite caters to developers looking to implement AI capabilities such as machine learning and neural network inference directly on low-power, resource-constrained devices.

eIQ® Auto Machine Learning (ML) Toolkit: Offers a suite of tools aimed at simplifying and accelerating the deployment of machine learning models on edge devices, particularly within the automotive sector. This toolkit is part of NXP's broader eIQ Machine Learning Software Development Environment, designed to cater to various ML applications across diverse hardware platforms offered by NXP.

Google Coral: Provides the Edge TPU with an SDK for accelerating ML inferencing at the edge, supporting TensorFlow Lite models.

NVIDIA Jetson: Offers a comprehensive platform with SDKs like JetPack and deep learning tools that support high-performance AI applications.

Azure IoT Edge: Microsoft's platform for deploying AI workloads on the edge, integrated with Azure cloud services for a seamless hybrid solution.

AWS Greengrass: Facilitates the execution of local computing, messaging, and machine learning inference capabilities on connected devices.

OpenVINO Toolkit: Developed by Intel, optimises AI model performance on Intel hardware across CPUs, VPUs, FPGAs, and integrated GPUs.

2.3.4 Edge AI Frameworks, Methods, and Techniques

Edge AI frameworks are software libraries and tools designed to simplify and facilitate the deployment, development, and optimisation of edge AI models on edge devices. These frameworks address the unique challenges and requirements of edge computing, where resources such as processing power, memory, and energy are often constrained.

Edge AI frameworks enable the deployment of pre-trained AI models on various edge devices, ensuring they perform efficiently in resource-constrained environments by using techniques like quantisation, pruning, and sparsification to reduce the size and computational requirements of AI models, making them suitable for edge conditions. This ensures compatibility with machine learning libraries and frameworks like TensorFlow, PyTorch, and ONNX, allowing developers to leverage existing models and facilitate efficient data preprocessing and handling on edge devices to support real-time analytics and decision-making.

The edge AI frameworks support multiple AI models, including deep learning neural networks, machine learning algorithms, and computer vision models. They are designed with a small footprint to operate effectively on devices with limited resources without compromising performance.

TensorFlow Lite: An extension of TensorFlow designed to run machine learning models on mobile and edge devices. It supports model optimisation through quantisation and provides an interpreter capable of running efficiently on various types of hardware accelerators.

ONNX Runtime: An open-source inference engine focused on providing performance-enhanced runtime options for models expressed in the ONNX format. It provides extensive hardware compatibility, supporting CPU, GPU, and custom accelerators.

Apache MXNet: A flexible and efficient deep learning framework supported by open-source. With its Gluon API, MXNet is efficient for training and deploying models on edge devices in real time.

NVIDIA TensorRT: A high-performance deep learning inference library specifically for NVIDIA GPUs and designed to accelerate inference for deep learning models.

OpenVINO Toolkit: Developed by Intel, it optimises and deploys AI inference across Intel architectures, including CPUs, integrated GPUs, and FPGAs, primarily for vision applications.

Arm NN: Provides an inference engine optimised for Arm Cortex CPUs and GPUs, enabling efficient execution of neural networks on edge devices.

2.3.5 Data

Data plays a multifaceted role in edge AI systems, serving as the foundation upon which these systems operate and derive insights. The characteristics and handling of data significantly influence the performance, reliability, and efficiency of edge AI applications.

Several challenges relate to data in edge AI, among them data quality, volume, and integration. Data quality ensures that the data used for training and inference is accurate and representative of real-world conditions, which is vital for model reliability. Data volume requires handling volumes of data generated by numerous devices to match edge devices' processing power and storage capacity. Integrating data from diverse sources and formats is complex in heterogeneous environments with different types of devices and sensors requiring new multi-modality techniques and algorithms.

Data is driving the ability of edge AI systems to learn, adapt, and make informed decisions quickly and efficiently. As edge AI continues to evolve, strategies to optimise data usage focus on quality, security, and efficient processing.

The datasets used in edge AI applications span various domains and can differ based on the application and field for which the edge AI model is being developed. Several categories of edge AI datasets based on their applications are presented below:

Anomaly Detection Datasets are designed to train edge AI models to identify anomalies or outliers in various types of data, such as electrical signals, network traffic, industrial machinery, motors, or the behaviour of edge devices.

Audio and Speech Datasets encompass audio recordings paired with corresponding transcriptions or labels. These datasets are essential for developing

edge AI models for speech recognition, keyword spotting, speaker identification, and other audio-related tasks.

Environmental Datasets consist of data collected from sensors that monitor environmental parameters such as temperature, humidity, and air quality. These datasets are utilised for applications in environmental monitoring and sustainability.

Gesture and Motion Datasets involve data captured from sensors like accelerometers and gyroscopes to recognise human gestures or movements. These datasets are used in gesture and human activity recognition applications.

Healthcare Datasets focus on applications related to healthcare, including medical image analysis, disease diagnosis, and patient monitoring.

Image Classification Datasets consist of images labelled with corresponding class identifiers. These datasets are used to train and evaluate edge AI models for tasks such as object recognition and scene classification.

Natural Language Processing (NLP) Datasets for edge AI models contain text data and corresponding labels or annotations. These datasets are applied in text classification, sentiment analysis, and named entity recognition tasks.

Object Detection Datasets feature images annotated with bounding boxes around objects of interest. These datasets are essential for training edge AI models to detect and localise one or multiple objects within an image. Object detection is a critical technique in computer vision that involves identifying relevant objects in images and video frames. The algorithms used for object detection employ advanced machine learning and deep learning architectures to analyse image data and recognise and pinpoint objects of interest. The output typically includes the object's name and a bounding box indicating its location. Deep learning architectures for object detection include Single Shot Detector (SSD), You Only Look Once (YOLO), and Region-based Convolutional Neural Networks (R-CNN), among others.

Semantic Segmentation Datasets provide pixel-level annotations for each image, indicating the class to which each pixel belongs. These datasets are utilised for tasks like image segmentation and instance segmentation.

Time Series Datasets are employed across edge AI applications, including IoT sensor data, equipment health monitoring, predictive maintenance, and forecasting. These datasets contain sequential data points, including timestamps and associated target values.