1

Industrial AI Technologies for Next-Generation Autonomous Operations with Sustainable Performance

Ovidiu Vermesan[1], Frédéric Pétrot[2], Marcello Coppola[3], Mathias Schneider[4], Alfred Höß[4]

[1]SINTEF AS, Norway
[2]University Grenoble Alpes, CNRS, Grenoble INP, TIMA, France
[3]STMicroelectronics, France
[4]Ostbayerische Technische Hochschule Amberg-Weiden, Germany

Abstract

This book lays down the technological foundation for and introduces key artificial intelligence (AI) concepts and technologies for the digitising industry. While this chapter does not exhaustively cover all types of AI, it comprehensively prioritises the features of AI-based industrial applications and designs and defines the reference terminology used in the other chapters of the book.

AI integrates several interrelated technologies to solve problems and perform tasks to achieve defined objectives; hence, AI can be approached from many viewpoints, such as mathematics and computer science, linguistics, psychology, neurology, and philosophy. The approach in this chapter is from a technological and industrial perspective, and concepts and functions are presented intuitively and visually, focusing on AI, as it is applied to embedded systems, with industrial automation, interactivity, and sustainability in mind.

This already reflects the next-generation deployment of AI into edge devices (called edge AI) and the emergence of different edge layers (i.e., micro-, deep- and meta-edge), which contrasts existing solutions that are currently deployed in the cloud. The edge processing continuum includes sensing, processing and communication devices (micro-edge) close to the physical industrial assets under monitoring, gateways and intelligent controller processing devices (deep-edge) and on-premise multi-use computing devices (meta-edge).

Furthermore, instead of attempting to present a definition of AI that is common to all industries, the chapter relies on a framework of classifications and continuums along various dimensions, including the industrial intelligence spectrum, the intelligent capabilities spectrum, the edge-cloud continuum, the symbolic reasoning – pattern recognition continuum and, not the least, the problem-solving spectrum. The chapter introduces some of the main pillars of problem solving, such as expert systems, genetic and evolutionary computation, intelligent agents, machine learning (ML) and more.

This chapter, in particular, will detail ML approaches and neural networks. During the past decades, the trends and developments in AI have followed a recurring pattern, where the focus has moved back and forth between logic (symbolic reasoning) and pattern recognition (neural networks), driven by the varying abilities of technologies to acquire data, learn, derive new information and reason to reach decisions. In the last years, machine learning and neural network models have been the primary focus due to advances in hardware development and processing capabilities. Furthermore, embedded machine learning has been increasingly gaining popularity in industrial applications.

This chapter introduces several contributions. First, it gives a high-level overview of how AI works. Second, it shows how AI methods and techniques can be incorporated into an industrial design workflow. Finally, it provides a valuable intuitive understanding of how AI methods and techniques work when deployed in edge devices and how they operate in industrial settings.

Keywords: artificial intelligence, industrial AI, sustainable AI, intelligent embedded systems, symbolic AI, logic-based technologies, machine learning, AI-based problem-solving, AI technology stack, neural network architectures, embedded ML development.

1.1 Industrial AI

A recent report [1] predicted that the smart manufacturing market is expected to reach $446.24 billion by 2029, growing at a compound annual growth rate (CAGR) of 21.5% from 2022 to 2029. The smart manufacturing market is segmented into industrial Internet of Things (IIoT), cloud computing and storage, robotics and automation, industrial cybersecurity, additive manufacturing, augmented reality (AR), virtual reality (VR), digital twin, artificial intelligence (AI) and blockchain-based technology. In 2022, the industrial IoT segment is expected to account for the largest share of the smart manufacturing market. The large market share of this segment is attributed to factors such as the consistent declining cost of industrial IoT sensors, the significant rise in overall equipment effectiveness (OEE) through industrial IoT usage and increasing government initiatives to promote digital transformation. According to another market research report [2], the industrial AI market is expected to grow from $1,482.50 million in 2021 to 17,925.50 million by 2028 at a CAGR of 51.50% during the forecast period of 2022–2028.

Industrial AI refers to the application of AI in various industrial sectors and is considered a game changer in the manufacturing industry. The transition to Industry 5.0 is likely to drive the market's growth in the next few years. In manufacturing plants, the information obtained from various sensors, software and IIoT-driven systems may become too complex for humans to analyse. The use of AI is an efficient solution that can assist the manufacturing sector in transforming completely through machine learning and pattern recognition. The use of AI in manufacturing plants allows users to analyse and predict user behaviour, to perform predictive maintenance to prevent unwanted shutdowns, detect abnormalities in the production process and much more. AI also facilitates the use of real-time information, which could improve decision-making time, lead to better fabrication quality and yield, and boost organisational growth. The increasing volume of data gathered through various devices together with the widespread availability of high-speed communication networks and the upcoming implementation of wireless technologies will contribute to the increased use of AI in manufacturing in the future.

Although embedded in an increasing variety of products, processes and services in many industrial sectors, AI remains difficult to define. In scientific terms, AI is for example defined as "The designing and building of intelligent agents that receive precepts from the environment and take actions that affect that environment." [8]. AI and machine intelligence can also be defined as follows: "Artificial Intelligence is [...] the study of the computations that make it possible to perceive, reason, and act." [4]; "[Intelligence is] the capability of a system to adapt its behaviour to meet its goals in a range of environments." [5]; "Intelligence measures an agent's ability to achieve goals in a wide range of environments." General definition: "A very general and flexible capacity to succeed when faced with a wide range of problems and situations." [6]; "Intelligence is the computational part of the ability to achieve goals in the world." [7]. In more general terms AI refers to the ability for machines, systems, models, computers, to be able to mimic and improve intelligence in general, and human intelligence in particular.

Currently, neither the industry nor the scientific community has agreed on a particular definition. The presently available definitions are too vague, or too broad or too narrow. This is largely because of the growing variety and specific properties of AI technologies and partly because of the convergence of multiple technologies in the last years into AI, such as semiconductor technologies, cyber-physical systems (CPSs), internet of things/industrial internet of things (IoT/IIoT), supervisory control and data acquisition (SCADA), programmable logic controllers (PLCs), 5G, distributed ledger technologies, edge computing, etc. The AI ecosystems are extending to related fields, such as edge computing, to address the challenges and requirements of various industrial sectors, and each field defines AI from its own perspective [11].

In this chapter, AI is approached from a computer science and information technology perspective, encompassing numerous technologies and frameworks, and focusing largely on embedded hardware/software systems that use searching algorithms, logic-based procedures or ML methods. ML represents a paradigm shift in computing - a change from explicitly modelling solutions to modelling systems that approach such solutions, which drives one to think in a new framework. ML - both software and hardware - is therefore addressed in several sections.

1.1.1 Challenges of Industrial AI versus Consumer AI

Although industry stakeholders have different perceptions of AI technologies and their industrial applications, industrial AI poses unique challenges that are absent from consumer AI or are present but of less importance or differ from the challenges related to the latter. Some of these challenges are described below.

Industrial training data are in short supply. AI-based models require large amounts of data, and their performance relies strongly on training data set availability. These data sets exhibit tremendous potential for optimising industrial processes in cases in which traditional approaches, such as stochastics and analytical or numerical models, can no longer be used. However, for many industrial sectors, it is not easy to create training data sets that are sufficiently large and cover common aspects that would allow them to be used by different industrial stakeholders to benchmark similar AI models.

Industrial training data are often noisy or inaccurate. While data coming from consumers are hard to misinterpret, this is not the case with industrial data, which is frequently captured from sensors and IoT/IIoT that produce noisy datasets. Sensor data can also be voluminous, and not all data is relevant. Data can also be inaccurate when generated by "digital twins" models that are not always created and maintained in tandem with the real system. Furthermore, the actual deployment of sensors close to production environments that are generally ungracious of the sensors (higher likelihood of sensor malfunction) and redundancy to alleviate this problem introduces additional challenges and costs. Nevertheless, despite the high volume of noisy, incomplete, or faulty data, industrial AI needs to be highly accurate.

Industrial AI runs mostly on the edge. Consumer data are processed on computers with seemingly infinite capacities, and current AI tools are optimised for cloud services and therefore do not always fulfil the stringent requirements of industrial applications, such as real-time processing, low latency, high reliability, safety, data privacy and guaranteed QoS. To be successfully implemented in industry, AI must be deployed on the edge to support distributed on-site data processing with state-of-the-art AI components, algorithms, techniques, and methods.

Industrial AI can be subject to compliance with industry standards and other regulations. While consumer AI can at most be subject to direct consumer scrutiny, industrial AI is subject to compliance requirements, including technical, legal and corporate requirements, as well as local and governmental regulations, which may impact operations, particularly when large budgets are at stake.

Industrial AI involves high costs. The development, implementation, deployment, repair, and maintenance of AI-based solutions necessitate vast investments. AI-based systems require frequent upgrades to meet the needs of changing environments and to make machines more intelligent day by day. In severe breakdowns, the recovery of lost codes and the restoration of AI-based systems may require considerable amounts of time and costs. Maintenance of the sensor part of the AI solution also contributes to overall costs.

Industrial AI must be explainable. Industrial AI applications must be able to explain and justify their predictions and decisions, especially when the consequences of wrong decisions can be disastrous.

Industrial AI systems are difficult to validate and test due to the costs and complexity involved. The complexity of AI tasks has increased steadily to address new paradigms for automating, conceptualising, designing, and implementing AI-based systems that include sensors, hardware, software, models, and algorithms. In many cases, industrial AI systems are trained and tested using simulations and virtual validations.

1.1.2 Sustainable AI

The search for improved accuracy on large-scale problems is driving the use of new AI techniques and increasingly deeper neural networks, thereby increasing energy consumption and climate-changing carbon emissions.

Advances in scientific computing have demonstrated the advantages of modelling and simulation across industrial and scientific domains. However, energy consumption is a feasibility constraint for computational modelling, and AI must reduce the energy computation costs associated with high-performance computing in front of trends such as declining Moore's Law.

The evolution and expansion of AI-based technologies require moving towards sustainable AI and using AI for sustainability in various industrial sectors. To build and strengthen sustainable AI technologies and applications, new solutions need to be developed to move AI processing from the cloud to the edge, optimise and reduce the need for datasets and amount of data for training and learning and address the analytics close to the data sources.

Sustainable AI (or AI sustainability) requires stimulating change in the entire lifecycle of AI technologies and applications (e.g., AI function generation, AI technology stack, HW/SW platforms, training/learning, re-tuning, re-training/learning implementation, governance) towards more efficient ecological integrity and economic efficiency. Sustainable AI technologies are compatible with sustainable environmental resources for current and future generations, and new digital economic models are aligned with industrial and societal values.

Research and developments in industrial edge AI incorporate two essential elements: sustainable edge AI (e.g., edge AI technologies development for optimised resource processing consumption, resource consumption for AI models, reduction of carbon emissions, computing power, etc.) and edge AI for sustainability (e.g., the use of edge AI to address sustainability goals in different applications and industrial sectors). These elements can be viewed from the perspective of the different pillars of sustainability (e.g., social, economic, and environmental).

Sustainable industrial edge AI focuses on developing AI HW/SW/ algorithms and resource-efficient edge AI technologies to reduce carbon emissions and computing power consumption of AI models.

Industrial edge AI for sustainability and sustainable computing leverages intelligent processing technologies to address environmental and climate problems and ameliorate the accelerating trend towards high-performance computing in modelling and simulation.

Leveraging hardware modules and platform characteristics to generate compact and accurate models that require less computational resources is essential for sustainable edge AI. Combining different techniques, including knowledge distillation, AI HW/SW co-optimisation for power efficiency and energy-aware model compression, can result in models with negligible loss of accuracy.

Sustainable edge AI implies less data for model training to achieve high-accuracy model performance, thereby reducing the expensive data collection and annotations, accelerating model training when faced with a new problem and reducing the resource-intensive process of training a new model from scratch.

The development of semi-supervised methods [16] by incorporating external knowledge, active learning, transfer learning and short learning approaches, such as meta-learning and unsupervised representation learning, are elements of the AI technology stack that supports sustainable edge AI developments. These methods facilitate domain adaptation across problems, including natural language processing and predictive maintenance in different industrial sectors.

Sustainable edge AI requires enabling more accurate modelling techniques, reducing computing costs by reducing time-to-solution, decreasing the need for high-resolution models where possible and leveraging resource/data-efficient AI developments to ensure that the application of AI is not energy/resource-consuming.

Sustainable edge AI requirements advance the development of new embedded hardware modules, platforms, and accelerator architectures (e.g., system on module (SoM), system on chip (SoC), a system in package (SiP), neuromorphic, hybrid, tensor-based, etc.).

This chapter relies on a framework of classifications and continuums along various dimensions, including the industrial intelligence spectrum, the intelligent capabilities spectrum, the edge granularity, the edge continuum, the symbolic reasoning-pattern recognition continuum and not the least, the problem-solving spectrum.

The foundation supports the ongoing projects and stakeholders across the industrial sectors with a common methodology and roadmap. Meanwhile, it prioritizes the right features for AI-based applications and designing them in the right way in different use cases across various industrial sectors using synergies among different solutions, methods, or techniques.

The foundation assists in choosing state-of-the-art AI technologies and having a clear overview over the existing state of play in the field for optimal selection and trade-off of these technologies, methods, and techniques for use cases in different industrial sectors.

AI empowers computers to mimic human intelligence, such as decision-making, text processing and visual perception. In this context, AI is a broad field, encompassed by multiple contributing branches, such as ML, robotics, and computer vision.

1.2 Capabilities Spectrum of Industrial AI

This chapter relies on a framework of classifications and continuums along various dimensions, including the industrial intelligence spectrum, the intelligent capabilities spectrum, the edge granularity, the edge continuum, the symbolic reasoning-pattern recognition continuum and not the least, the problem-solving spectrum.

The foundation supports the ongoing projects and stakeholders across the industrial sectors with a common methodology and roadmap. Meanwhile, it prioritizes the right features for AI-based applications and designing them in the right way in different use cases across various industrial sectors using synergies among different solutions, methods, or techniques.

The foundation assists in choosing state-of-the-art AI technologies and having a clear overview over the existing state of play in the field for optimal selection and trade-off of these technologies, methods, and techniques for use cases in different industrial sectors.

AI empowers computers to mimic human intelligence, such as decision-making, text processing and visual perception. In this context, AI is a broad field, encompassed by multiple contributing branches, such as ML, robotics and computer vision.

AI can be understood in the context of the tasks that we expect an intelligent machine, IoT/IIoT device to be capable of performing.

An intelligent machine, IoT/IIoT device is any system whose behaviour could be interpreted as reflecting human intelligence, which may be demonstrated in basic capabilities, such as perceiving, comprehending, acting, and

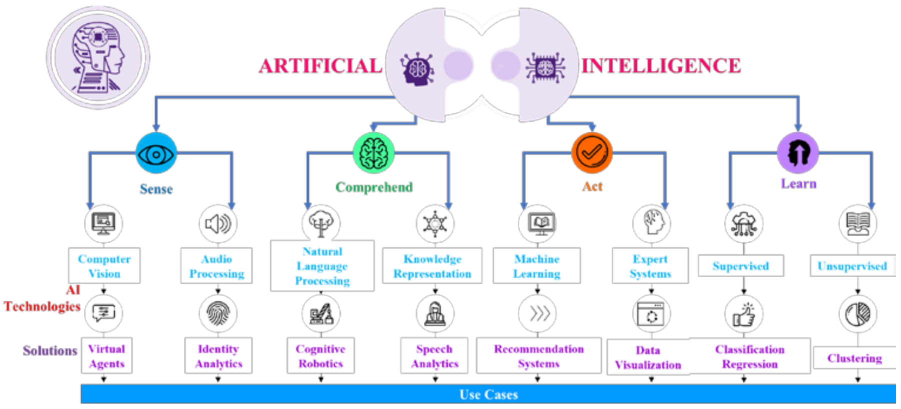

Figure 1.1 AI systems capabilities.

learning. For example, the three-dimensional classification scheme for evaluating an AI-based systems in [37] differentiates four capabilities: perception, understanding, action and communication.

In the context of the European projects contributing to this book, four capabilities are differentiated as shown in Figure 1.1, but the list is extended with more capabilities, which are elaborated below from the perspective of their application to industry.

The **perceiving** or **sensing** capability allows industrial machines, IoT/IIoT devices to scan their environment using various sense organs, either artificial or real, and to collect and process data streams (images, sounds, speech, text, and other data) from diverse sources, such as radar, lidar, cameras, ultrasound sensors, etc.

The processing is often complicated, as it involves great numbers of distinct appearances over multiple occasions, varying by view and angle, as well as scenes suggesting objects that may be hidden. Mechanisms, such as data, information, and sensor fusion are employed to assimilate various sources of information, often imperfect and uncertain, and deal with multiple dimensions of remote sensing (spatial, temporal, spectral, and radiometric resolution).

The **comprehending** capability enables industrial machines, IoT/IIoT devices to recognise patterns and context in the information it collects, just as humans interpret data/information by understanding patterns and context in their perceptions of the environment, but it is important to note that comprehending does not have the same meaning for machines as for humans.

Machines do not actually "comprehend" the world around them; rather, they are trained to "learn" how to recognise patterns.

An industrial machine's and IoT/IIoT device's **learning** capability enables it to continually improve its performance by learning from the success or failure of its actions. Like humans, machines learn in various ways, for example, by trial and error. A machine tries various solutions to a problem until it achieves the desired results. It records all the steps actions that produced those results in its memory for use the next time it is given the same problem.

The **reasoning** capability enables industrial machines and IoT/IIoT devices to draw relevant inferences from the situation at hand. Reasoning has become an essential component of AI only in the past decades before which the ability was limited to humans.

Logic employs two broad methods of reasoning: the deductive and inductive approaches. Deductive reasoning works from the "top down", moving from a theory to its confirmation (or rejection) by collecting observations to address the hypotheses and narrowing down the possibilities.

By contrast, inductive reasoning works from the "bottom up", moving from specific observations to broader generalisations and theories by detecting patterns and then formulating testable hypotheses.

The **problem-solving** capability enables industrial machines and IoT/IIoT devices to move from a problem's initial state to the final goal state through a stepwise gradual reduction of the difference between the current state and any intermediate goal state.

This involves using several techniques, such as algorithms and heuristics, to solve a problem. The ability to solve problems, a highly prized skill in both humans and machines, involves two distinct, possibly conflicting processes: creativity and decision making. The former, creativity, generates options and possible solutions, and then the latter, decision making, selects the optimal one.

The **acting** capability enables industrial machines and IoT/IIoT devices to act (inspired by their perception or comprehension) in the physical or digital environment. It is implicitly assumed that machines will act rationally and determine the best and safest course of action for achieving their goals.

The **interacting** capability enables industrial machines and IoT/IIoT devices to connect to the environment and to everything and collaborate with humans, other machines, and infrastructure (physical and digital, edge/cloud, etc).

This emerges primarily when industrial machines have to interact with people, which assumes the ability to understand language. For example, when an AI system explains how it came to its decision, it must adopt the normal conventions of human interaction to make itself understood.

The **locating** capability enables industrial machines and IoT/IIoT devices to determine (relative) positions very precisely and accurately on network, dynamic maps, GPS, GNSS, etc., that help in identifying the context of actions.

1.3 The Industrial AI Spectrum

In the previous section, a classification of AI was given in the context of the tasks that we expect an intelligent industrial machine to be capable of performing.

Additionally, AI can be understood from the perspective of the (theoretical) ability of an intelligent machine situated on a continuum, from specific to general intelligence or from basic to super intelligence. Some forms of AI within this continuum can be distinguished by names, such as Narrow AI, General AI, Weak AI, Strong AI [35].

These various forms of AI differ primarily in their range of abilities/capabilities and the level of training required to implement them. In the following, they are described in contrasting pairs from the perspective of their relevance to industrial applications.

1.3.1 Narrow AI vs. General AI

General AI defines an AI system that parallels human intelligence. As such, it is considered an ultimate vision of AI that can handle a wide variety of cognitive tasks across multiple domains.

General AI is the basis for future human-like autonomous systems and robots, which will implement hundreds of systems working in parallel while communicating with each other in a manner that mimics human reasoning. While the development of technology pushes the abilities of AI ever closer to General AI, most AI surrounding us today is Narrow AI.

Broadly speaking, Narrow AI can be thought of as anything that is not General AI. Narrow AI defines an AI system capable of performing a particular task that any human would ordinarily perform. Narrow AI systems are designed to precisely execute a well-defined task. These systems are optimised to excel in controlled environments with a limited set of parameters,

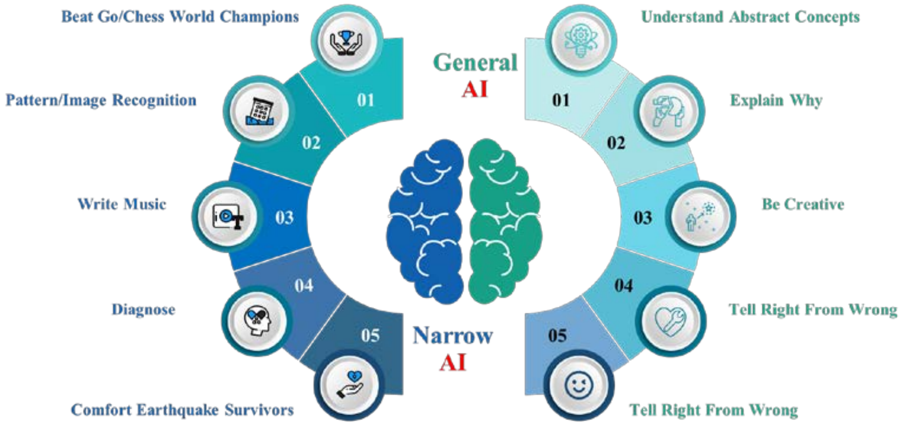

Figure 1.2 Narrow AI vs General AI.

demonstrating capabilities that match, or even surpass, those of a human. However, their capabilities are narrow (e.g., e-commerce suggestions based on user search patterns, weather prediction, predictive maintenance, etc.), and they cannot do anything that is not explicitly stated in their programming.

A comparison of the features of Narrow AI and General AI is illustrated in Figure 1.2. The primary difference between Narrow and General AI comes down to adaptability. For AI to be generally intelligent, it must be able to adapt rapidly to changing surroundings in the same way that humans do. In practice, this would mean to pass the Turing test repeatably and consistently. Turing defined intelligent behaviour as the ability to achieve human-level performance in all cognitive tasks sufficient to fool an interrogator.

1.3.2 Weak AI vs. Strong AI

The terms Weak AI and Strong AI are sometimes used in place for Narrow AI and General AI, respectively. Weak and Strong AI were coined in [9] to differentiate the performance levels in different kinds of AI machines.

Strong AI systems (e.g., advanced robotics, intelligent robotic things, etc.) behave intelligently, think as humans do, and have a conscious, subjective mind; they know who the AI systems are, what they are doing, and why. Strong AI systems are represented by an AI-based application with a larger scope, using high-level clustering and association to process data, information, and knowledge.

In contrast, Weak AI defines the simulated thinking of the brain processes with the help of a computer. It behaves intelligently (e.g., chatbots, Siri, Alexa, etc.) but does not exhibit any kind of consciousness about what it is doing.

Weak AI systems are represented by Narrow AI-based applications with a limited scope that are optimised by using supervised and unsupervised learning to process data collected from different sources (e.g., real-time or from databases, etc.).

Although Weak AI systems never attain the breadth of a General AI, most Narrow AI systems are very powerful and focused. It is therefore important to not conflate Narrow AI, which deals with specific tasks, with weak AI.

1.3.3 Basic AI vs. Super AI

The term Super AI refers to the combination of General AI and Strong AI at the point at which it surpasses the intelligence and ability of the human brain. This is made possible primarily due to the amount of memory and instantaneous access to data, which far exceeds human limits. In addition, this AI will improve self-capabilities to feel things and emotions.

Nevertheless, since Strong AI is still theoretical, the realization of Super AI lays far in the future, relying strongly on technological advancements in hardware (quantum computing), software, and other fields (biomimicry).

Basic AI, in contrast, can be considered for any AI that is under the threshold of Super AI. It is an all-encompassing term that denotes the simplest tasks and technologies used and is mentioned here merely as a foil to Super AI.

1.3.4 Red AI vs. Green AI

The term Green AI [12] defines AI research that yields novel technological results while considering the financial cost of developing, training, and operating, as well as encouraging a positive impact both on the environment and inclusiveness.

Green AI includes the optimisation of the use of data/information, the processing at the deep edge/edge/cloud, the transfer and exchange of data/information, and storage.

The term Red AI defines AI research that seeks to achieve progress regardless of the huge computational power required and environmentally unfriendly impact involved.

While Red AI research has made valuable scientific contributions to the field, making AI both greener and more inclusive will lead to wider acceptance of AI in industry.

Nevertheless, ensuring a smooth transition from Red AI to Green AI is not straightforward. For instance, the type of energy sources used for powering the data centres, edge computing facilities, or the intelligent devices at the edge is part of the efficiency equation associated with training/learning/reasoning algorithms. Even if powered by renewable sources, massive power consumption for stronger results from the algorithms may not be considered an improvement step towards Green AI.

To conclude the discussion on the industrial AI spectrum, the degree to which more General, Strong, Super and Green AI can be achieved will largely depend on the abilities of the particular AI system to continuously learn how to solve problems from multiple application domains without requiring extensive retraining for each, to learn in a self-supervised manner, and to adapt the knowledge and skills acquired to new situations with little-to-no training.

1.4 AI Problem Solving Domains

Problem-solving is a method used to reach a desired goal or find a solution to a problem. In the context of computer science, problem-solving refers to various techniques, such as forming efficient algorithms and heuristics, to find desirable solutions. A single problem can have many different solutions, and these can be achieved by different methods. Also, some problems have unique solutions, depending on the nature of the given problem and the domain.

AI has always been beneficial for solving complex problems and challenges that cannot be solved by other means. This section presents some of the major AI problem solving domains that are most used in industrial problem-solving and/or that have great potential for sustainable developments. The various branches of AI and AI problem-solving domains are illustrated in Figure 1.3. For a more complete overview of problem-solving techniques, we refer the interested reader to [4][8].

1.4.1 Expert Systems

Expert systems (ES) are computer programs designed to act as experts in a particular domain or area of expertise. In other words, they are designed to model human expertise in that specific knowledge area. Problem-solving relies on organising considerable amounts of knowledge and then systematically searching through them when selecting the path to go with each decision, ultimately leading to a solution. A typical architecture is shown in Figure 1.4.

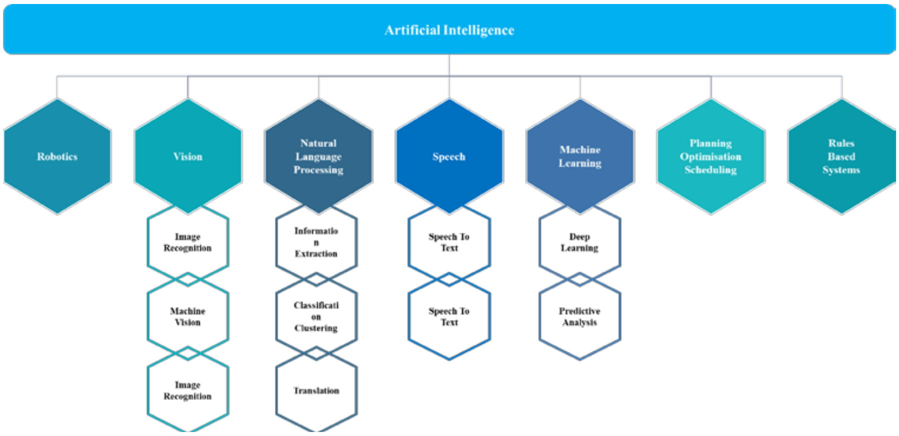

Figure 1.3 AI problem solving domains.

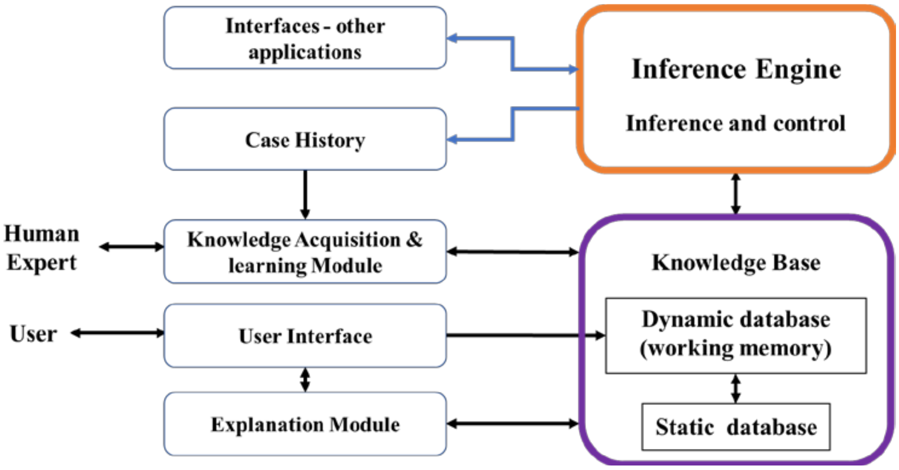

Figure 1.4 Typical expert system architecture.

There are two basic components in an expert system: a knowledge database and an inference engine. The knowledge base mainly consists of facts about that domain (declarative knowledge) and rules for applications to those facts (procedural knowledge). The most common representation of human expert knowledge is in the form of rules, for example, an ‘if A, then B’ structure.

The inference engine processes the input information (for example, that A is true) and draws the deductions based on the rules (for example, B). It consists of algorithms, which, via step-by-step inferences, draw deductions based on the knowledge rules. Depending on the application, ES may also have a user interface to interact with users.

In the absence of generalised knowledge-based systems, the industry embraced the idea of practical ES for specific tasks, and there are many successful applications of ES in medicine, agriculture, and other areas, where ES assist or even replace human experts with specialised knowledge. ES remain important tools for decision support or decision-making; nevertheless, they have evolved in both the technological and business directions. ES can now be embedded into applications and can be designed to handle uncertainty. Furthermore, new knowledge representation and reasoning tools have been developed for ES: MYCIN [13] for disease diagnosis, DENDRAL [30] for chemical analysis to predict molecular structure, R1 for configuring orders for new computer systems [14], Fuzzy Logic UAV (Unmanned Aerial Vehicle) Motion Planning [31], etc. There are other application areas such as environment, manufacturing, diagnostic tools for vehicles, and machinery.

ES remain a feasible solution when there is a lot of human expert knowledge and experience that can be modelled but not enough data to build other problem-solving systems. ES can also be the preferred solution because of their unique capability, namely explainability, which other systems lack in spite of advanced problem-solving capabilities. This could be an obstacle in some application areas, such as autonomous vehicles, where unexpected decisions need to be understood.

On the downside, knowledge bases take time to acquire and represent on computers, and if some knowledge is missing or incomplete, a less reliable result will be produced. Hence, verification and validation methods and techniques aimed at ensuring quality are fundamentally important.

Initially, ES were built around rules established by humans, but gradually the rules are being set by computers, which can interpret and extrapolate from large volumes of data. In this respect, the AI learning process can be implemented using top-down approaches (e.g., expert systems) or bottom-up approaches (e.g., machine learning).

ES and its technology have been one of the most important and widely used parts of AI and goes back to the beginning of AI, so they have been used in business for decades. This is an area that will continue to be important in the future, either independently or in combination with other major branches of AI.

As a concluding remark, there is a lot to learn from the earlier generation of AI in our pursuit of the development of explainable and verifiable AI.

1.4.2 Machine Vision

Machine vision (MV) is a branch of AI that enables machines to imitate the human visual system and perform various tasks, such as image classification and segmentation, object detection and recognition, and object tracking, using information collected from various sources including IIoT image sensors. MV enables intelligent vision devices to grasp their visual surroundings and to process, analyse and understand digital images. For example, in the case of autonomous vehicles, MV detects traffic signs, buildings, vehicles, pedestrians and other participants in the traffic.

Machine vision and computer vision (CV) are sometimes used interchangeably, but they are different. Both are used for image processing, so they both use similar components, such as cameras, IIoT image sensors to capture images and software to handle the data. However, CV uses systems with PC-based processors to analyse the imaging data it collects, so it has a lot of processing power and is commonly applied in the medical, financial and security industries. CV can be used alone, without needing to be part of a larger machine system.

MV, by contrast, is integrated into perception systems in industrial sectors, such as autonomous vehicles, manufacturing, food processing and semiconductors. An MV system uses algorithms to process and interpret an image, and it instructs other components in the system to act upon that data. MV systems are therefore designed to quickly analyse image data and make simple, automated decisions on different tasks, such as quality control, inspection, and guidance. The image could be obtained from a thermal or infrared sensor, IIoT image sensors, motion detectors or other sources.

Analysis of reams of images produced by sensors requires that the machines be able to see and understand images, and this is where AI comes into the picture because its methods and techniques permit the automatic extraction of information from images.

Machine vision is one of the areas that has greatly benefitted from the rapid advances in AI/ML, and implementation of MV's capabilities is now possible at all micro, deep and meta edge levels. Modern MV systems are usually built using different types of neural networks, including DL. DL allows machines, robots and intelligent IIoT devices to recognise objects with close to human-like ability. At the lower levels, ML algorithms perform processing techniques on the image, extract features from the image and access and intertwine multiple views. At the higher levels, they perform more advanced tasks, such as image classification, and they make inferences about whether the object in the image belongs to a specific class of objects.

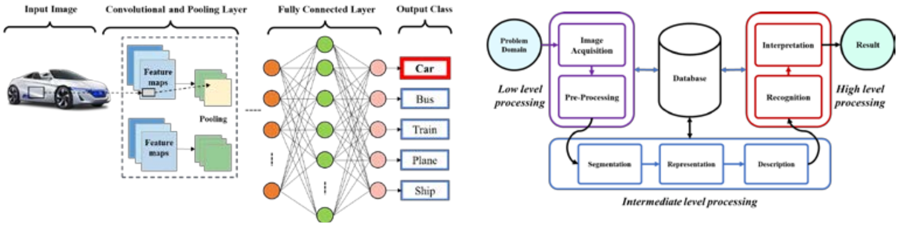

Figure 1.5 Typical CNN-based machine (left) and workflow (right).

The highest level is where DL is employed to build intelligent, scalable MV systems that can recognise/identify and react/respond to objects in images and videos.

From the multitude of neural network architectures, Convolutional Neural Networks (CNNs) have become increasingly powerful in large-scale image recognition by combining the feature extraction processes and classifying the extracted features in the same algorithm. When DL technology is deployed in IIoT devices, it relies on pretrained DL models, and transfer-learning techniques are employed to retrain an existing image classifier into a custom classifier by retraining a small image dataset using minimal resources. An intuitive illustration of CNN-based MV is shown in Figure 1.5

Nonetheless, some challenges arise when deploying MV on IIoT edge devices. Most deep NNs are too complex to be created and trained on most present-day microcontrollers, but if optimised in terms of memory, processing, and power capabilities, they can run on them. The optimisation can be done either by rewriting the models in low-level languages or by quantising to improve the latency and the model size.

Real-time object detection on edge devices with live video analytics using YOLO (You Only Look Once) are widely used for video surveillance and are important for mobile robots, including self-driving vehicles.

The machine vision system uses embedded edge AI for use-case applicability and autonomous optimisation in industrial manufacturing visual inspection and are extensively used in various industrial applications and sectors.

1.4.3 Robotics

Robotics is a branch of AI that deals with creating machines that can perform some actions like humans. AI capabilities enable robots to act intelligently in certain situations by solving problems in a limited sphere or even learning in controlled environments. Many industries are implementing robotics solutions to overcome critical issues related to production and execution by eliminating the potential for human error while reducing redundancy in manual labour.

In recent years, there has been consistent progress in intelligent robotics, driven by an increased availability of complex and intelligent sensor systems, powerful computing and communication capabilities, and software platforms. Progress in deep learning in particular is opening up new opportunities in industrial robotics – leveraging improvements in machine vision, robotic grippers that can pick up randomly placed objects and stack them, and other agile and dynamic robotics systems that operate at speeds essential for many industrial applications. Thus, implementing inference at the edge, without connecting to the Internet, enables robots to make decisions independently.

The greatest impact has been on automobile industry and the use of autonomous vehicles. The design of self-driving vehicles requires the integration of technologies such as sensor fusion, AI decision-making, vehicle-dynamics prediction, on-the-fly rerouting, and inter-vehicle communication to carry out tasks such as adaptive cruise control, to safely adjust speed, and lane-keeping assistance, to keep vehicles centred on the road. A schematic illustration of an end-to-end deep learning for self-driving vehicles is shown in Figure 1.6.

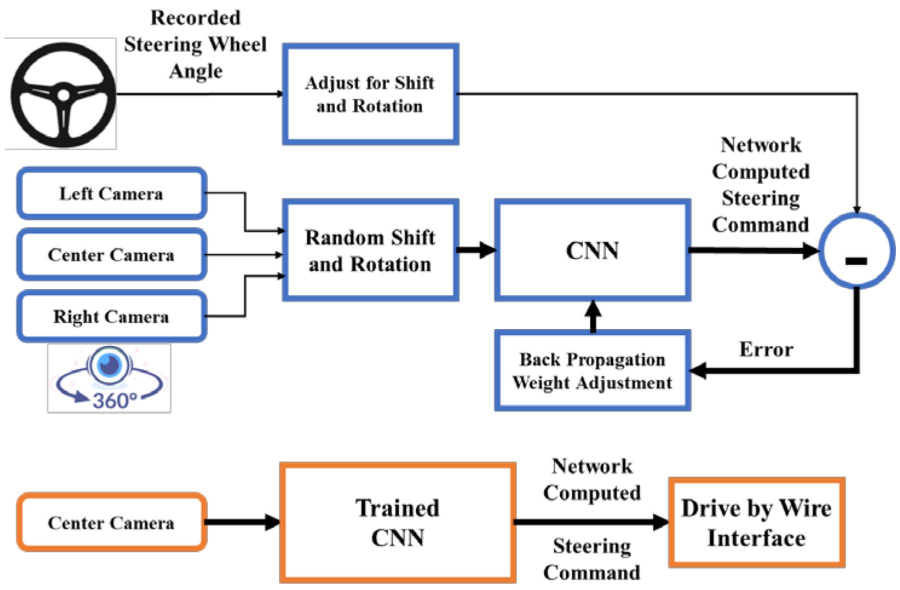
Recorded Steering Wheel Angle
Adjust for Shift and Rotation
Left Camera
Center Camera
Right Camera
360°
Random Shift and Rotation
CNN
Network Computed Steering Command
-
Error
Back Propagation Weight Adjustment
Center Camera
Trained CNN
Network Computed Steering Command
Drive by Wire Interface

Figure 1.6 Self-driving vehicles: Training and inference (generate steering commands). Adapted from [33].

Training data contains single images sampled from video, paired with the corresponding steering command. Training with data from the human driver is insufficient. The network must likewise learn how to recover from any mistakes, or the vehicle can slowly drift off the road. In this case, the training data is augmented with additional images that show the vehicle in different shifts from the centre of the lane and rotations from the direction of the road. The images for two specific off-centre changes can be obtained from the right and left cameras [33].

Images are fed into a CNN which computes a proposed steering command. The proposed command is compared to the desired command for that image, and the weights of the CNN are adjusted to bring the CNN output closer to the desired output. During inference, the trained model generates steering commands from the input video images.

1.4.4 Biomimicry

The term AI typically connotes emulating, mimicking, or replicating human intelligence in machines. However, AI also encompasses biological intelligence, including that of plants, animals, and other living organisms. Plants, for instance, do not possess brains, but they have senses. Hence, one of the many lessons to be learned from billions of years of evolution, natural engineering, and natural design is that embedding more processing power close to the sensors and actuators will improve intelligent functioning in IIoT technologies in different industrial applications.

There are many examples of AI problem-solving architectures and techniques that incorporate insight from nature into their solutions, e.g., machine learning, robotic vision, and path-planning. Biomimicry is an approach to problem-solving that produces innovative sustainable solutions by learning from and replicating the natural patterns observed in living systems and beings (e.g., plants, animals, humans) to create remarkably intelligent technologies and products. These patterns, which appear in nature with varying degrees of frequency, can be found not only in forms and shapes, but also in processes (chemical, physical, and behavioural) and ecosystems. Highly fundamental patterns observed across species that appear very frequently are known as Life's principles [25].

Life's Principles are lessons from nature and can serve as inspirational lessons for designers and developers and can be applied to various industrial applications. Being locally attuned and responsive is one of the six primary Life's Principles, and it refers to how an organism fits into and integrates with

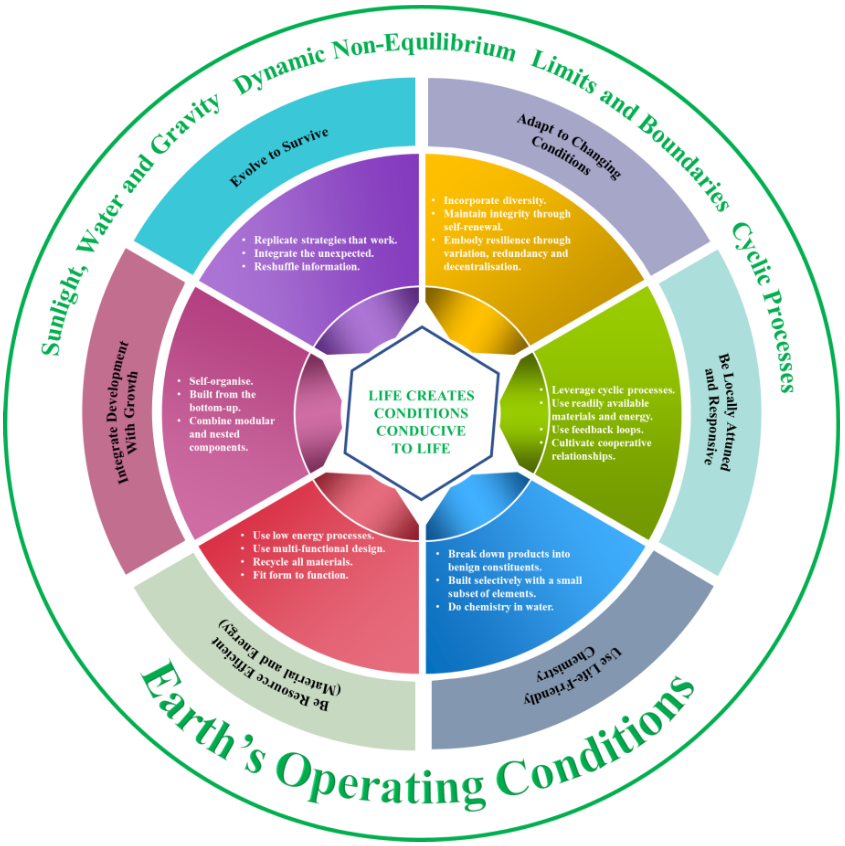
Dynamic Non-Equilibrium
Limits and Boundaries
Cyclic Processes
Sunlight, Water and Gravity
Earth's Operating Conditions
Evolve to Survive
Replicate strategies that work.
Integrate the unexpected.
Reshuffle information.
Adapt to Changing Conditions
Incorporate diversity.
Maintain integrity through self-renewal.
Embody resilience through variation, redundancy and decentralisation.
Be Locally Attuned and Responsive
Leverage cyclic processes.
Use readily available materials and energy.
Use feedback loops.
Cultivate cooperative relationships.
Use Life-Friendly Chemistry
Break down products into benign constituents.
Built selectively with a small subset of elements.
Do chemistry in water.
Be Resource Efficient (Material and Energy)
Use low energy processes.
Use multi-functional design.
Recycle all materials.
Fit form to function.
Integrate Development With Growth
Self-organise.
Built from the bottom-up.
Combine modular and nested components.
LIFE CREATES CONDITIONS CONDUCIVE TO LIFE

Figure 1.7 Life's principles.

Adopted from [25].

it surrounding environment. The use of feedback loops is a sub-principle or strategy for being locally attuned and responsive. Feedback loops are cyclical information flows that allow organisms to adequately modify their reactions to environmental stimuli and situations.

We can find hundreds of unique strategies and mechanisms in nature for sending and receiving signals in a feedback loop. The white clover is a good example of the information flow that occurs between a prey and the formation to which it belongs, finalised by feedback to the predator. To survive, this plant employs a chemical defence mechanism to ward off herbivores. When white clover leaves are damaged (chewed), two chemicals mix to form hydrogen cyanide, a bitter substance that makes the leaves less tasty.

For an AI system, feedback loops are essential because they enable intelligence. A feedback loop entails the assessing and leveraging of its output (predictions or recommendations) to retrain and improve the model over time. Feedback loops are used in machine learning and deep learning, especially in neural networks. A good example is the object recognition technology in self-driving vehicles and its ability to recognise traffic lights, road signs, pedestrians, automobiles, and various types of objects, with feedback loops improving its accuracy. Through feedback from the output layer in a neural network model, the variations of weights in the hidden layer(s) are adjusted to fit the expected outputs. Positive feedback increases the change or output, while negative feedback decreases the change or output.

Returning to the white clover example, when a white clover leaf is attacked, this action triggers signals in every direction, making the other leaves harder to chew and upgrading the mechanical and chemical resistance of the entire formation. This is made possible by its network infrastructure composed of runners—commonly found in many plant species.

Runners are stems growing just at or below the soil surface. They form roots at the nodes, new plants grow from their buds, and they are part of a propagation strategy. Above ground, these plants most often appear to be distinct individuals, but underground, they are interconnected, such that when one member of the formation senses something, a signal is sent to every other member, facilitating a quick response to a predator.

Feedback loops also occur in ecosystems, with connections within the formation allowing for rapid information circulation, information processing, and reaction. To function like ecosystems, AI systems must be strongly interconnected and equipped with built-in IIoT technologies that continually capture stimuli from the environment. These stimuli are then converted into information that is circulated and processed rapidly, resulting in an almost instantaneous self-regulation and adaption to any change, along with feedback on the origin of the change.

AI systems functioning like ecosystems will foster collaborative infrastructure design and sustain innovation, enabling these systems to evolve and rapidly learn how to evolve. AI systems with *collaborating sensors* reminiscent of such collaborative infrastructure would behave almost organically.

1.4.5 Genetic and Evolutionary Algorithms

Genetic algorithms (GAs) represent a branch of AI searching for a range of potential solutions to find one which solves a particular problem. GAs save information about the paths traversed during the search, simulating an evolutionary process and in this way overcoming known issues such as inefficient searches, and convergence to local optimums rather than global optima.

The idea of GAs can be traced back to Alan Turing's paper from 1950 [26], where concepts derived from natural evolution are used to evolve AI machines. These include mutation, hereditary material, survival of the fittest, and keeping track of the different genetical combinations that have been tried and tested, to avoid trying the same ones again.

GAs are a subset of a much larger branch of computation known as evolutionary computation. The main concept behind GAs and evolutionary algorithms (EAs) is inspired by the natural selection principle in biological evolution [27], in which organisms evolve and adapt to thrive in the surrounding environmental conditions.

According to this principle, new candidates can be produced from a current population of individuals using crossover and mutation, which perform different roles. Mutation is a divergence technique, driving the population to discover new regions and enlarge the search space. Crossover is a convergence technique, driving the population towards a local optimum. The fitness of individuals is evaluated against a fitness function related to the optimisation problem being solved, subsequently the stronger candidates are selected to breed, the rest are 'discarded'. Since the ultimate 'goal' is to bring the population to a state of convergence, selection/crossover occur more frequently than mutation.

This process is iterative, in that the new generation of candidate solutions becomes the current population in the next iteration. The cycle terminates after the maximum number of iterations has been executed, or earlier if the fitness functions reach a satisfactory level. The advantages of GAs include their relatively simple application to new problems – merely requiring redefinition of the fitness function, and they are also effective and scalable, due to the "survival of the fittest" principle, according to which the unfit candidates are eliminated during the process.

GAs and EAs have a wide range of applications, such as in robotics, evolutionary machine learning, generative design applications and evolvable hardware. For example, GAs can accelerate the NN learning process to solve a certain problem, by learning the best hyper-parameters. This is illustrated in Figure 1.8.

Evolvable hardware (EH) is another field focusing on the use of EAs and GAs to create specialised hardware and electronics without manual

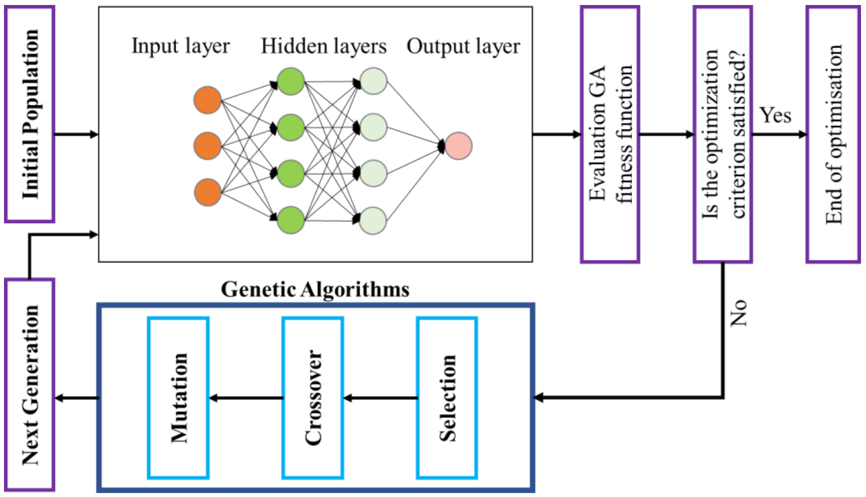

Figure 1.8 Using Genetic Algorithms in the iterative process of fine-tuning NN hyperparameters.

engineering. Although it started out as a branch of electrical engineering and computer science, EH now brings together reconfigurable hardware, evolutionary computation, fault tolerance, sensors; connectivity and processing modules; and autonomous systems. In a broader sense, EH refers to any form of hardware that can change its architecture and behaviour dynamically and autonomously by interacting with its environment.

Regardless of the industry, the generation and testing of different solutions is a critical part of every design process, including generative design. The generative design algorithm creates and tests different configurations, diverging to explore a large variety of solutions based on the pre-set requirements, and then converging on the best solution. Often such processes are cyclical/iterative, where initial requirements are adjusted, leading to another cycle/iteration of generating candidate solutions. Such processes can become very complex, hence the need for AI systems, such as GAs. Thus, one of the most powerful benefits of generative design is the speed with which new candidates can be generated and evaluated because the entire cycle is automated.

1.4.6 Generative AI

Generative AI is a branch of AI that enable computers to learn underlying patterns related to their input, which can be text, audio files or images, and then use these to create similar content [23]. While advances in ML have mostly been the result of discriminative modelling, the most significant advances in AI in recent years have been attributed to generative modelling, not least due to its ability to create new things.

In contrast to discriminative techniques that learn to classify data, generative AI techniques are mostly involved in creating new data based on training data. Discriminative modelling is focussed on learning a function that maps an input to an output using a labelled dataset, a notion synonymous with supervised learning. Generative modelling is usually performed with an unlabelled dataset, that is, as a form of unsupervised learning.

The differences between discriminative and generative modelling are best visualised in Figure 1.9. Discriminative models draw boundaries in the data space, focusing on predicting the data labels, while generative models try to model how data are placed throughout the space, focussing on explaining how the data were generated.

There are three main classes of generative AI techniques: general adversarial networks (GANs), autoregressive convolutional neural networks (AR-CNN), and transformer-based models.

GANs are a breakthrough, empowering deep networks with the ability to produce artificial content that passes for the real thing. GANs consist of two competing components – the generator network, which learns the distribution of classes, and the discriminator network, which learns the boundaries between those classes. Each network can be any type of neural network, such as artificial neural network (ANN). The discriminator must have fully connected layers with a classifier at the end.

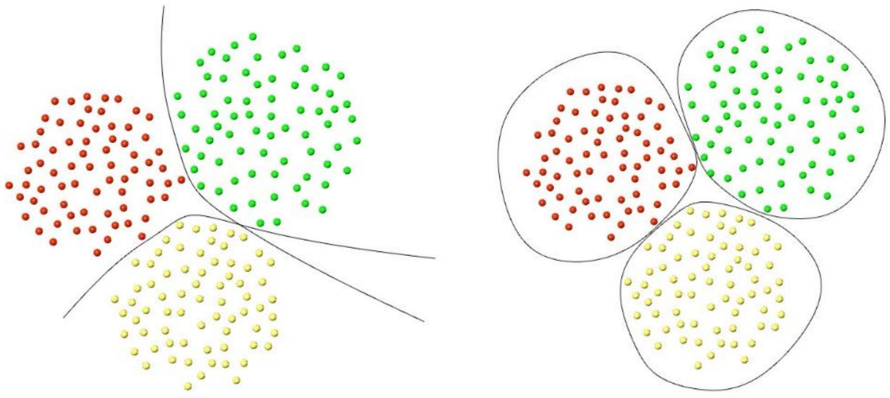

Figure 1.9 Discriminative (left) vs Generative (right) Models in ML.

GANs are the cutting-edge technology of AI, not least due to two essential key advantages: they solve the problem of generating data when there is not enough to start with, and they require no human supervision.

One of the practical applications of GANs can be seen in anomaly detection. Anomaly detection is known for identifying signal behaviours that do not fit the normal patterns and which can be addressed as a supervised learning problem. Depending on the application, this may require large, labelled datasets. However, in many industrial applications, samples from abnormal class may be insufficient for effective modelling. This is a challenge that can be addressed by another approach, using GANs (i.e., training only on samples considered 'normal' and then identifying the unusual, insufficiently available samples (abnormal) that differ from the learned sample distribution of normal).

An example of the network architecture of the generator and discriminator based on deep convolutional GAN is shown in Figure 1.10. In the training stage, only normal samples are involved. In the testing stage, abnormal samples can be discriminated by a higher anomaly score. The generator is trained only using the extracted features from normal samples. Anomaly scores are designed for anomaly detection.

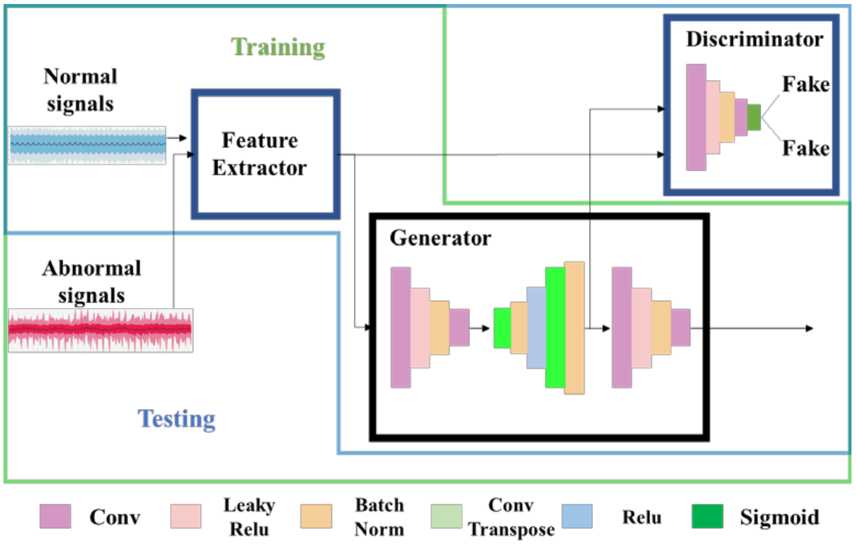

Figure 1.10 Network architecture of generator and discriminator based on deep convolutional GAN.
Adapted from [24].

Another practical application is GAN-based robotics control. Generative modelling helps reinforce ML models, so they are less biased and comprehend more abstract concepts, both in simulations and the real world.

GANs generate data that are like real data; therefore, they are widely used in industrial applications. They also have advantages over methods of supervised and unsupervised learning. A GAN is a semi-supervised learning framework that uses manually labelled training data for supervised learning and unlabelled data for unsupervised learning to build models that can make predictions beyond the labelled data by leveraging the same.

The other two classes of generative AI techniques are AR-CNN and Transformer-based models. AR-CNN are used to examine systems that evolve, predicting future outcomes of a sequence from the previously observed results of that sequence. They rely on previous time-series data to generate accurate new data as an autoregressive model is a feed-forward model which predicts future values from past values. Transformer-based models are used to analyse data with a sequential structure and have become a standard tool for processing sequential input data, such as natural language. Core to their architecture is the ability to identify and learn context within an input sequence and thus refine the meaning of the other part of the sequence (the so-called attention mechanism).

1.4.7 Artificial Swarm Intelligence

Swarm intelligence is a branch of AI that is based on an extrinsic type of intelligence inspired from nature and biological systems and is connected to collective behaviour of decentralized and self-organised systems.

Swarm intelligence systems typically consist of many independent but similar individuals that follow very simple rules without a centralised control system. These systems' overall behaviour is a result of the interactions of the individuals, with each other and with their environment, but globally they act quite intelligently.

For example, ant colonies can optimise routes and shortest paths, and bee colonies can find the location of their nest in an extremely efficient manner. Swarm logic is a behaviour demonstrated by many animals, and while each individual is less capable of independently making decisions or solving problems, in a swarm they communicate, coordinate, organize, and seemingly problem solve, seemingly without a central command.

The essence of swarm logic is the sharing of information, along with interaction with other individuals and the surroundings, to derive new information

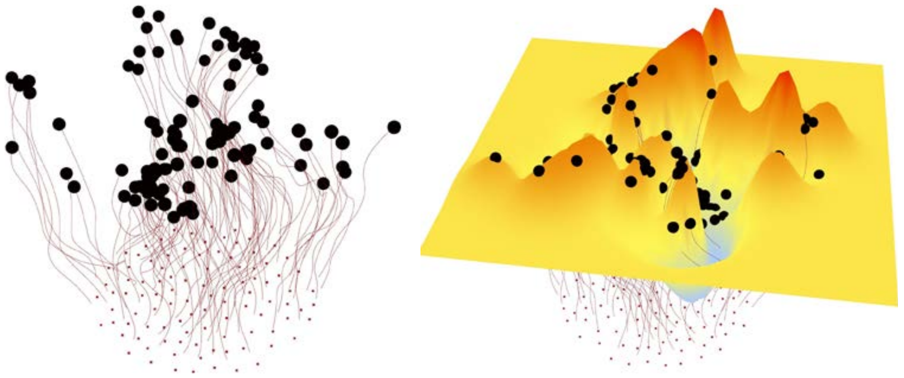

Figure 1.11 Swarm intelligence visualized: population of agents searching for a destination (left) and search space represented by a nonlinear regression generated surface (right).

as a basis for global actions. Adopting a broader perspective, swarm intelligence is the action of having decentralised agents swarm collectively towards a goal. These agents can be ants, bees, cars, robots, among other things. An IIoT system can be seen as multiple agents, where the intelligence lies both within agents and in their interaction with each other.

Figure 1.11. intuitively illustrates the concept behind swarm intelligence, the starting point of which is a population of agents (like birds or bees) searching for a destination (left). Complicated intelligent behaviour to solve complex tasks emerges from simple agents following simple rules such as keeping diverging trajectories, avoiding collisions and interaction with near neighbours (rule is known as self-organization). These agents will simultaneously know when the destination is reached, based on the goal parameters of the destination. Swarm intelligence's aim is to optimise the goal parameters and minimise the search space, represented by a surface generated using nonlinear regression (right).

1.4.8 Natural Language Processing

Natural language processing (NLP) is a branch of AI that focuses on developing algorithms to enable computers to understand speech and text. NLP systems are developed to imitate the human capacity to use language. NLP is used in a variety of tasks, including text understanding, text summarization, information extraction, machine translation, and speech recognition and synthesis. Examples of AI techniques include support vector machine (SVM), for text classification (such as spam detection); hidden Markov models (HMMs) for speech or text generation; neural networks, for machine translation; and logic-based methods, for text summarisation.

NLP technology has made major progress in recent years, leading to the development of network architectures better able to learn from complex and context-sensitive data. These advances have been supported by the constantly increasing data resources from intelligent sensors and computing power.

Current challenges include obtaining quality data and detecting and removing data biases. Future applications are expected to meet these challenges, as well as to improve human–AI interactions across diverse languages and situations.

NLP is too computationally expensive to run on microcontrollers, so applications running on edge devices are often limited to looking for keywords in speech, such as short commands for executing some actions. Identifying non-voice sounds is also extremely useful. NLP based on embedded machine learning will make edge devices more intelligent in future applications.

1.4.9 Machine learning

Machine learning is a branch of AI that provide systems with the ability to automatically learn and improve their performance in some tasks through experience without being explicitly programmed. The rules of ML programs are not determined in the same way as those of normal computer programs are; instead, ML uses specialised algorithms to *learn* rules from data, in a process known as *training*.

This training process starts with feeding data and then training the machines by building various models using different algorithms. The choice of algorithms depends on the kind of task we are attempting to automate. Most ML tasks are narrowly specified to optimise specific functions using particular data set.

Inference is the process of taking a trained model and deploying it into a device, which will then process incoming data to look for and identify whatever it has been trained to recognise.

During the inferencing phase, predictions and decisions are made concerning new data, based on the learned parameters. Prediction is the process of using a model to make a prediction about something that has yet to happen. Inference is the process of evaluating the relationship between the predictor and response variables.

Deep learning and neural networks are examples of **ML techniques** frequently used today. **Deep-learning** (DL) systems learn from large amounts of data to subsequently recognise and classify related, but previously unobserved, data. For example, **neural networks**, often described as being loosely modelled after the human brain, consist of thousands or millions of processing nodes generally organised into layers. Advances in hardware have allowed these networks to have many layers, which is what puts the "deep" in deep learning. What differentiates DL from ML techniques is the former's ability to extract features automatically.

Humans and machines both acquire knowledge in the process of learning based on experience; however, the former do so based on either direct or shared experience, while the latter do so through experience shared in the form of past data. With respect to which input data an ML process receives and how it handles this data, three types of **ML training methods** can be distinguished: supervised (labelled data required), unsupervised (no labelled data; these attempt to discover patterns) and reinforcement learning (actions taken to maximise cumulative rewards).

Supervised-learning algorithms learn from labelled input data and are widely used for classification and regression tasks. The system learns which components of the data are useful for classifying it correctly and uses that information to correctly classify data it has not encountered before. Such algorithms can also detect patterns in data and then use the uncovered patterns to predict future data or other outcomes of interest. By contrast, **unsupervised-learning** algorithms seek to discover hidden patterns and other underlying structures in unlabelled data and are used in clustering tasks.

Reinforcement learning algorithms enable computer programs to learn from experience and to be rewarded for reaching specified objectives – both immediate actions and long-term goals. Reinforcement learning is akin to how humans learn from their own mistakes over time through trial and error. This means that the algorithm decides the next action by learning behaviours that are based on its current state and that will maximise the reward in the future.

More detailed description of the learning algorithms can be found in Section 1.8.1.

1.4.10 Neural Networks

This section presents a high-level overview of neural networks (NNs) thus providing a valuable and intuitive understanding of how the models work when deployed in edge devices and operated within industrial settings.

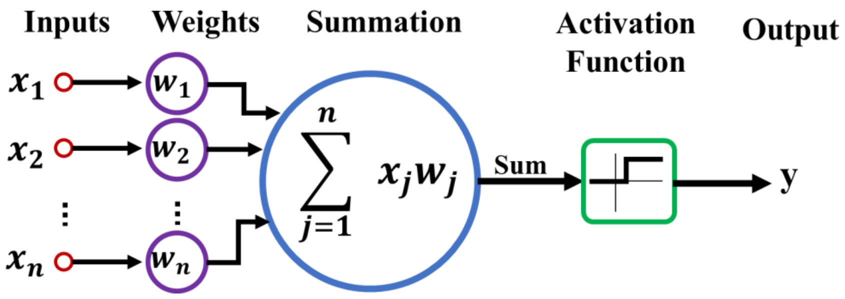

Figure 1.12 Perceptron illustration.

Neural networks simulate the learning capacity of biological neurons in the human brain. The fundamental unit of a neural network is a perceptron (Figure 1.12).

The perceptron model multiplies all inputs with a weight parameter, whose value is representative of how important each feature is in the calculation of the results. The resulting values are added together with a bias term, resulting in the so-called weighted sum, on which an activation function is finally applied. The activation functions introduce non-linearity in NN models, thus differentiating them from linear regression models.

During training, model parameters are gradually calibrated so that the NN output comes increasingly nearer to the desired one, when given a certain input. A loss function keeps track of how far the model is from predicting the correct output, meaning that the higher the loss value, the less efficient the model is at predicting. Accuracy is another metric, inversely proportional to the loss.

Anyone interested in more detailed reading regarding training is directed to [4][8] and other sources. Generally, the first step is to forward feed one signal sample or a batch of samples through the network. Feedforward is the process of passing input values, through the hierarchical layering of neurons, to produce an output in the final layer. Network loss is then calculated by comparing the predictions with the actual outputs and this is then used to update the model's parameters in the next step known as backpropagation. Backpropagation is the process by which the error contribution of each neuron is calculated and passed backwards through the network. The weights and biases are adjusted proportionally to this error contribution, and this is how the machine learns.

This stepwise procedure is run several times with the aim of improving the output of feedforward, ultimately optimising the network's predictions.

One feedforward and backward pass is called an iteration, while a pass of the entire dataset is called an epoch. After each epoch, the algorithm will perform a forward pass of each validation sample, looking at loss and accuracy. Usually, accuracy improves over time as loss drops.

The number of epochs and the learning rate, i.e., how much the model's internal parameters are updated during each training step, are known as hyperparameters, and can be configured to make a model more efficient.

1.4.11 Automated Planning and Plan Recognition

Automated planning is a branch of AI that concerns providing goal-oriented, deliberative behaviour to both physical and virtual intelligent agents [29]. It takes as inputs a planning domain, an initial state and a goal, and it employs optimisation algorithms to return a sequence of actions that guides the agent's behaviour. The correct representation of states, conditions and actions and the suitable algorithms all contribute to the agent reaching its goals and optimising performance.

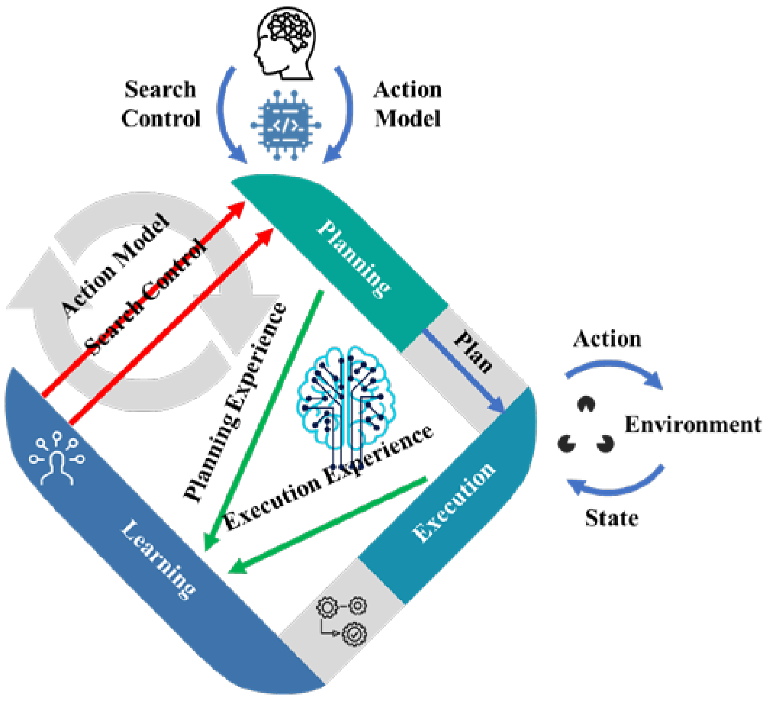

Figure 1.13 Automated planning, states, and actions.

In many industries, automation is an emergent trend that requires efficient automated planning, such as robotic and autonomous systems. Mobile and fixed robotic systems can perform various tasks in the industrial application domain without the need to acquire knowledge, relying on only the accuracy of the model. The model encompasses explicitly represented domain knowledge acquired from human experts.

Incorporating AI capabilities in automated production planning in manufacturing also has significant potential. Embedded industrial AI optimisation algorithms can balance the product result and the resources used during production and can learn by collecting considerable amounts of "cause and effect" data that can be used for what-if simulations and analysis.

Embedded edge AI solutions for path planning for swarms in mobile autonomous systems are evolving, and they have been applied in several manufacturing optimisation and logistics applications. Swarm automated planning algorithms are used as planning methods based on planning graph technology to improve the searching efficiency using swarm intelligence in fleets of autonomous devices operating on the manufacturing floor.

AI planning techniques are widely used when explainability is necessary, i.e. the planner can explain why a specific course of action has been chosen.

On the downside, there are some challenges to AI planning techniques when they are employed in real-time applications, due to slow response time. The more complex the planning domain, the larger the search space becomes, thus increasing the response time to find a proper sequence of actions. This is especially critical when planning and acting are intertwined.

An alternative to acquiring knowledge from human experts is to learn the model in time. Architectures relying on ML have the advantage of not requiring much prior knowledge about the domain; once trained, they act quickly. Nevertheless, they need a large amount of data for the training. They are usually limited to the industrial application domain they were trained for and presenting them with new situations might be challenging.

The two approaches can be incorporated into the same agent architecture, thus achieving better trade-offs than if only one approach were used. More about the synergistic benefits of combining symbolic AI and ML can be found in Section 1.6.

Plan recognition deals with inferring the goals or plans that explain the observed actions of an agent; as such, it is considered the opposite of planning. Plan recognition algorithms require knowledge about the potential behaviours of the agent and how the agent makes its decisions. When this

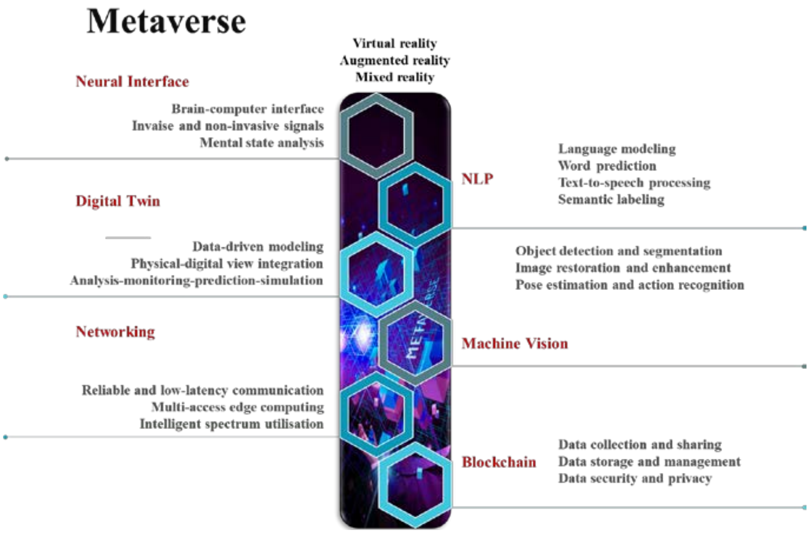
Metaverse
Virtual reality
Augmented reality
Mixed reality
Neural Interface
Brain-computer interface
Invaise and non-invasive signals
Mental state analysis
Digital Twin
Data-driven modeling
Physical-digital view integration
Analysis-monitoring-prediction-simulation
Networking
Reliable and low-latency communication
Multi-access edge computing
Intelligent spectrum utilisation
NLP
Language modeling
Word prediction
Text-to-speech processing
Semantic labeling
Object detection and segmentation
Image restoration and enhancement
Pose estimation and action recognition
Machine Vision
Blockchain
Data collection and sharing
Data storage and management
Data security and privacy

Figure 1.14 Application of Metaverse.

knowledge is unavailable, neural networks can be employed to learn the decision model automatically.

1.4.12 AI for the Metaverse

Metaverse is a term formed by combining meta and universe and has been introduced as a shared virtual world that is fuelled by many emerging technologies, such as virtual reality, and AI. AI has shown the great importance of processing large amounts of data to enhance immersive experience and enable human-like intelligence of virtual agents.

ML algorithms, DL architectures and other emerging technologies such as swarm intelligence have had a role in the foundation and development of the metaverse, such as AR, VR, mixed reality (MR). These are now ready to be employed in applications such as machine vision, blockchain, networking, digital twin, and in different industrial applications (Figure 1.14).

1.5 Edge AI continuum

Edge processing can redefine the landscape of interconnected devices by moving data processing and analytics to the edge and employing AI techniques and embedded security. Edge AI computing and processing allow for the development of new real-time applications due to the processing being performed close to the data source. It can reduce the amount of transmitted data by transforming extensive amounts of raw data into essential insight data. It can also decrease communication bandwidth and data storage requirements while reducing energy consumption and increasing security, privacy, and data protection.

Edge AI technology developments are used to implement applications that benefit from AI-based technology advances across the edge continuum.

Various forms of AI have already been adopted by multiple industries, governments, and society. However, a breakthrough is needed in several industrial sectors to bring the intelligence close to the data source and implement it in industrial processes. However, this breakthrough may face several hurdles that challenge its advancement.

Leveraging AI methods and techniques at the edge is essential for increasing the performance and capabilities of the intelligent sensor systems and IIoT devices used in various industrial applications. The edge AI processing concept is reflected in the emergence of micro-, deep-, and meta-edge layers for several industrial intelligent applications.

The edge processing continuum includes sensing, processing, and communication units close to physical industrial assets (micro edge), gateways and intelligent controller processing (deep edge), and on-premises multi-use computing (meta edge). This computing continuum creates a multi-level structure that advances processing, intelligence, and connectivity capabilities.

The edge AI computing and processing concept for intelligent applications is mirrored in the development of different edge-processing levels. Figure 1.15 shows an all-encompassing edge AI architecture incorporating the computing and intelligence continuum from sensors and actuators, processing, units, controllers, gateways, and on-premises servers to multi-access, fog, to cloud computing interfaces.

Edge AI computing and processing device functions cover edge computing, communication, and data analytics capabilities, which make it smart/intelligent. An edge AI computing and processing device is designed around the computing units (CPUs, GPUs, FPGAs, application specific integrated circuits (ASICs), AI accelerators/processing), communication networks, storage infrastructures, and applications (workloads) that run on it. Single- and multi-core microcontrollers (MCUs) are based on ARM Cortex-M cores or on cores using new open-source RISC-V instruction set architecture (ISA) and high-performance embedded processors with varying

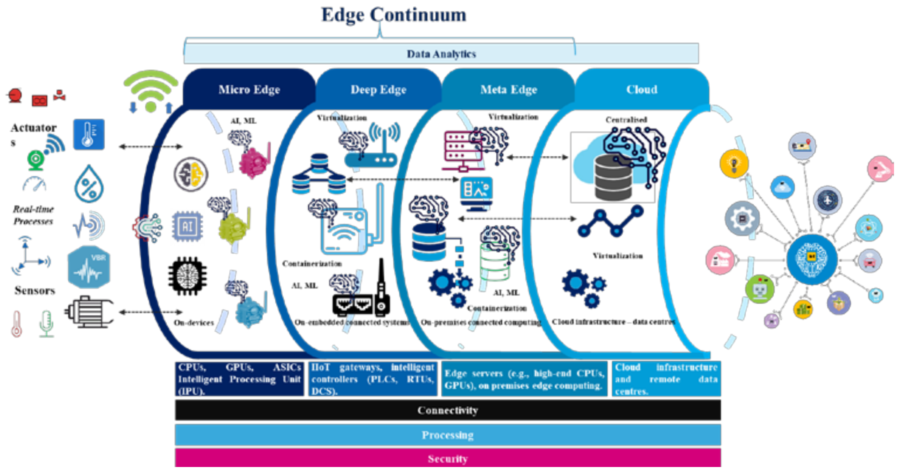
Edge Continuum
Data Analytics
Micro Edge
Deep Edge
Meta Edge
Cloud
Actuators
Real-time Processes
Sensors
VBR
AI, ML
On-devices
Virtualization
Containerization
AI, ML
On-embedded connected systems
Virtualization
AI, ML
Containerization
On-premises connected computing
Centralised
Virtualization
Cloud infrastructure – data centres
CPUs, GPUs, ASICs Intelligent Processing Unit (IPU).
IIoT gateways, intelligent controllers (PLCs, RTUs, DCS).
Edge servers (e.g., high-end CPUs, GPUs), on premises edge computing.
Cloud infrastructure and remote data centres.
Connectivity
Processing
Security

Figure 1.15 AI across the edge continuum.

capabilities. The memory footprint, computing time, transmission, and power consumption requirements always depend on whether the device operates at the micro, deep, or meta edge. ML and DL models need to be converted into efficient formats before compiling and flashing them into edge devices.

AI building blocks are optimised for the type of processor, the amount of RAM, and the number and types of sensors. The solutions are usually provided as a C library, which can be embedded into the main microcontroller program and compiled and downloaded into the embedded system.

The edge can scale from a few devices to tens of thousands of devices distributed in various locations with unique identities. Edge AI computing and processing devices are physically separated, yet they can be connected by wireless/wired topology connections, such as mesh topologies. Edge AI computing and processing devices can operate independently, and local decisions can be supported by inference actions, including the unexplored evolution of training on edge devices.

Micro-edge

The micro edge includes intelligent sensor systems (physical, chemical, environmental parameters, perception, etc.) with processing and connectivity capabilities that use IIoT devices 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) is minimised, and the micro-edge devices have cost and power consumption constraints. Micro-edge hardware devices implement analytics and intelligent functions by integrating AI-based components and algorithms and running the AI algorithms for inference, training, and self-training. The intelligent micro edge makes IIoT real-time applications ubiquitous and merges them with the industrial environment.

Deep-edge

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 IIoT 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.

Meta-edge

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.

Fog computing extends the computing capabilities of industrial systems and interfaces cloud computing capabilities with the edge of the network. Fog computing enables repeatable structures in the edge computing concept, so enterprises can push computing out of centralised systems or clouds for a better and more scalable performance. A Fog computing implementation is a virtualised platform located between cloud data centres (hosted within the Internet) and end-user devices that provides support for edge processing and is complementary to cloud computing platforms.

AI models increase various potential industrial applications; however, developing AI functionalities for the edge continuum is complex and presents several challenges, such as scalability, interoperability, and performance optimisation versus the resource constraints of the edge devices. Overall, implementing AI models on edge-embedded devices has advantages for different use cases and applications in various industrial sectors.

A key element for the transition of AI processing to the edge is the capabilities of the developer edge environment, covering the hardware, interfaces, platforms, training/learning, applications, and services. The intelligent infrastructure at the edge refers to the tools, platforms, and techniques used to run store data, build, and train AI/ ML algorithms, and the algorithms themselves.

1.6 Symbolic AI – ML Continuum

Human intelligence comes in two distinct but complementary forms of arriving at conclusions, one based on structured and rational decisions and the other on perception and understanding patterns. Machine intelligence also comes in two similar forms, one based on symbolic knowledge representation and reasoning (the symbolic AI approach) and the other on deep learning and the interpretation of data patterns (the ML approach).

Therefore, when faced with an AI problem, one can look for a solution combining technologies in the symbolic AI – ML/DL continuum, instead of choosing between the symbolic or ML/DL approach in solving it. Nevertheless, it is essential to understand the difference between and the advantages and disadvantages of these two approaches.

Generally, the symbolic AI approach is suitable when the AI problem is abstract, no large amounts of data about the AI problem are available (for example, data coming from sensors, such as images, sounds, etc.) but the steps to the solution are commonly known so that this knowledge can be modelled explicitly.

On the contrary, ML is useful when the steps to the solution are not known, but the large amount of data allows us to look for larger patterns, which may ultimately lead to the likely solution. This approach requires several iterations and massive computational power to arrive at a conclusion. Nevertheless, as computing hardware becomes faster and cheaper and ML algorithms become more powerful, ML becomes more inclusive (i.e., available not only to actors with strong computational resources).

The concept is easier to grasp if we consider a simple-use case of the automated diagnosis of a malfunctioning motor problem. In the case of symbolic AI approaches, this would require that a human expert fully describe the motor and its features, functioning and malfunctioning situations. This knowledge would then be represented in a form that could be processed by machines. With the help of algorithms and step-by-step inferences from this knowledge base, it is possible to arrive at a diagnostic for the motor problem when fed real-time sensor data.

The advantage of this approach is that it does not rely on massive data and might work for most motors. However, the knowledge base takes time to acquire and set up, and if some knowledge (about a particular motor) is missing or incomplete, it will yield no result.

The ML approach would be to feed a neural network with many signals/data of the motor, vibration data and audio data in both functioning and malfunctioning situations. The trained network would then be able to accurately guess the motor diagnosis when fed real-time sensor data. The advantage of this approach is that it does not rely on a motor expert's knowledge to be made explicit, and it allows for automation due to its ability to handle large amounts of real-time sensor data.

It is technologically possible to combine symbolic AI and ML, for example, by using symbolic AI to generate answers (constraints) and then feeding these answers to ML to generate predictions. A balance between the two can be achieved based on experimentation.

In short, with symbolic AI, the rules of the AI algorithms are decided by a human. These rules and some data are provided as input to the AI algorithms, and data are processed according to these rules to produce answers in the output. With ML, on the other hand, the inputs to the ML algorithms during the training process include some data and answers, while the rules are the output. These rules are then used during inference to produce predictions about input data that have not been seen before (i.e., data that was not part of the training).

Therefore, AI can also be understood from the perspective of combining technologies in the symbolic AI – ML continuum and balancing them to achieve better trade-offs than otherwise achieved if only one technology were used.

1.7 Logic-based AI: Knowledge Representation and Reasoning

As full-scale AI applications increase in number and complexity, accelerating digital innovation across industries and boosting productivity, so does the need for AI to be more comprehensible, explainable, and therefore trustworthy. Thus, symbolic approaches to classical AI are re-gaining momentum.

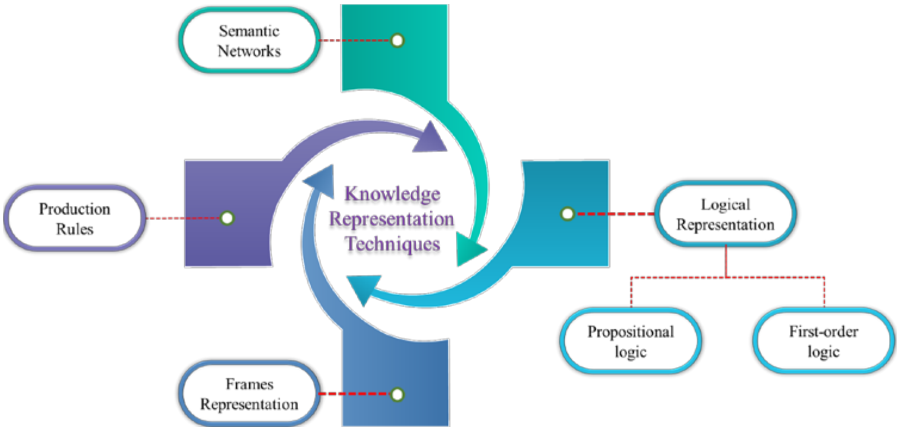

Figure 1.16 Knowledge representation.

This section summarises some of the logic-based approaches that are likely to be adopted by various industrial sectors and discuss future perspectives for exploiting logic-based technologies.

Intelligent machines require knowledge to make intelligent decisions, the same way as humans do. This usually entails that expert knowledge need to be acquired and represented in a form that machines can process, called a knowledge base. Predicate logic and propositional logic are representative ways to reflect knowledge; semantic networks, rules, frames, or programming languages are also good examples (Figure 1.16). Languages that are designed specifically for AI include LISP and Prolog.

A knowledge representation should have specific properties, for example be unambiguous, easy to use, inferential adequate and efficient, and able to represent all types of knowledge: declarative, procedural, heuristic, structural, meta-knowledge (Figure 1.17). The choice of knowledge representation method largely depends on the problem to solve.

Inference is a term representing the derivation of new knowledge from existing knowledge and axioms (i.e., rules of derivation) within a single step, using logical constructions. The rule of derivation can be one of many kinds, such as, induction, deduction, and abduction. Modus ponens (*if A is true, then B is true. A is true. therefore, B is true*) and modus tollens (*if A is true, then B is true. B is not true. therefore, A is not true*) are two such logical argument constructions.

Reasoning is a term used in the context of a goal (e.g., proof whether a propositional statement is satisfiable or not) and is carried out via a search

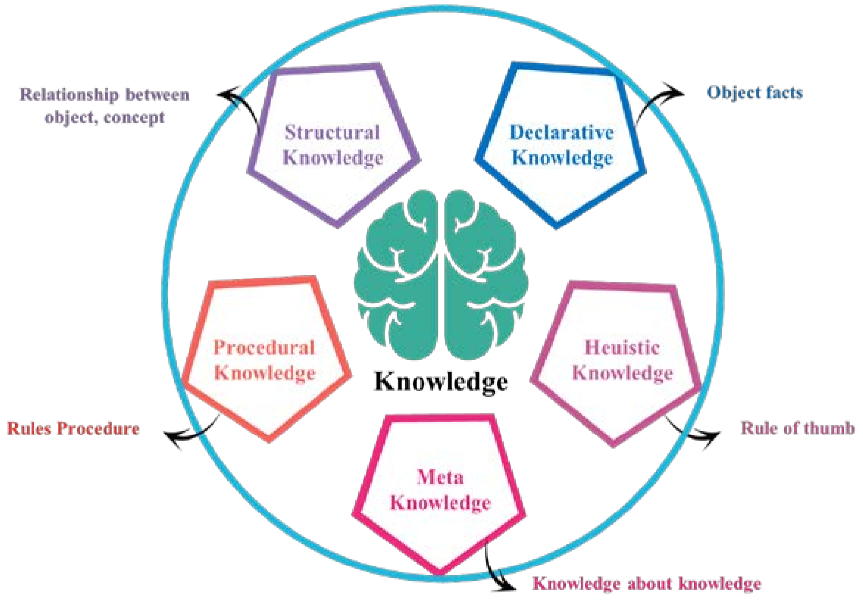

Figure 1.17 Type of knowledge.

process involving multiple inferences. Choices during such search have to be made such as which axiom to “fire” along with which knowledge in order to derive new knowledge. Resolution is a particular kind of reasoning involving the “resolution rule”.

Reasoning from premises to logical consequences, have been a major part of AI since its beginnings. Inferences are steps in reasoning, moving from premises to logical consequences. One motivation behind is the utilization of knowledge of a domain for obtaining answers for given problems. In this case knowledge must be available in a formal form like propositional or first-order logic. Reasoning and mechanical theorem proving is used for computing an answer using the formalized knowledge directly. It is worth noting that this kind of application of logic comes with several advantages, i.e., making knowledge explicit (and thus understandable for humans), allowing to use the same knowledge for various problems, and allowing to explain solutions based on knowledge. On the downside logical theorem proving requires high computational resources, but which are widely available today.

For more information about the foundations of logic (and in particular propositional and first-order logic) we refer the interested reader to [8] (with the direct context to AI).

To solve this problem, other classes of (non-monotonic) logic has been proposed like default logic [15], and abduction, which is also non-monotonic. In the past decades, research in non-monotonic logics and their applications has been a very active part of AI. This includes model-based reasoning [17][18] with a strong relationship to default logic, and also more recently answer set programming (ASP) [19]. All these inference mechanisms can be used to solve practical challenges, like diagnosis and fail-operational behaviour. More about these topics and reasoning from first principles for self-adaptive and autonomous systems can be found in [20].

Logical inference has been an active research area of AI since its beginnings, ranging from expert systems to more recent developments on non-monotonic inference. Due to the increased available computational power and the availability of efficient reasoning and inference engines the direct use of knowledge formalized in ontologies and knowledge bases for solving various tasks can be achieved. Recent work describing a mapping from neural networks to a logical representation can be found in [21][22].

1.8 Hardware/Software Technology Stack

Technology stacks are widely used to structure technologies in a particular area. AI is no exception, as it is possible to conceptualise AI as a technology stack with various layers. A five-layer stack is presented in Figure 1.18.

During the past decades, the focus has moved back and forth between logic (symbolic reasoning) and pattern recognition (neural networks), driven by the varying abilities of technologies to acquire data, learn, derive new information and reason to reach decisions. In the last years, machine learning and neural network models have been the primary focus due to advances in hardware development and processing capabilities. Hence, the technology stack is illustrated by machine and deep learning, covering topics such as learning/training and inference.

The foundation of the stack is represented by the hardware layer, which contains at least three sets of components that reflect the processing units responsible for performing specialised AI operations. The neuromorphic hardware components consist of new ultra-low-power silicon chip architectures (e.g., neuromorphic modules and chips, analogue NN, spike NN) that incorporate different chip designs and algorithms to mimic how the human brain works. The accelerator set of components consists of silicon chips designed to perform the highly parallel functions required during training

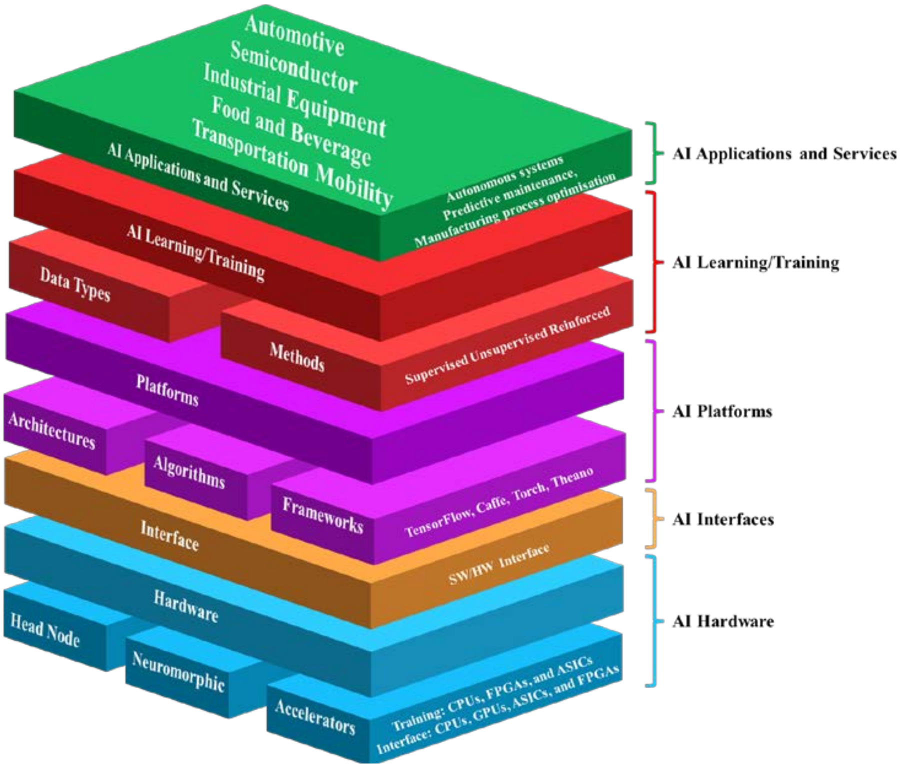

Figure 1.18 Five-layer (with sublayers) AI technology stack.

and inference, such as GPUs, FPGAs, or ASICs. The head node components are units that coordinate computations among accelerators.

The platforms layer is used for AI and ML/DL deployment and consists of three sublayers, the goal of which is to abstract firmware from the underlying hardware. The frameworks sublayer consists of packages that trigger HW algorithms, such as Caffe, Torch, Theano, etc. This happens through the interface layer, which connects the hardware and platform layers and is in charge of facilitating communication between them. 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 AI training/learning layer consists of two sublayers. The methods sublayer involves techniques for optimising the model for specific domain data, such as supervised, unsupervised and reinforcement learning. The data type sublayer consists of categories of domain input data, such as labelled and unlabelled data.

Finally, the applications and services layer incorporate ready-to-use AI functionality into solutions to real industry problems and use cases, such as autonomous vehicles and object recognition. The solutions can be customised based on generic data or on 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. The elements presented in the different sections - spectrum, continuums, methods, techniques, concepts, and others - are all connected through the AI technology stack defined by European projects such as AI4DI [3].

This section briefly introduces the industry-adopted ML terms and the ML methods such as supervised, unsupervised, and reinforcement learning, and neural networks architectures. Specifically, the focus is on embedded ML, for which the advances in hardware architectures opened an entirely new space of applications and opportunities. The new hardware architectures make possible to run complex ML workloads on microcontrollers, with limited compute and memory profiles.

1.8.1 ML Methods and Techniques

There are a multitude of methods and techniques that depend on the type of learning, and the type of learning – supervised, unsupervised, or reinforcement –depends on the data available for the application. A taxonomy is shown in Figure 1.19.

Supervised Machine Learning

Supervised learning algorithms learn from a training set of data that is labelled with the correct description; the system subsequently learns which components of the data are useful for classifying it correctly and uses that information to correctly classify data it has never encountered before. These algorithms are widely used for classification and regression tasks, as detailed below.

Regression is considered the fundamental ML paradigm. The process of regression connects outputs to inputs. It shows an output for a given input, and the regression component creates a transfer function to best fit that

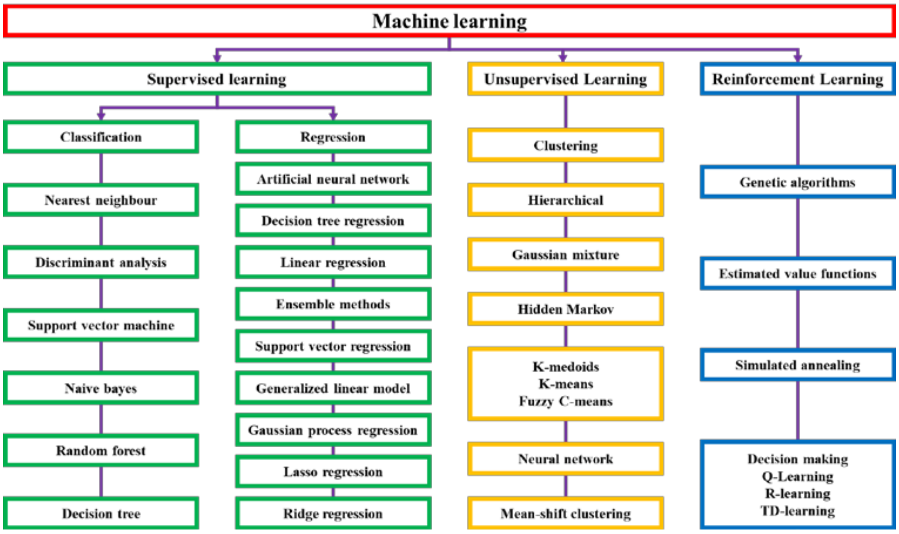

Figure 1.19 ML taxonomy.

data. This transfer function then provides a method to predict an output for an untested input. In other words, if the independent variable is time, then the model forecasts future values; otherwise, the model predicts present but unknown values. Typically, when selecting a regression strategy, the number and type of inputs and the type of transfer functions need to be considered. The transfer functions can be represented as curves (Figure 1.20 left) and surfaces (Figure 1.20 right).

There is a wide variety of regression strategies employed in industrial applications such as simple linear regression, polynomial regression, logistic regression, support vector for regression (SVR), decision tree regression, random forest regression. Figure 1.21 shows an example of logistic regression that predicts a binary outcome, such as normal or abnormal, based on

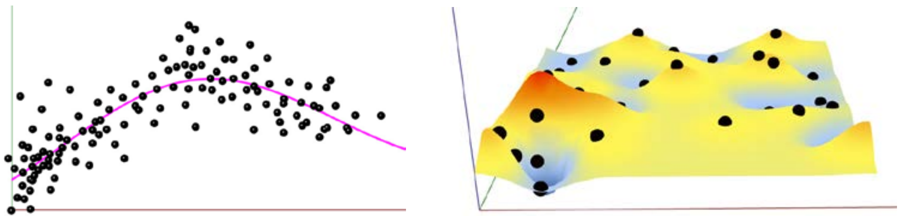

Figure 1.20 Regression visualized 2D (left), 3D (right).

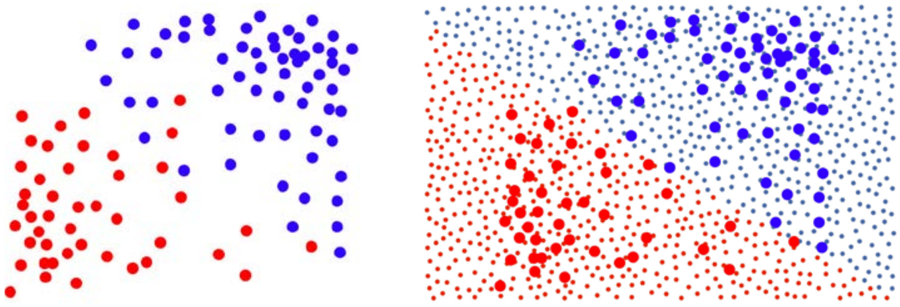

Figure 1.21 Normal(blue) - abnormal(red) (left). Predicted values using logistic regression (right).

observations of the data set, which could be motor vibration measurements. The large dots are the learning data, while the small dots are the data to test against learning data.

Classification addresses the problem of determining the class that a given data instance belongs to. It requires more input, i.e., training data must be provided for the definition of classes. The more training, the more accurate the classification algorithm. Given sufficient training data, classification tools can distinguish between classes as well as or better than humans.

Many of the most powerful applications of ML are classification systems. Neural networks based on the layered architecture of biological brains have emerged as a common classification technique because they are able to group explicit, visible features into abstract or inferred features that correlate closely to the predefined classes in the training data.

Classification methods are widely used in machine vision with the classification of images, e.g., to determine whether an image contains specific objects. Another example is with time series, e.g., motor classification in predictive maintenance. Among their other benefits, classification tools can extend automation to incorporate the ability to differentiate inputs automatically, alleviating one of the most time-consuming manual steps in the generative-design workflow.

The intuitive images in Figure 1.22 show how the two classification and regression can be distinguished. Regression searches for a line or plane that fits the given input points, while classification searches for a line or surface to separate the classes.

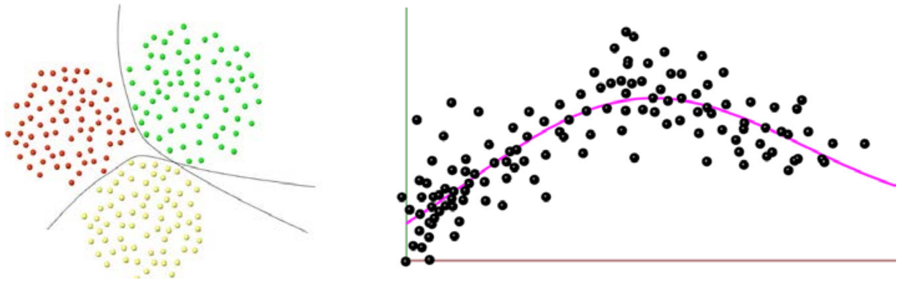

Figure 1.22 Classification (left) vs Regression (right)

Unsupervised machine learning

In contrast to supervised learning, unsupervised learning algorithms search for underlying structures in unlabelled data. Unsupervised learning is where there is only input data and no corresponding output variables. The goal for unsupervised learning is to model the underlying structure or distribution in the data to learn more about the data. These algorithms are widely used for clustering and dimension reduction tasks, as detailed below.

Clustering is one of the most flexible techniques in ML: it is easy to apply and requires no sample data or predetermined classifications. Clustering algorithms group together data with similar characteristics without any prior training or guidance on how to distinguish between groups. This is very powerful precisely because it is so flexible. The raw data and the number of groups are given as inputs, and the clusters are generated as outputs. K-means is one of the most used methods for clustering, where K is the number of clusters to be created. In short, the algorithm places the centres of the K clusters in the data set, assigns the closest points to the K cluster and recalculates the centre of the cluster iteratively. Another powerful clustering algorithm is the Gaussian mix (example in Figure 1.23). The better the data describes different features within the data, the better or likelier the grouping result.

Clustering is powerful in its flexibility and simplicity. Often, clustering is the starting point when organising poorly structured data or sorting continuous data into useful groups. On the downside, the results are difficult to control precisely and depend on the resolution of the input data.

Dimension reduction is used to simplify the model by removing the less important or redundant information from the data set to make it manageable while maintaining relevance and performance. Data sets can sometimes have

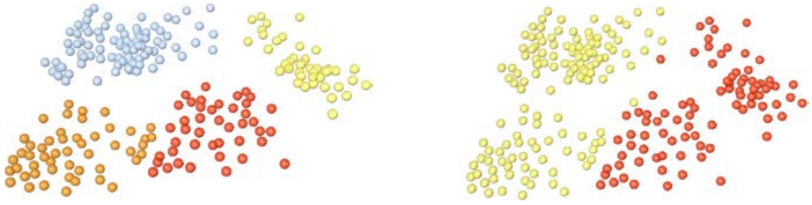
Figure 1.23 Cluster (Gaussian mix) 4 clusters (left) vs 2 clusters (right).

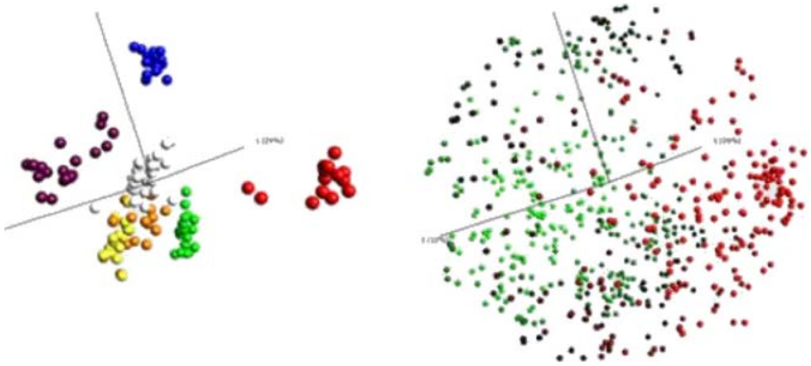
Figure 1.24 Principal component analysis. Intuitive visualisation, select variables that capture the largest variability in data.

hundreds of features, and by extracting fewer independent features, the complexity of the model can be greatly reduced. The most used algorithm is PCA (Principal Component Analysis), which finds new vectors that maximise the linear variation of the data by drastically reducing the size of the data without losing too much information (Figure 1.24). Another commonly used method is t-Stochastic Neighbour Embedding (t-SNE), used for automatic learning by reducing the space of functions.

There are two types of dimensionality reduction techniques: feature selection and feature extraction. Feature selection techniques are backward elimination, forward selection, bidirectional elimination, score comparison and more. Feature extraction techniques are, Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Kernel PCA, Quadratic Discriminant Analysis (QDA).

Reinforcement learning

Reinforcement learning (RL) enables computer programs to learn from experience by trial and error and to be rewarded for reaching specified objectives

– both immediate actions and long-term goals. The two main components are the environment, which represents the problem to be solved, and the agent, which represents the learning algorithm.

Different from other ML approaches is RL's emphasis on simulated motivation and learning from direct interaction with humans and the environment, without requiring explicit examples and models. RL is akin to how humans learn from their own mistakes over time through trial and error. This means that the algorithm decides the next action by learning behaviours that are based on its current state and that will maximise the reward in the future. RL shifts the focus of machine learning (ML) from pattern recognition to experienced-based sequential decision-making and execution. Many applications in robotics and machine vision use RL to perform tasks.

One of the core concepts in RL is the Q-Learning, which is about learning an action-value function, representing the measure of the overall expected reward assuming the agent performs the action. A simple data structure such as a table can be used to keep track of the states, actions, and their expected rewards. In case of an infinite state space, this function is implemented with DNNs, hence the term deep Q-learning illustrated in Figure 1.25.

Deep RL has demonstrated great potential for addressing the challenges of real-time decision-making based on information captured by sensors. The

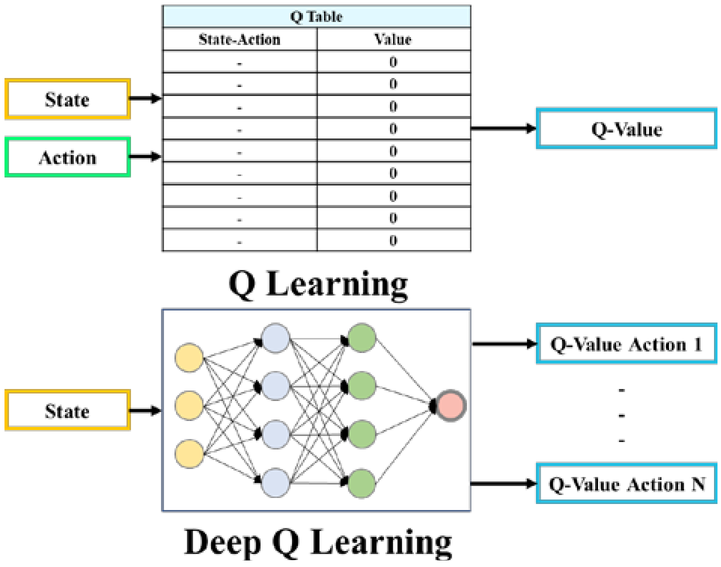

Figure 1.25 Q-Learning vs Deep Q-Learning.

increased complexity of sensor-intensive systems with expensive subsystems and costly repairs requires efficient real-time control and decision-making approaches. Thus, many research efforts have recently been devoted to applying deep RL to the field of predictive maintenance [28].

1.8.2 Neural Networks Architectures

An artificial neural network (ANN) encompasses any form of a DL model and can have one hidden layer connecting the input and the output. DL is a class of machine learning algorithms that uses multiple stacked layers of processing nodes to learn high-level representations from data, such as images, audio, and text. ANN can have many hidden layers, in which case they are called "deep", hence the term deep neural network (DNN). By adding more hidden layers, the model gets more parameters, which in turn allows the model to fit more complex functions.

A DNN consists of a series of stacked layers, and each layer is made up of nodes that are connected to the previous layer's nodes through a set of weights. By stacking layers, the nodes in each subsequent layer can represent increasingly sophisticated aspects of the original input. Understanding how each layer changes the shape of the data as it flows through the network is a key aspect of truly understanding the mechanics of DL. There are many different types of layers, but one of the most common layers is the dense layer that connects all units in the layer directly to every unit in the previous layer.

The DNN architecture is forward in nature, i.e., the information does not shift between two consecutive layers, i.e., the layers give no feedback to the previous layers. A feed forward neural network (FFNN) is the most basic type of multi-layer NN, and as the name suggests, information is passed in the forward direction. Data flows from the input layer to the output layer without going backwards, and the links between the layers move one way, which is in the forward direction. FFNNs are the foundations of deep networks, such as CNN and RNN. Other architectures include LSTMs.

CNN is an FFNN that is generally used for image/object recognition and classification and for other complex classification problems, such as predictive maintenance. CNNs can extract the local features of the input data and combine them layer by layer to generate high-level features. As illustrated in Figure 1.26, a typical CNN has two phases. The first phase is a series of convolutions of layers, usually followed by pooling layers, while the second phase is a series of dense layers. CNNs can be used for deep learning with

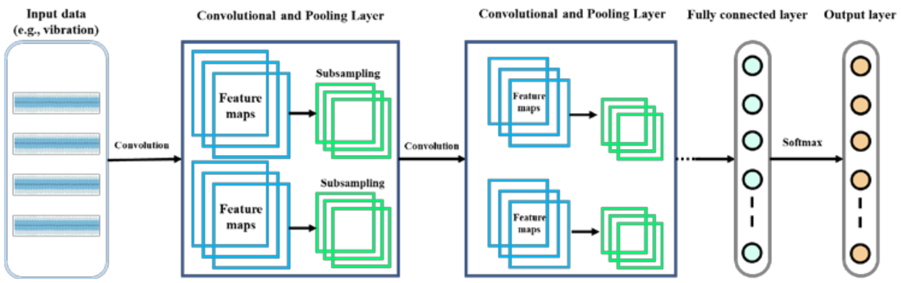

Figure 1.26 Typical CNN architecture.

a few parameters; thus, there are fewer parameters to learn as compared to dense layers.

RNN is the time-series version of an FFNN. It has connections between passes and through time. The connections form a directed graph along a sequence of features that link one layer to previous layers, allowing information to flow back into the previous parts of the network. Thus, each model in the layers depends on past events, allowing information to persist. The key idea behind RNN is to share parameters over time so that decisions can be made at each point in a sequence of events about what has happened so far in the sequence. In short, it ends up with a network that has a relatively simple repeating pattern, with part of the classifier connecting to the input at each time step and another part, called the recurrent connection, connecting you to the past at each step, as shown in the following Figure 1.27. On the downside, training RNNs can often be a challenging task due to their memory associated with the recurrent aspect (i.e. signals travel both forward and back and may contain loops, thus adding to their complexity).

LSTM is a type of RNN that, in addition to standards cells, also includes memory cells that can retain information for long periods of time. The enhanced architecture allows LSTMs to learn about long-term dependencies, which makes them smart at remembering things that have happened in the past and finding patterns across time.

Deep architectures are continuously evolving. Thus, the number of industrial applications in which DL is employed has grown steadily over the last decade. Many reported architectures have proven their superior ability in specific tasks, such as fault classification and fault prediction. An example of an architecture useful for fault diagnosis is shown in Figure 1.28.

It uses source domain-labelled datasets (such as vibration signals) to pre-train a CNN model, and a discriminator with two independent classifiers

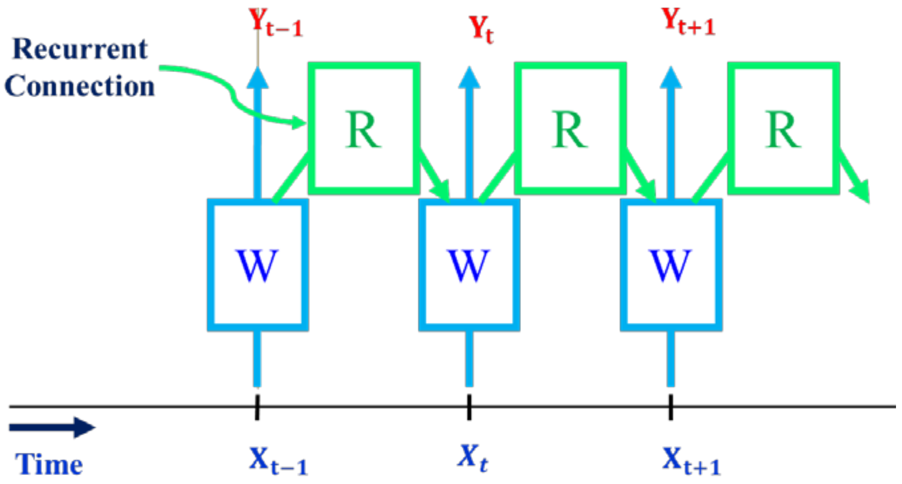

Figure 1.27 The repeating module underlying RNN architecture.

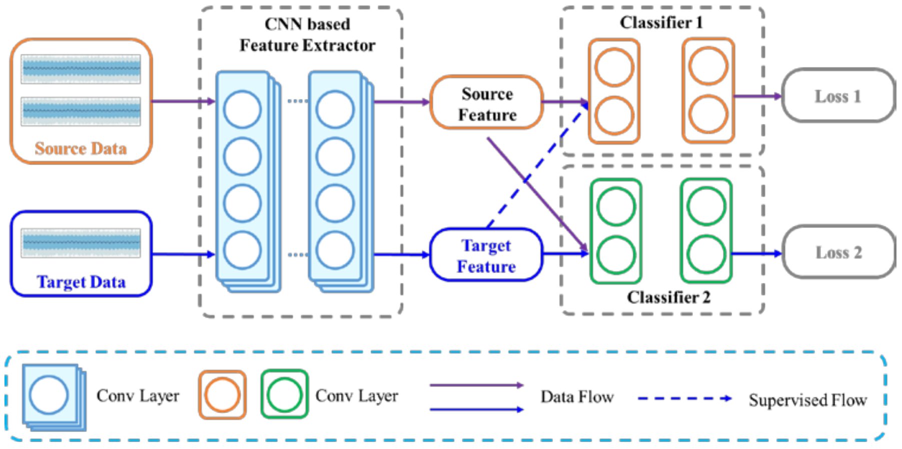

Figure 1.28 Example of an architecture useful for fault diagnosis.
Adapted from [28]

(fully connected layers) to optimize the CNN-based feature extractor parameters by minimizing distributions between the source and target domains.

1.8.3 Industrial Embedded AI/ML

Embedded AI is the application of AI at the embedded device level. While there are many examples of intelligent devices in the consumer space, embedded AI may have far higher potential in industrial applications. There are many contexts in which embedded AI may be very useful for collecting and understanding important phenomena in industrial settings, right where the sensors are located.

Embedded ML is the field of ML when applied to embedded systems such as microcontrollers. An embedded system is a combination of computer hardware and software, and additional parts, either mechanical or electronic, designed to perform a dedicated function.

The trend has been to connect the embedded devices via the Internet, collect the data and run the inference on servers in the cloud. However, according to the demand of the industry, the processing is moving from the cloud to the edge by using embedded ML, where lots of application can be designed having features of low cost, low power consumption, low bandwidth, secure and intelligent processing.

Embedded ML and DL techniques enable electronic systems to learn from real-time sensor data, audio and video and use the acquired knowledge to make standalone assessments, predictions, and decisions locally rather than in the cloud. Even more potential lies in combining real-time data from multiple sensors and thus deriving new types of information, leading to a continuous refinement and improvement of the ML/DL techniques. These techniques are applied on low power devices at the edge, hence the terms "edge ML and DL" are used interchangeably with "embedded ML and DL".

Edge AI refers to processing the data at the edge using AI methods and techniques, including ML and DL; however, edge AI has much more potential to accomplish edge intelligence than ML and DL alone. Edge AI equips sensor data with "the what" and "the how" to drive problem-solving processes, design, and development; hence, edge AI can be seen as the edge ML/DL of the future, encompassing architectures, frameworks, applications and edge intelligence and concepts, such as meta-learning and meta-intelligence.

The applications of embedded ML span many market segments and applications, for some of which the best pathways to development and deployment, such as time- and safety-critical applications, have yet to be found. The chapter seeks to cover a wide range of terms and concepts, not only with the aim of achieving a broader understanding of ML/DL applications but also to provide a valuable vantage point of where ML/DL are heading in the near future.

Many industrial applications target embedded ML and DL into edge devices, addressing the challenges and solving the problems posed by the gap between the advanced state-of-the-art models developed in and for the cloud and the limited capabilities of edge devices. The memory, processing, transmission and power consumption capabilities and limitations always depend on whether the device is micro-, deep-, or meta-edge device, but the challenges are the same. The AI/ML model needs to be converted into an efficient format, before compiling and flashing it into the device.

Benchmarking experiments are needed to demonstrate that state-of-the-art models with the right design and optimisation are compatible with the stringent resource requirements of edge devices and to suggest areas of improvement for the AI models.

Edge devices are typically single- and multi-core microcontrollers, with varying capabilities and limitations and unique identities. The edge can scale from a few devices to tens of thousands of devices distributed in different locations, so the devices are able to operate independently, with an unexplored evolution to training and inference actions. Although physically separated, the edge devices can be connected using wireless/wired connections in topologies such as mesh, with an unexplored potential for communication and distributed learning across devices inspired by recent advances in emergent intelligence.

ML model architectures can allow for highly interactive flows, starting with capturing the data straight from the embedded device all the way to production and deployment. This entails gathering sensor data directly from the products and environments and turning that data into useful data sets to be applied to ML algorithms and signal processing, instead of relying on predefined data sets. Furthermore, interactivity involves the verification, validation, and testing (VV&T) of algorithms, so that the most optimal solution given the device's capabilities and limitations is finally deployed.

The data, hardware/software platforms and more are the ingredients to design vertically integrated AI stacks, ensuring that edge AI is optimised for its hardware and its target application with optimised performance and efficiency.

The inference is performed on static models implemented on edge devices or other types of devices depending on the application. The inference requires many mathematical operations such as matrix multiplications and dot product operations and the processing run on a CPUs, GPUs, FPGAs, DSPs, ASICs depending on the processing power, energy efficiency, speed, and memory requirements.

Edge inference requires optimised hardware acceleration and when the process is connected to other performance-critical functions there is a need to provide interfaces by tightly coupling other accelerators or processing units into a common dynamic architecture.

1.8.4 On-device ML Applications Enabling True Edge Computing

The typical ML workflow takes advantage of several tools and frameworks, such as TensorFlow, TensorFlow Lite, and PyTorch, as shown in Figure 1.29. Some of them are optimised to run in very small footprints of memory and processing cycles, and thus can be employed in industrial embedded systems at the edge.

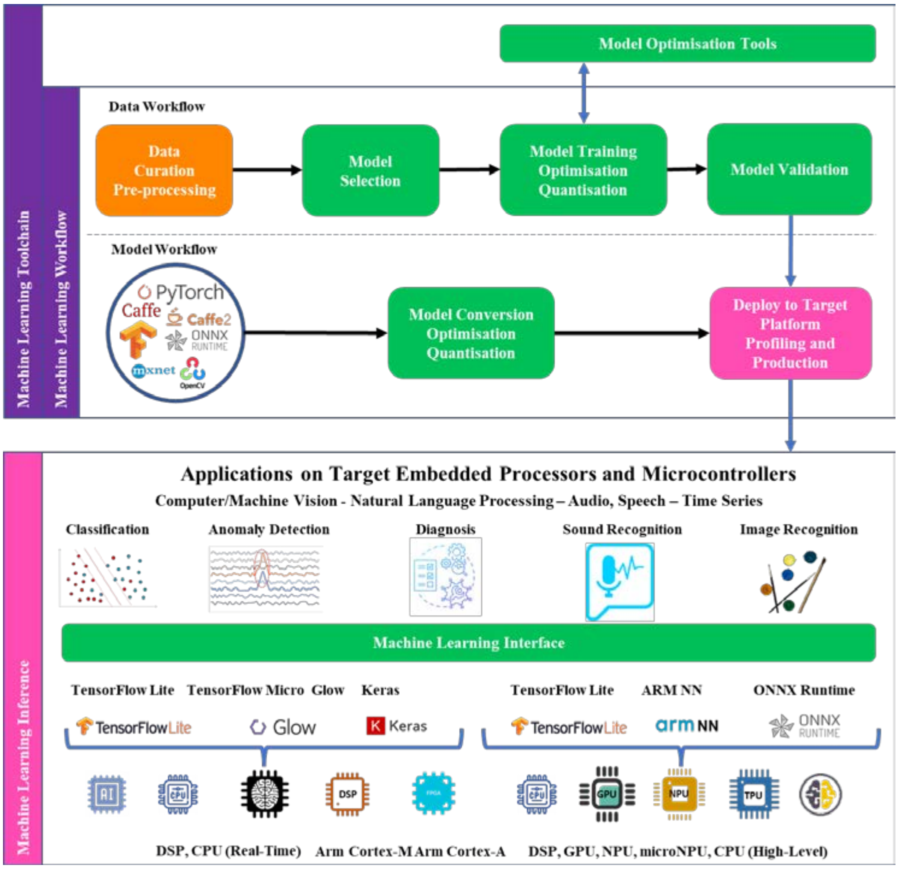
Machine Learning Toolchain
Machine Learning Workflow
Model Optimisation Tools
Data Workflow
Data Curation Pre-processing
Model Selection
Model Training Optimisation Quantisation
Model Validation
Model Workflow
PyTorch
Caffe
Caffe2
ONNX RUNTIME
mxnet
OpenCV
Model Conversion Optimisation Quantisation
Deploy to Target Platform Profiling and Production
Machine Learning Inference
Applications on Target Embedded Processors and Microcontrollers
Computer/Machine Vision - Natural Language Processing – Audio, Speech – Time Series
Classification
Anomaly Detection
Diagnosis
Sound Recognition
Image Recognition
Machine Learning Interface
TensorFlow Lite
TensorFlow Micro
Glow
Keras
TensorFlow Lite
ARM NN
ONNX Runtime
DSP
GPU
NPU
TPU
DSP, CPU (Real-Time)
Arm Cortex-M
Arm Cortex-A
DSP, GPU, NPU, microNPU, CPU (High-Level)

Figure 1.29 Embedded ML design and development ecosystem view.

Most industrial embedded systems can be loosely classified into three main categories:

- *Vibration and motion* include industrial systems with sensors that allow not only for the control of the device but also for its predictive maintenance.
- *Voice and sound* include industrial systems with microphones for voice keyword detection and speech recognition.
- *Vision* includes industrial systems recognizing objects to sort them or spot defects, or systems identifying, for example, faces to unlock devices.

The problems that may arise in industrial embedded systems worth investigating to solve with the help of ML are many and multi-folded, but three general main categories can be identified:

- Detecting anomalies in the operation of edge devices before something breaks because industrial equipment can be expensive to produce and even costlier to repair or replace.
- Classifying things, behaviour, or objects from any variety and combination of sensors, either internal or external to the edge device.
- Forecasting, such as what the signal will look like in the near or far future, based on historical data.

All the potential use cases will have different workload performance and scalability requirements, depending on the application:

- For the prediction and maintenance of machines, it is essential to predict and give feedback on their health status as early as possible to avoid instant shutdown.
- For security systems, it is essential to implement features such as facial and voice recognition on edge devices to ensure they effectively contribute to providing security, through their use with security locks for home, offices, vehicles, and so forth.
- For autonomous vehicles, it is important that the devices installed on the car analyse local surroundings to recognize traffic lights, pedestrian roads, and people to make smart decisions.
- For surveillance and monitoring, it is crucial that any suspicious activities are monitored on edge devices and in real time by, for instance, recognizing human movements.
- For robots and robotic things, it is essential to make decisions independently without the need to connect to the internet.

While ML can be used to arrive at innovative solutions, it is important to note that embedding AI on the edge has limitations and that ML alone cannot always solve complex problems. Many industrial applications require other technologies to work in tandem with ML to achieve effective, low-power solutions to be deployed close to the sensor, thus enabling true edge computing.

More in-depth insights into use cases implementing industrial AI applications at the edge and the transition to Industry 5.0 can be found in [36].

1.8.5 Machine Learning on Embedded Devices

Most AI frameworks have been developed for desktops, servers, and laptops with large resources. By contrast, embedded edge AI frameworks run on smaller but efficient devices, such as single-board computers and microcontrollers. Single-board computers usually have a powerful microprocessor with a separate memory, can run a full operating system, and can provide a full-user interface; hence, they can adapt ML algorithms (such as Scikit-learn, TensorFlow, PyTorch, Keras, and Caffe) that use high-level programming languages such as Python, provided that they have enough power to fulfil the task effectively and efficiently.

The situation is rather different for microcontrollers, which are usually less expensive and require much less power, with only a few buttons or a simple LCD screen of the user interface. Hence, the adaption of the existing AI frameworks to run on microcontrollers has started to show results only recently.

Software platforms

TensorFlow Lite was the first AI software framework specifically designed for micro-controllers that allows running simple NNs without manually programming the matrix operations and with only a few kilobytes of memory. Since it was introduced, many AI software tools have been developed to address the different requirements for designing and implementing ML on edge devices. However, it was the optimisation of both hardware and software in tandem that allowed for the use of more complex ML algorithms in microcontrollers, which led to industries embracing the application of embedded ML.

Optimisation can be multi fold: enable more complex models to be deployed, meet real time latency constraints, extend the battery life of edge

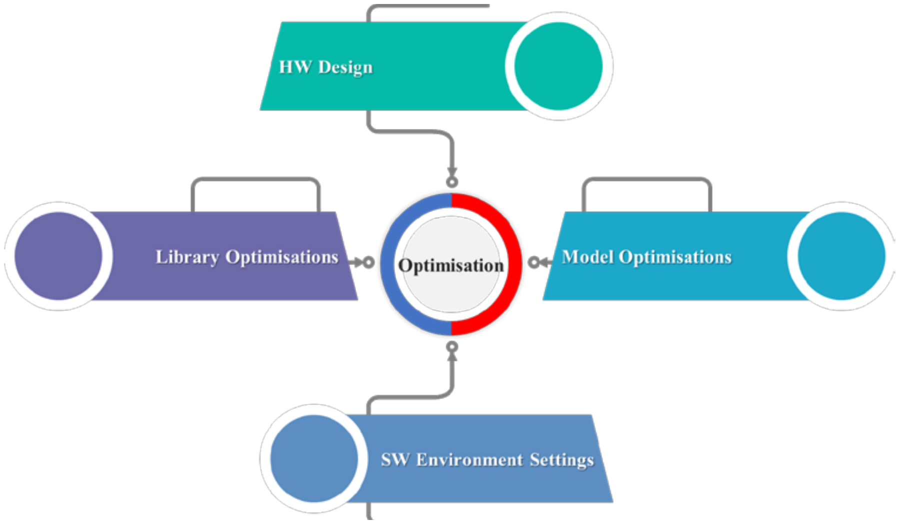

Figure 1.30 Embedded ML optimisation.

devices. The important point is that even the smallest optimisation anywhere in the system can make a difference, be it in hardware, software algorithms, framework, libraries, as shown in Figure 1.30.

Hardware platforms and hardware-software co-design for ML

Embedded edge AI can be defined from the perspective of both hardware and software, depending on whose capabilities are focused on. From the hardware perspective, embedded edge AI is defined as the capability of low-power, resource-constrained devices such as sensors and actuators to execute AI algorithms. From the software perspective, embedded edge AI is defined as the capability of AI algorithms to adapt and run effectively and efficiently on devices with limited resources.

The ability to embed AI in low-end devices is highly dependent on the availability of automated frameworks with easy-to-use design flows that can generate optimised AI models for the hardware targets. Thus, all hardware components (microcontroller, communication- modules, sensors, actuators, etc.) are part of the design flow. Hence, regardless of whether embedded edge AI is defined from the hardware or software perspective, a hardware-software co-design is key to embedding AI in edge devices.

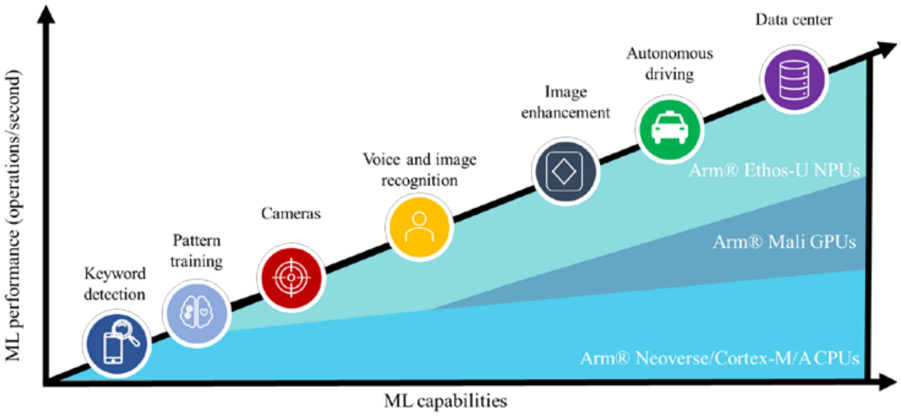

Figure 1.31 ML hardware options for various AI tasks.

Embedded edge ML has changed the way microprocessors and microcontrollers are used. AI can be embedded by augmenting development boards with components such as sensors and additional chips, all geared towards executing AI programs spanning from simple ML algorithms to resource intensive deep NNs.

Until recently, most of the chips developed only supported a subset of functions used in modern deep NNs, imposed by the memory restrictions and computing capabilities of the hardware; not even specialised hardware could execute deep NNs.

With the recent advances in hardware, developments have been directed towards integrating AI and deep NNs directly into sensor hardware. NNs targeting constrained devices are more efficient in terms of memory footprint and inference time. Techniques such as quantisation are used to reduce computing precision with no significant decrease in algorithm accuracy.

When designing hardware, special attention must be paid to the three main classes of AI-related building blocks, namely memory, storage, and logic. Memory is used for short-term storage during processing and consists of dynamic random-access memory (DRAM). Storage represents the long-term repository of large electronic data sets and consists of NAND flash memory. Logic is used for processing, computing, and optimising the calculation of NN operations or other specific AI functions and consisting of CPUs, GPUs, FPGAs, different custom ASICs, etc.

The edge processing units under development must have several characteristics such as a heterogeneous computing architecture (e.g., CPU, GPU, ASIC, FPGA, neuromorphic, etc.), support for the main AI edge frameworks (e.g., TensorFlow, Caffe, Keras, etc.), multi-modality, end-to-end embedded security, and high energy efficiency.

Accelerators and Neuromorphic hardware

Accelerators and neuromorphic hardware are both represented as sub-layers of the hardware layer, which is at the foundation of the technology stack. Employing both generic and hardware-specific optimisations can lead to a significant decrease in the memory footprint of NNs and accelerate inference latency.

Hardware accelerators are specialised hardware components within the system that enable greater efficiency when running certain computing tasks than is possible with software running on a general-purpose CPU alone. A wide variety of dedicated hardware acceleration systems exist, and the most common hardware used for acceleration include GPU, ASIC, FPGA.

Neuromorphic computing is a new computing technology that reproduces human brain activity with models of selective spiking ensembles of neurons in models that reproduce biological reactions.

Neuromorphic computers—as opposed to Von Neumann computers, which are composed of separate CPUs and memory units—are inspired by the human brain and are composed of neurons and synapses governing both processing and memory. Programs in neuromorphic processing units are determined by the structure of the neural network and its parameters instead of explicit instructions, as in a von Neumann computer. Neuromorphic computers receive spikes as input that can be used to encode numerical information, as opposed to Von Neumann computers that encode numerical values represented by binary values [34]. This is intuitively illustrated in Figure 1.32.

Consequently, neuromorphic computers present some essential operational differences: they are highly parallel, meaning that, in principle, all neurons and synapses can operate simultaneously. Both neurons and synapses perform processing and store values, resulting in no separation between processing and memory. In addition, increasing the number of neurons and synapses can be done easily; thus, neuromorphic computers are highly scalable. Neurons and synapses ‘spike’ only when there are spikes to process, making them “event-driven”.

Most of the work in neuromorphic computing has focused on hardware development. A neuromorphic chip can contain thousands of neurons, with

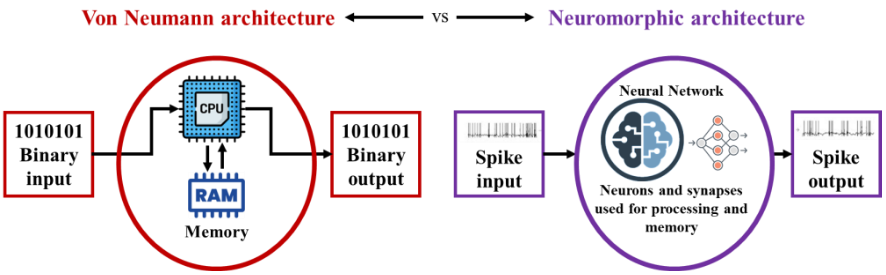

Figure 1.32 Comparison of the von Neumann architecture with the neuromorphic architecture.

their synapses, dendrites and axons reproducing human brain activity. However, neuromorphic computing requires both hardware and software, and to be widely adopted by industry in the future, neuromorphic algorithms and applications must catch up with technological advances in hardware. Spiking Neural Networks (SNN), which mimics the energy-efficient signal system in the brain, has drawn much recent attention. The main difference between SNNs and traditional networks is that neurons in SNNs accumulate charge from the environment or from other neurons over time; thus, time is a new element in their operation. Algorithms that have been successful for deep-learning applications will need to be adapted to work on SNNs [34].

1.8.6 Embedded ML Development Flow in Industrial Setting

It is important to emphasise that the embedded edge ML flow and its associated processes are different from most typical ML flows. Many applications deal with static ML flows. A ML flow is static when there are no time variables in the equation. Hence, the static model is trained offline exactly once, and then the trained model is used for inference for some time, at least until an update is required. Moreover, many pre-built data sets are available for various domains and applications that ML practitioners can use as a start.

By contrast, most industrial applications must cope with time series problems and thus deal with data continuously entering the system over time. Pre-built data sets are not configured for use with smaller ML applications such as those intended for microcontrollers. In edge embedded systems, data are not extracted from data stores such as files or databases but rather are acquired directly from sensors. Thus, inference occurs in real time, and in many cases, so does training. The timeline can be short (seconds or

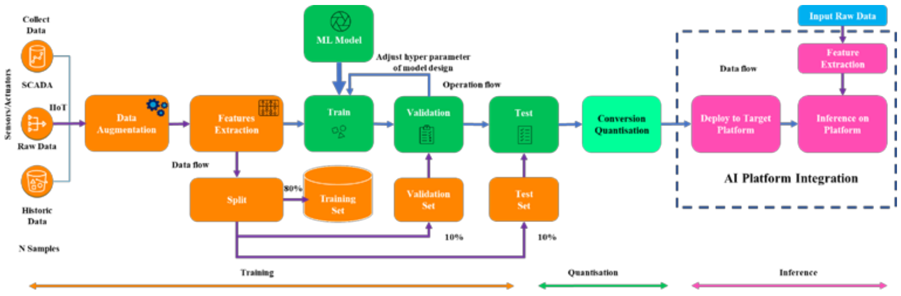

Figure 1.33 The high-level embedded ML development flow.

minutes) or long (days or months). Owing to the dynamic aspect, re-training is necessary.

Figure 1.33 illustrates a typical embedded ML development flow. In short, the flow starts with the collection of signals. Continuous raw data are sliced into smaller windows and processed into extract features. The trained model is then deployed on the IIoT device and used to run inferences, whose result, depending on the application, can be a prediction, a class detected, or an anomaly detected. Pre-processing steps such as cleaning or filtering data may be necessary to obtain a representative data set for the application and make it easier to process.

In the following paragraphs the basic steps of the embedded ML design flow are described, with examples from a generic use case, i.e., classification of the state of a motor based on the vibration measurements using an accelerometer sensor from an IIoT device. The motor is operating at fixed speeds, which are divided into several classes based on various percentages of the maximum speed.

The **data collection** process is essential, as good results are dependent of qualified data for the training and can require considerable effort and expertise to design the correct signal acquisition and sampling methodology suitable for a particular application.

The signals for each of the classes can be acquired straight from the device. The continuous raw data are usually sliced into smaller windows whose size can be configured with parameters. From a three-axis accelerometer sensor and with a buffer size of 256 samples on each axis, a total of 768 values are produced per signal. With a sampling frequency of 833 Hz, each buffer represents a snapshot of approximately 300 milliseconds of the accelerometer temporal vibration data. The number of signals and the split

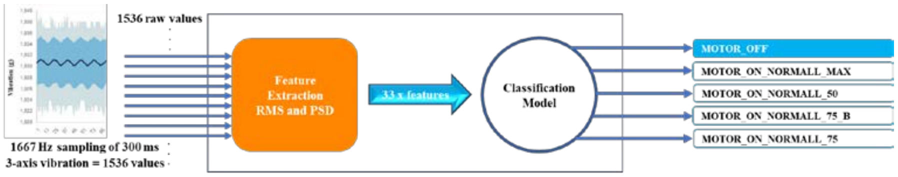

Figure 1.34 Temporal and frequency plots as input to motor classification.

between training and validation data can also be configured (usually 80% training, 20%).

The vibration signals collected can be visualised as shown in Figure 1.34, in both temporal and frequency plots for each of the classes.

One common **pre-processing** technique when examining vibration or motion data to identify features is to take the Fourier transform of the data to obtain information about them in the frequency domain and break the signal into its various frequency components. By providing filtering, only the frequencies that represent the characteristics of the motor vibration are kept, and the rest are attenuated.

A feature is an individual measurable property or characteristic of a phenomenon being observed and deciding what features to select is an important task. Poor features will have negative impacts. For example, if the feature only takes one snapshot in time, it is a poor feature because it provides no information about how the signal changes in time.

Extracting the features to be fed into the AI model for training and, ultimately, inference from large inputs can be performed automatically by many AI frameworks. In a matter of seconds or minutes, all raw samples are converted into sets of features.

A useful aspect of this automation is the possibility to visualise and explore the features. In the case of a classifier, features that are visually clustered are a good sign that the model can be trained to do the same. On the contrary, if features overlap in various degrees and are intertwined, it is very likely the trained model will have difficulties in differentiating between classes. This problem can be solved in various ways such as increasing the buffer size, that is, prolonging the sampling signal, to better capture signal patterns or even changing some of the features.

The **training** process employs back-propagation algorithms to configure and update the parameters inside the model that can improve the chances of predicting each feature set. Parameters are usually configured automatically.

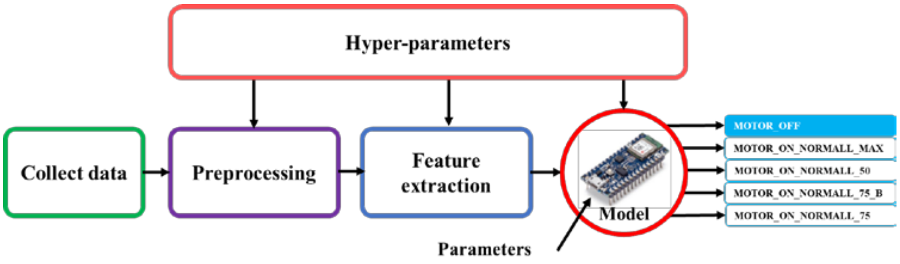

Figure 1.35 Hyperparameters (outside the model) vs parameters (inside the model).

In contrast to the model parameters, hyper-parameters cannot be tuned by the data and lie outside the model (Figure 1.35). These are values that must be set manually, such as the size and shape of the model, the learning rate, and the number of training steps to take, the features to use, and the methods and calculations to pre-process the data.

The **model validation** datasets and test datasets are not part of the training datasets. The validation data set can be used to analyse how well the model performs against unseen data and to adjust identified problems prior to using the test set (Figure 1.36).

Two common issues in ML are when the model underfits or overfits the input data. The former is when the model performs poorly on training and validation data, whereas the latter is when the model performs better on the training data than it does with the validation or test data.

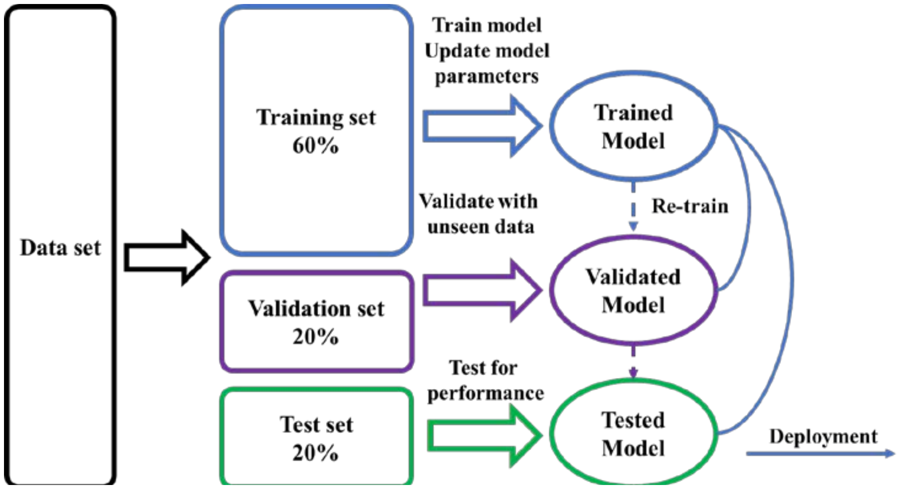

Figure 1.36 Categories of datasets and where they are used.

The solutions to these issues present particularities in the case of embedded edge ML, but, in short, collecting more signals, selecting different features, extending the training time and increasing model complexity will usually work for underfitting, while gathering more data, training for a shorter length of time and reducing model complexity and adding dropout layers will work for overfitting.

Understanding NN architecture is essential to explore how increasing or reducing model complexity affects model accuracy. A neural network architecture can be optimised by several means (adding more layers to deeper the model or increasing the number of hidden units to wider the model, changing the activation, and optimization functions, learning rate, fitting more data, and more), and knowing what and how to optimise it is a matter of experimentation. Fortunately, most platforms can automatically tune hyperparameters.

Much of creating a better model is trial and error: gathering more data or adjusting the hyperparameters and re-training your model to see if it improves the per-class accuracy. Or sometimes, there may not be enough or the right kind of data to train a good model.

One of the most useful evaluation tools is the confusion matrix of the validation data (Figure 1.37).

The predicted labels are on the x-axis and the true labels on the y-axis. The diagonal elements are the number of points for which the predicted label is equal to the true label, while off-diagonal elements are those that are mislabelled. The higher the diagonal values of the confusion matrix the better. The matrix is a good way to visually interpret how well the model is doing at its predictions and understand where it may need improvement.

In the context of micro-edge embedded systems, the **deployment** is dependent on the hardware/software platform and is more or less automated, and in essence comprises of three steps: the first is a format conversion of the fully trained model, then a weight/model compression to reduce the amount of memory to store the weights in the target hardware platform and to simplify the computation so it can run efficiently on target processors. This step is usually to quantize, i.e., converts all parameters from floating point values to integers. Finally, the last step is compiling the model and generating the code to be integrated with the MCUs firmware. The implementation of these steps must follow the back-end flow specific to the target. The optimisation challenge is to save as much memory as possible in the processor or microcontroller, with as little reduction in the accuracy of the model as possible.

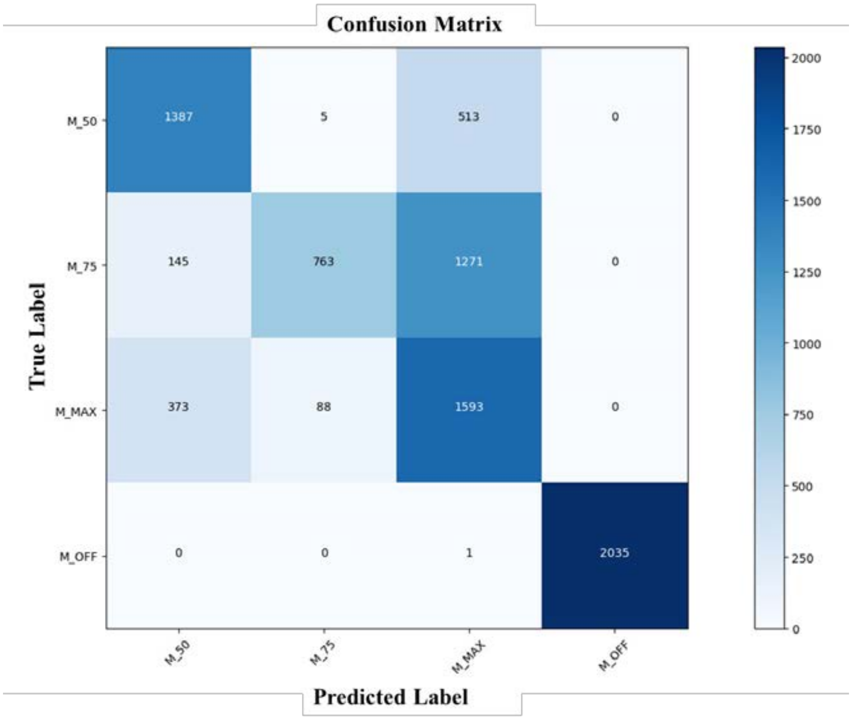

Figure 1.37 Confusion matrix.

Inference is the process of using live unseen data with a fully trained model to make predictions. The inference and the output will look different depending on the actual target device and production environment, but in essence, it happens in three steps.

First, the input signal is sampled for a period of time sufficient to capture the essence of the signal patterns before sending the raw accelerometer data to the library for inference. With 833 Hz sampling rate and 300 milliseconds time length, the buffer size will be 256 samples producing in total 768 values. The library expects these values to be stored in an array containing raw sensor values. Next, features are extracted, and finally, inference is performed, with the inference function returning the predicted probabilities, each corresponding to one of the classes. The highest probability will indicate the correct class, but threshold comparison and other algorithms can be used. A minimum threshold can also be considered. This process loops indefinitely. The state machine usually consists of two states with two functions "init" and "inferencing", respectively, with the former initializing the NN model and the latter being a continuously running function for collecting raw data from the sensors on board and making predictions in real-time. While feature extraction and inference are performed, the buffer fills up with raw sensor data in the background. More about applications that benefit from inference at the edge can be found in [10].

To conclude the discussion on the Hardware/Software technology stack, machine learning and neural networks can now be efficiently deployed on resource-constrained devices, which allow for cost-efficient deployment, widespread availability, and the preservation of sensitive data. However, the trade-offs that optimisation methods, software frameworks and hardware architecture have on key performance metrics, such as inference latency and energy consumption, have yet to be studied in depth.

1.9 Summary

Industrial AI and IoT/IIoT are enablers for building the foundations for digital transformation and business innovation. Full-scale and full-stack industrial AI technology accelerates digital innovation across industries and therefore boosts productivity. The adoption of AI helps industries climb the value chain and drive innovation, thus providing new paths to growth for manufacturing, service, and other industries.

In this context, managing the end-to-end (E2E) AI technologies connected with the IoT/IIoT, supervisory control and data acquisition (SCADA), and edge computing, is crucial for various industrial sectors. Addressing the developments in silicon-born AI that enable and generate AI-born embedded and industrial systems accelerates harnessing the silicon and embedded systems designed specifically for AI, thus supporting E2E solutions and advancing the adoption of AI technologies across industrial sectors.

Contributions to this chapter come from a diverse number of disciplines and communities and cover related technologies across different layers in the AI technology stack.

As result, the chapter provides an overview of the main concepts and terminology related to industrial embedded edge AI technologies.

The shifting of AI methodologies from operating in the cloud to operating on the edge as a fundamental approach for future developments on digitising industries marks the beginning of a widespread transition in the control of industrial processes and the functionality of devices. AI methodologies operating on the edge must drive the major milestones of this transition on any roadmap.

Embedded edge AI platforms, training and learning, and applications form the foundation that supports the development of edge AI applications.

AI-optimised hardware provides the core infrastructure for embedded edge AI applications. It includes AI chips (neuromorphic, CPUs, GPUs, FPGAs, ASICs), large-capacity, low-latency, and all-flash arrays, and solid-state storage devices; high-performance, high-throughput, and highly scalable edge servers and network equipment. Turning data into descriptive, diagnostic, and predictive analytic insights requires visualised modelling and code testing environments, as well as ML and DL edge platforms configured for general AI applications or real-time embedded environments.

Edge AI technologies and applications require advanced industrial enterprise high-level architecture as a reference for implementing embedded edge AI technologies in an environment that can manage the large-scale deployment of AI applications.

The infrastructure layer requires edge computing and modular processing units integrated with on-premises platforms. In the industrial platforms and application layers, the analytics and flexible service capabilities of edge must support the integration of industrial enterprise applications with various industrial AI applications.

As AI matures, AI technological development often intersects other technological areas.

The chapter introduced an overview of AI concepts, including definitions to establish a common vocabulary for the stakeholders involved and for the presentation of E2E industrial embedded edge AI technologies across the technology stack, application, and industrial sectors. The chapter can thus serve as a reference for various partners and stakeholders to help reach the full potential of edge AI for digitising industry by introducing developments in silicon-born AI to enable and generate AI-born embedded and industrial systems and accelerate the adoption of edge AI technologies across various industrial sectors.

Industrial edge AI technologies differ from consumer AI technologies that provide citizens with direct technology exposure, so industrial AI solutions may lack direct consumer scrutiny. Nevertheless, societal perception has an impact on how unions perceive the introduction of edge AI technologies, how management decisions on investment are made and how policymakers decide upon regulations.

Acknowledgements

This work is conducted under the framework of the ECSEL AI4DI "Artificial Intelligence for Digitising Industry" project. The project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 826060. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Czech Republic, Italy, Latvia, Belgium, Lithuania, France, Greece, Finland, Norway.

References

[1] Research and Markets. "Smart Manufacturing Market by Technology (Robotics, AI, IIoT, Cloud, AR/VR), Application (Machine Inspection; Energy, Quality, and Warehouse Management; Planning, Surveillance, Optimization), End-use Industry, and Geography - Global Forecast to 2029", June 2022. Available online at: https://www.researchandmarkets.com/

[2] Vantage Market Research. "AI in Manufacturing Market Size, Share & Trends Analysis Report by Offering (Hardware, Software, Services), by Technology (Machine Learning, Natural Language Processing, Context-aware Computing, Computer Vision), by Application (Predictive Maintenance and Machinery Inspection, Material Movement, Production Planning, Field Services), by Industry (Automobile, Energy and Power, Pharmaceuticals, Heavy Metals and Machine Manufacturing), by Region (North America, Europe, Asia Pacific, Latin America and Middle East & Africa) - Global Industry Assessment (2016 - 2021) & Forecast (2022 - 2028)". Available online at: https://www.globenewswire.com/

[3] AI4DI, Artificial Intelligence for Digitising Industry. Available online at: https://ai4di.eu/

[4] P. H. Winston. Artificial Intelligence. Third Edition, Addison-Wesley Publishing Company, 1992.

[5] D. B. Fogel, "Defining Artificial Intelligence". In Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Third Edition, The Institute of Electrical and Electronics Engineers, Inc., IEEE Press, pp. 1-32, 2006.

[6] S. Legg, and M. Hutter, "Universal Intelligence: A Definition of Machine Intelligence. Minds and Machines", 17(4):391-444, Springer, 2007. Available online at: https://arxiv.org/abs/0712.3329

[7] J. McCarthy, “What Is Artificial Intelligence, Basic Questions”, Stanford Formal Reasoning Group, 2007.

[8] S. J. Russell, and P. Norvig, Artificial Intelligence: A Modern Approach, Fourth Edition. Prentice Hall, 2022.

[9] J. R. Searle, Mind, language and society, New York, NY: Basic Books, ISBN 978-0-465-04521-1, 1999.

[10] What is AI Inference at the Edge? Available online at: https://www.steatite-embedded.co.uk/what-is-ai-inference-at-the-edge/

[11] O. Vermesan, J. Bacquet, (Editors). Next Generation Internet of Things - Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation. ISBN: 978-87-7022-008-8 (Hardback), 978-887-7022-007-1 (Ebook). River Publishers, 2018.

[12] R. Schwartz, J. Dodge, N. A. Smith, O. Etzioni, (2019). “Green AI”. Available online at: https://arxiv.org/pdf/1907.10597.pdf

[13] Buchanan, B.G., Shortliffe, E.H. (eds.). Rule-Based Expert Systems - The MYCIN Experiments of the Stanford Heuristic Programming Project. Addison-Wesley Publishing Company (1984)

[14] J. McDermott. R1: an Expert in the Computer Systems Domain. Proceedings of the National Conference on Artificial Intelligence (AAAI), pp. 269-271 1980.

[15] R. Reiter. A logic for default reasoning. Artificial Intelligence 13(1–2), 1980.

[16] X. Yang, Z. Song, I. King and Z. Xu. “A Survey on Deep Semi-supervised Learning”. https://doi.org/10.48550/arXiv.2103.00550

[17] R. Reiter, (1987). A theory of diagnosis from first principles. Artificial Intelligence, 32(1), 57–95.

[18] J. De Kleer, A. K. Mackworth, and R. Reiter, (1992). Characterizing diagnosis and systems. Artificial Intelligence, 56.

[19] T. Eiter, G. Ianni, T. Krennwallner, “Answer Set Programming: A Primer”, pp. 40–110. Springer Berlin Heidelberg, Berlin, Heidelberg (2009). https://doi.org/10.1007/978-3-642-03754-2_2

[20] F. Wotawa, “Reasoning from first principles for self-adaptive and autonomous systems”. Springer (2019). https://doi.org/10.1007/978-3-030-05645-2

[21] A. Choi, R. Wang, A. Darwiche, “On the relative expressiveness of Bayesian and neural networks”. *Int. J. Approx*. Reason. 113: 303-323 (2019)

[22] W. Shi, A. Shih, A. Darwiche, A. Choi, “On Tractable Representations of Binary Neural Networks”. CoRR abs/2004.02082 (2020)

[23] D. Foster. Generative Deep Learning. (Kindle Locations 242-243). O'Reilly Media. Kindle Edition.

[24] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "Ganomaly: Semi-supervised anomaly detection via adversarial training," in *Asian Conference on Computer Vision*, pp. 622–637. Springer, 2018

[25] Biomimicry 3.8 https://biomimicry.net/resource-handbook/

[26] A. M. Turing, (October 1950). "Computing machinery and intelligence". Mind. LIX (238): 433–460. https://academic.oup.com/mind/article/LIX/236/433/986238?login=false

[27] E. B. Baum, D. Boneh and C. Garrett, "On genetic algorithms." COLT '95 (1995).

[28] Y. Ran, X. Zhou, P. Lin, Y. Wen and R. Deng, "A Survey of Predictive Maintenance: Systems, Purposes and Approaches", *IEEE Communications Surveys and Tutorials*, Nov. 2019.

[29] M. Ghallab, D. Nau, and P. Traverso. Automated planning and acting. Cambridge University Press, 2016.

[30] R. K. Lindsay, B. G. Buchanan, E. A. Feigenbaum, and J. Lederberg. Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project. New York: McGraw-Hill

[31] C. Sabo, K. Cohen, "Fuzzy logic unmanned air vehicle motion planning", Advances in Fuzzy Systems Volume January 2012 Article No.: 13, pp 13. https://doi.org/10.1155/2012/989051

[32] M. Xu, W. C. Ng, W. Yang, B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. S. Shen, C. Miao. (2022). A Full Dive into Realizing the Edge-enabled Metaverse: Visions, Enabling Technologies, and Challenges. https://doi.org/10.48550/arXiv.2203.05471

[33] M. Bojarski, B. Firner, B. Flepp, L. Jackel, U. Muller, K. Zieba and D. Del Testa, "End-to-End Deep Learning for Self-Driving Cars". https://developer.nvidia.com/blog/deep-learning-self-driving-cars/

[34] C. D. Schuman, S. R. Kulkarni, M. Parsa, et al. "Opportunities for neuromorphic computing algorithms and applications". *Nat Comput Sci 2*, 10–19 (2022). https://doi.org/10.1038/s43588-021-00184-y

[35] J. Popper, J. Hermann, K. Cui, et al. (2018). Artificial intelligence across industries - IEC Whitepaper.

[36] O. Vermesan, J. Reiner, C. De Luca, M. Coppola (Eds). Artificial Intelligence for Digitising Industry Applications. ISBN: 9788770226646, River Publishers, 2022.

[37] The German Artificial Intelligence (AI) Standardization Roadmap, 2020, https://www.din.de/resource/blob/772610/e96c34dd6b12900ea75b460538805349/normungsroadmap-en-data.pdf