1 Introduction to Intelligent Systems

In which we consider what it means to be intelligent and whether machines could be such a thing.

1.1 Intelligent Machines, or What Machines Can Do

Philosophers have been trying for over 2000 years to understand and resolve two big questions of the universe: how does a human mind work, and can non-humans have minds? However, these questions are still unanswered.

Some philosophers have picked up the computational approach originated by computer scientists and accepted the idea that machines can do everything that humans can do. Others have openly opposed this idea, claiming that such highly sophisticated behaviour as love, creative discovery and moral choice will always be beyond the scope of any machine.

The nature of philosophy allows for disagreements to remain unresolved. In fact, engineers and scientists have already built machines that we can call 'intelligent'. So what does the word 'intelligence' mean? Let us look at a dictionary definition.

1 Someone's **intelligence** is their ability to understand and learn things.
2 **Intelligence** is the ability to think and understand instead of doing things by instinct or automatically.

(*Essential English Dictionary*, Collins, London, 2020)

Thus, according to the first definition, intelligence is a quality possessed by humans. But the second definition suggests a completely different approach and gives some flexibility; it does not specify whether it is some**one** or some**thing** that has the ability to think and understand. Now we should discover what thinking means. Let us consult our dictionary again.

Thinking is the activity of using your brain to consider a problem or to create an idea.

(*Essential English Dictionary*, Collins, London, 2020)

So, in order to think, some**one** or some**thing** has to have a brain or, in other words, an organ that enables some**one** or some**thing** to learn and understand things, to solve problems and to make decisions. So we can define intelligence as 'the ability to learn and understand, to solve problems and to make decisions'.

The very question that asks whether computers can be intelligent, or whether machines can think, came to us from the 'Dark Ages' of artificial intelligence (during the late 1940s and early 1950s). The goal of **artificial intelligence** (AI) as a science is to make machines do things that would require intelligence if done by humans (Boden, 1977). Therefore, the answer to the question 'Can machines think?' was vitally important to the discipline. However, the answer is not a simple 'Yes' or 'No', but rather a vague or fuzzy one. Your everyday experience and common sense would have told you that. Some people are smarter in some ways than others. Sometimes we make very intelligent decisions but sometimes we also make very silly mistakes. Some of us deal with complex mathematical and engineering problems but are moronic in philosophy and history. Some people are good at making money, while others are better at spending it. As humans, we all have the ability to learn and understand, to solve problems and to make decisions; however, our abilities are not equal and lie in different areas. Therefore, we should expect that if machines can think, some of them might be smarter than others in some ways.

One of the earliest and most significant papers on machine intelligence, 'Computing machinery and intelligence', was written by the British mathematician Alan Turing over 70 years ago (Turing, 1950). However, it has stood up well to the test of time, and Turing's approach remains universal.

Alan Turing began his scientific career in the early 1930s by rediscovering the Central Limit Theorem. In 1937 he wrote a paper on computable numbers, in which he proposed the concept of a universal machine. Later, during the Second World War, he was a key player in deciphering Enigma, the German military encoding machine. After the war, Turing designed the 'Automatic Computing Engine'. He also wrote the first program capable of playing a complete chess game; it was later implemented on the Manchester University computer. Turing's theoretical concept of the universal computer and his practical experience in building code-breaking systems equipped him to approach the key fundamental question of AI. He asked: Is there thought without experience? Is there mind without communication? Is there language without living? Is there intelligence without life? All these questions, as you can see, are just variations on the fundamental question of AI: Can machines think?

Turing did not provide definitions of machines and thinking. Instead, he avoided semantic arguments by inventing a game, **Turing's imitation game**. Instead of asking, 'Can machines think?', Turing said we should ask, 'Can machines pass a behaviour test for intelligence?' He predicted that by the year 2000, a computer could be programmed to have a conversation with a human interrogator for five minutes and would have a 30 per cent chance of deceiving the interrogator that it was a human. Turing defined the intelligent behaviour of a computer as the ability to achieve human-level performance in cognitive tasks. In other words, a computer passes the test if interrogators cannot distinguish the machine from a human on the basis of the answers to their questions.

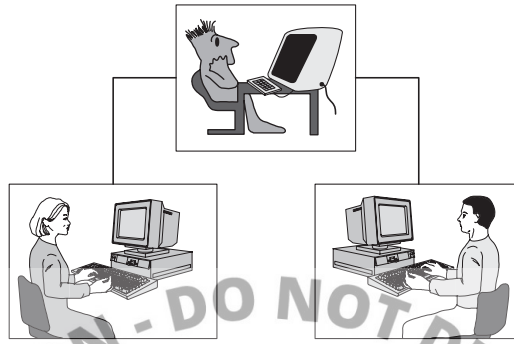

Figure 1.1 Turing's imitation game: phase 1

The imitation game proposed by Turing originally included two phases. In the first phase, shown in Figure 1.1, the interrogator, a man and a woman are each placed in separate rooms and can communicate only via a neutral medium such as a remote terminal. The interrogator's objective is to work out who is the man and who is the woman by questioning them. The rules of the game are that the man should attempt to deceive the interrogator that *he* is the woman, while the woman has to convince the interrogator that *she* is the woman.

In the second phase of the game, shown in Figure 1.2, the man is replaced by a computer programmed to deceive the interrogator as the man did. It would even be programmed to make mistakes and provide fuzzy answers in the way a human would. If the computer can fool the interrogator as often as the man did, we may say this computer has passed the intelligent behaviour test.

Physical simulation of a human is not important for intelligence. Hence, in the Turing test the interrogator does not see, touch or hear the computer and is therefore not influenced by its appearance or voice. However, the interrogator is allowed to ask any questions, even provocative ones, in order to identify the machine. The interrogator may, for example, ask both the human and the machine to perform complex mathematical calculations, expecting that the computer will provide a correct solution and will do it faster than the human. Thus, the computer will need to know when to make a mistake and when to delay its answer. The interrogator also may attempt to discover the emotional

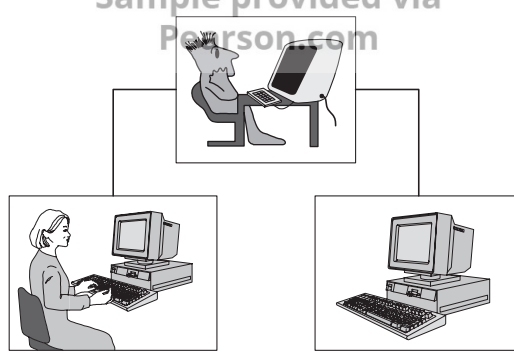

Figure 1.2 Turing's imitation game: phase 2

nature of the human, and, thus, he might ask both subjects to examine a short novel or poem or even painting. Obviously, the computer will be required here to simulate a human's emotional understanding of the work.

The Turing test has two remarkable qualities that make it universal:

- By maintaining communication between the human and the machine via terminals, the test gives us an objective standard view on intelligence. It avoids debates over the human nature of intelligence and eliminates any bias in favour of humans.
- The test itself is quite independent from the details of the experiment. It can be conducted either as a two-phase game as just described, or even as a single-phase game in which the interrogator needs to choose between the human and the machine from the beginning of the test. The interrogator is also free to ask any question in any field and can concentrate solely on the content of the answers provided.

Turing believed that by the end of the 20th century it would be possible to program a digital computer to play the imitation game.

Can modern computers pass the Turing test?

There have been remarkable advancements in natural language processing and AI; however, achieving a machine that can consistently and convincingly simulate human conversation across a wide range of topics and contexts remains a challenging task. Until now, no computer system has definitively passed the Turing test.

Although modern computers still cannot pass the Turing test, it provides a basis for the verification and validation of knowledge-based systems. A program thought intelligent in some narrow area of expertise is evaluated by comparing its performance with the performance of a human expert.

Our brain stores the equivalent of over 10^{18} bits and can process information at the equivalent of about 10^{15} bits per second. By 2050, the brain will probably be modelled by a chip the size of a sugar cube – and perhaps by then there will be a computer that can play – even win – the Turing imitation game. However, do we really want a machine to perform mathematical calculations as slowly and inaccurately as humans do? From a practical point of view, an intelligent machine should help humans to make decisions, to search for information, to control complex objects and finally to understand the meaning of words. There is probably no point in trying to achieve the abstract and elusive goal of developing machines with human-like intelligence. To build an intelligent computer system, we have to capture, organise and use human knowledge in some area of expertise.

1.2 The History of Artificial Intelligence, or from the 'Dark Ages' to Deep Learning

Artificial intelligence as a science was founded by four generations of researchers. Some of the most important events and contributors from each generation are described next.

1.2.1 The 'Dark Ages', or the birth of artificial intelligence (1943–1956)

The first work recognised in the field of AI was presented by Warren McCulloch and Walter Pitts in 1943. McCulloch had degrees in philosophy and medicine from Columbia University and became the Director of the Basic Research Laboratory in the Department of Psychiatry at the University of Illinois. His research on the central nervous system resulted in the first major contribution to AI: a model of neurons of the brain.

McCulloch and his co-author Walter Pitts, a young mathematician, proposed a model of artificial neural networks (ANNs) in which each neuron was postulated as being in a binary state: that is, in either an *on* or *off* condition (McCulloch and Pitts, 1943). They demonstrated that their neural network model was, in fact, equivalent to the Turing machine, and proved that any computable function could be computed by some network of connected neurons. McCulloch and Pitts also showed that simple network structures could learn.

The neural network model stimulated both theoretical and experimental work to model the brain in the laboratory. However, experiments clearly demonstrated that the binary model of neurons was not correct. In fact, a neuron has highly non-linear characteristics and cannot be considered as a simple two-state device. Nonetheless, McCulloch, the second 'founding father' of AI after Alan Turing, had created the cornerstone of neural computing and ANNs. After a decline in the 1970s, the field of ANNs was revived in the late 1980s and is currently experiencing remarkable progress.

The third founder of AI was John von Neumann, the brilliant Hungarian-born mathematician. In 1930, he joined the Princeton University, lecturing in mathematical physics. He was a colleague and friend of Alan Turing. During the Second World War, von Neumann played a key role in the Manhattan Project that built the nuclear bomb. He also became an adviser for the Electronic Numerical Integrator and Calculator (ENIAC) project at the University of Pennsylvania and helped to design the **Electronic Discrete Variable Automatic Computer** (EDVAC), a *stored program* machine. He was influenced by McCulloch and Pitts's neural network model. When Marvin Minsky and Dean Edmonds, two graduate students in the Princeton mathematics department, built the first neural network computer in 1951, von Neumann encouraged and supported them.

Another of the first-generation researchers was Claude Shannon. He graduated from Massachusetts Institute of Technology (MIT) and joined Bell Telephone Laboratories in 1941. Shannon shared Alan Turing's ideas on the possibility of machine intelligence. In 1950, he published a paper on chess-playing machines, which pointed out that a typical chess game involved about 10^{120} possible moves (Shannon, 1950). Even if the new von-Neumann-type computer could examine one move per microsecond, it would take 3×10^{106} years to make its first move. Thus Shannon demonstrated the need to use heuristics in the search for the solution.

Princeton University was also home to John McCarthy, another founder of AI. He convinced Marvin Minsky and Claude Shannon to organise a summer workshop at Dartmouth College, where McCarthy worked after graduating from Princeton. In 1956, they brought together researchers interested in the study of machine intelligence, artificial neural nets and automata theory. The workshop

was sponsored by IBM. Although there were just 10 researchers, this workshop gave birth to a new science called **AI.** For the next 20 years the field of AI would be dominated by the participants of the Dartmouth workshop and their students.

1.2.2 The rise of artificial intelligence, or the era of great expectations (1956–late 1960s)

The early years of AI are characterised by tremendous enthusiasm, great ideas and very limited success. Only a few years before, computers had been introduced to perform routine mathematical calculations, but now AI researchers were demonstrating that computers could do more than that. It was an era of great expectations.

John McCarthy, one of the organisers of the Dartmouth workshop and the inventor of the term 'artificial intelligence', moved from Dartmouth to MIT. He defined the high-level language **LISP** – one of the oldest programming languages (FORTRAN is just two years older), which is still in current use. In 1958, McCarthy presented a paper, 'Programs with common sense', in which he proposed a program called the **Advice Taker** to search for solutions to general problems of the world (McCarthy, 1958). McCarthy demonstrated how his program could generate, for example, a plan to drive to the airport, based on some simple axioms. Most importantly, the program was designed to accept new axioms, or in other words new knowledge, in different areas of expertise without being reprogrammed. Thus the Advice Taker was the first complete knowledge-based system incorporating the central principles of knowledge representation and reasoning.

Another organiser of the Dartmouth workshop, Marvin Minsky, also moved to MIT. However, unlike McCarthy with his focus on formal logic, Minsky developed an anti-logical outlook on knowledge representation and reasoning. His theory of frames (Minsky, 1975) was a major contribution to knowledge engineering.

The early work on neural computing and ANNs started by McCulloch and Pitts was continued. Learning methods were improved and Frank Rosenblatt proved the **Perceptron Convergence Theorem**, demonstrating that his learning algorithm could adjust the connection strengths of a perceptron (Rosenblatt, 1962).

One of the most ambitious projects of the era of great expectations was the **General Problem Solver** (GPS) (Newell and Simon, 1961; 1972). Allen Newell and Herbert Simon from the Carnegie Mellon University developed a general-purpose program to simulate human problem-solving methods. GPS was probably the first attempt to separate the problem-solving technique from the data. It was based on the technique now referred to as **means–ends analysis.** Newell and Simon postulated that a problem to be solved could be defined in terms of *states*. The means–ends analysis was used to determine a difference between the current state and the desirable state or the *goal state* of the problem, and to choose and apply *operators* to reach the goal state. If the goal state could not be immediately reached from the current state, a new state closer to the goal would be established and the procedure repeated until the goal state was reached. The set of operators determined the solution plan.

However, GPS failed to solve complicated problems. The program was based on formal logic and therefore could generate an infinite number of possible operators, which is inherently inefficient. The amount of computer time and memory that GPS required to solve real-world problems led to the project being abandoned.

In summary, we can say that in the 1960s, AI researchers attempted to simulate the complex thinking process by inventing **general methods** for solving broad classes of problems. They used the general-purpose search mechanism to find a solution to the problem. Such approaches, now referred to as **weak methods**, applied weak information about the problem domain; this resulted in weak performance of the programs developed.

However, it was also a time when the field of AI attracted great scientists who introduced fundamental new ideas in such areas as knowledge representation, learning algorithms, neural computing and computing with words. These ideas could not be implemented then because of the limited capabilities of computers, but two decades later they have led to the development of real-life practical applications.

It is interesting to note that Lotfi Zadeh, a professor from the University of California at Berkeley, published his famous paper 'Fuzzy sets' also in the 1960s (Zadeh, 1965). This paper is now considered the foundation of fuzzy set theory. Two decades later, fuzzy researchers have built hundreds of smart machines and intelligent systems.

By 1970, the euphoria about AI was gone, and most government funding for AI projects was cancelled. AI was still a relatively new field, academic in nature, with few practical applications apart from playing games (Samuel, 1959; 1967; Greenblatt *et al.*, 1967). So, to the outsider, the achievements would be seen as toys, as no AI system at that time could manage real-world problems.

1.2.3 Unfulfilled promises, or the impact of reality (late 1960s–early 1970s)

From the mid-1950s, AI researchers were making promises to build all-purpose intelligent machines on a human-scale knowledge base by the 1980s, and to exceed human intelligence by the year 2000. By 1970, however, they realised that such claims were too optimistic. Although a few AI programs could demonstrate some level of machine intelligence in one or two toy problems, almost no AI projects could deal with a wider selection of tasks or more difficult real-world problems.

The main difficulties for AI in the late 1960s were:

- Because AI researchers were developing general methods for broad classes of problems, early programs contained little or even no knowledge about a problem domain. To solve problems, programs applied a search strategy by trying out different combinations of small steps, until the right one was found. This method worked for 'toy' problems, so it seemed reasonable that, if the programs could be 'scaled up' to solve large problems, they would finally succeed. However, this approach was wrong.

  Easy, or *tractable*, problems can be solved in polynomial time: that is, for a problem of size n, the time or number of steps needed to find the solution is

a polynomial function of *n*. On the other hand, hard or intractable problems require times that are exponential functions of the problem size. While a polynomial-time algorithm is considered to be efficient, an exponential-time algorithm is inefficient, because its execution time increases rapidly with the problem size. The theory of NP-completeness (Cook, 1971; Karp, 1972), developed in the early 1970s, showed the existence of a large class of nondeterministic polynomial problems (NP problems) that are NP-complete. A problem is called NP if its solution (if one exists) can be guessed and verified in polynomial time; non-deterministic means that no particular algorithm is followed to make the guess. The hardest problems in this class are NP-complete. Even with faster computers and larger memories, these problems are hard to solve because the algorithms for solving them have exponential time complexity. This means that as the size of the problem grows, the time taken to solve it increases exponentially.

- Many of the problems that AI attempted to solve were too broad and too difficult. A typical task for early AI was machine translation. For example, the National Research Council, USA, funded the translation of Russian scientific papers after the launch of the first artificial satellite (Sputnik) in 1957. Initially, the project team tried simply replacing Russian words with English, using an electronic dictionary. However, it was soon found that translation requires a general understanding of the subject to choose the correct words. This task was too difficult. In 1966, all translation projects funded by the US government were cancelled.
- In 1971, the British government also suspended support for AI research. Sir James Lighthill had been commissioned by the Science Research Council of Great Britain to review the current state of AI (Lighthill, 1973). He did not find any major or even significant results from AI research, and therefore saw no need to have a separate science called 'artificial intelligence'.

1.2.4 The technology of expert systems, or the key to success (early 1970s–mid-1980s)

Probably the most important development in the 1970s was the realisation that the problem domain for intelligent machines had to be sufficiently restricted. Previously, AI researchers had believed that clever search algorithms and reasoning techniques could be invented to emulate general, human-like, problem-solving methods. A general-purpose search mechanism could rely on elementary reasoning steps to find complete solutions and could use weak knowledge about the domain. However, when weak methods failed, researchers finally realised that the only way to deliver practical results was to solve typical cases in narrow areas of expertise by making large reasoning steps.

The DENDRAL program is a typical example of the emerging technology (Buchanan *et al.*, 1969). DENDRAL was developed at Stanford University to analyse chemicals. The project was required to determine the molecular structure of Martian soil based on the mass spectral data. Edward Feigenbaum (a former student of Herbert Simon), Bruce Buchanan (a computer scientist) and Joshua Lederberg (a Nobel prize winner in genetics) formed a team to solve this challenging problem.

There was no scientific algorithm for mapping the mass spectrum into its molecular structure. However, analytical chemists, such as Lederberg, could solve this problem by using their skills, experience and expertise. They could enormously reduce the number of possible structures by looking for well-known patterns of peaks in the spectrum, and thus provide just a few feasible solutions for further examination.

Feigenbaum's job was to incorporate the expertise of Lederberg into a computer program to make it perform at a human expert level. Such programs were later called **expert systems.** To understand and adopt Lederberg's knowledge and operate with his terminology, Feigenbaum had to learn basic ideas in chemistry and spectral analysis. However, it became apparent that Feigenbaum used not only rules of chemistry but also his own heuristics, or rules-of-thumb, based on his experience, and even guesswork. Soon Feigenbaum identified one of the major difficulties in the project, which he called the 'knowledge acquisition bottleneck' – how to extract knowledge from human experts to apply to computers. To articulate his knowledge, Lederberg even needed to study basics in computing. Working as a team, Feigenbaum, Buchanan and Lederberg developed DENDRAL, the first successful knowledge-based system. The key to their success was mapping all the relevant theoretical knowledge from its general form to highly specific rules ('cookbook recipes') (Feigenbaum *et al.*, 1971).

DENDRAL marked a major 'paradigm shift' in AI: a shift from general-purpose, knowledge-sparse, weak methods to domain-specific, knowledge-intensive techniques.

The next major project undertaken by Feigenbaum and others at Stanford University was in the area of medical diagnosis. The project, called MYCIN, started in 1972. It later became the PhD thesis of Edward Shortliffe (Shortliffe, 1976). MYCIN was a rule-based expert system for the diagnosis of infectious blood diseases. It also provided a doctor with therapeutic advice in a convenient, user-friendly manner.

MYCIN's knowledge consisted of about 450 IF-THEN rules and was clearly separated from the reasoning mechanism. The system developer could easily manipulate knowledge in the system by inserting or deleting some rules. For example, a domain-independent version of MYCIN called EMYCIN (Empty MYCIN) was later produced at Stanford University (van Melle, 1979; van Melle *et al.*, 1981). It had all the features of the MYCIN system except the knowledge of infectious blood diseases. EMYCIN facilitated the development of a variety of diagnostic applications. System developers just had to add new knowledge in the form of rules to obtain a new application.

MYCIN also introduced a few new features. Rules incorporated in MYCIN reflected the uncertainty associated with knowledge, in this case with medical diagnosis. It tested rule conditions (the IF part) against available data or data requested from the physician. When appropriate, MYCIN inferred the truth of a condition through a calculus of uncertainty called **certainty factors.** Reasoning in the face of uncertainty was the most important part of the system.

Another probabilistic system that generated enormous publicity was PROSPECTOR, an expert system for mineral exploration developed by the Stanford Research Institute (Duda *et al.*, 1979). The project ran from 1974 to 1983. Nine experts contributed their knowledge and expertise. To represent their knowledge, PROSPECTOR used a combined structure that incorporated rules and a semantic network. PROSPECTOR had over a thousand rules to represent extensive domain knowledge.

In exploration geology, important decisions are usually made in the face of uncertainty, with knowledge that is incomplete or fuzzy. To deal with such knowledge, PROSPECTOR incorporated Bayes' rules of evidence to propagate uncertainties through the system. PROSPECTOR performed at the level of an expert geologist and proved itself in practice. In 1980, it identified a molybdenum deposit near Mount Tolman in Washington State. Subsequent drilling by a mining company confirmed the deposit was worth over $100 million. You couldn't hope for a better justification for using expert systems.

The expert systems mentioned so far have now become classics. A growing number of successful applications of expert systems in the late 1970s showed that AI technology could move successfully from the research laboratory to the commercial environment. A 1986 survey reported a remarkable number of successful expert system applications in different areas: chemistry, electronics, engineering, geology, management, medicine, process control and military science (Waterman, 1986). Although Waterman found nearly 200 expert systems, most of the applications were in the field of medical diagnosis. Seven years later a similar survey reported over 2500 developed expert systems (Durkin, 1994). The new growing area was business and manufacturing, which accounted for about 60 per cent of the applications. Expert system technology had clearly matured.

Are expert systems really the key to success in any field? In spite of a great number of successful developments and implementations of expert systems in different areas of human knowledge, it would be a mistake to overestimate the capability of this technology. The difficulties are rather complex and lie in both technical and sociological spheres. They include the following:

- Expert systems are restricted to a very narrow domain of expertise. For example, MYCIN, which was developed for the diagnosis of infectious blood diseases, lacks any real knowledge of human physiology. If a patient has more than one disease, we cannot rely on MYCIN.
- Because of the narrow domain, expert systems are not as robust and flexible as a user might want. Furthermore, expert systems can have difficulty recognising domain boundaries. When given a task different from the typical problems, an expert system might fail in rather unpredictable ways.
- Expert systems have limited explanation capabilities. They can show the sequence of the rules they applied to reach a solution, but cannot relate accumulated, heuristic knowledge to any deeper understanding of the problem domain.
- Expert systems are also difficult to validate. No general technique is available for analysing their completeness and consistency. This makes the task of identifying incorrect, incomplete or inconsistent knowledge difficult.
- Expert systems have no ability to learn from their experience. They are built individually and cannot be developed fast. It might take from five to ten person-years to build an expert system to solve a moderately difficult problem (Waterman, 1986). The large effort would be difficult to justify if improvements to the expert system's performance depended on further attention from its developers.

Despite all these difficulties, expert systems have made the breakthrough and proved their value in a number of important applications.

1.2.5 How to make a machine learn, or the rebirth of neural networks (mid-1980s onwards)

In the mid-1980s, researchers, engineers and experts found that building an expert system required much more than just buying a reasoning system or expert system shell and putting enough rules in it. Disillusion about the applicability of expert system technology even led to people predicting an 'AI winter' with severely squeezed funding for AI projects. AI researchers decided to have a new look at neural networks.

By the late 1960s, most of the basic ideas and concepts necessary for neural computing had already been formulated (Cowan, 1990). However, only in the mid-1980s did the solution emerge. The major reason for the delay was technological: there were no PCs or powerful workstations to model and experiment with ANNs. The other reasons were psychological and financial. For example, in 1969, Minsky and Papert had mathematically demonstrated the fundamental computational limitations of one-layer perceptrons (Minsky and Papert, 1969). They also said there was no reason to expect that more complex multilayer perceptrons would represent much. This certainly would not encourage anyone to work on perceptrons and, as a result, most AI researchers deserted the field of ANNs in the 1970s.

In the 1980s, because of the need for brain-like information processing, as well as the advances in computer technology and progress in neuroscience, the field of neural networks experienced a dramatic resurgence. Major contributions to both theory and design were made on several fronts. Grossberg established a new principle of self-organisation (**adaptive resonance theory**), which provided the basis for a new class of neural networks (Grossberg, 1980). Hopfield introduced neural networks with feedback – **Hopfield networks**, which attracted much attention in the 1980s (Hopfield, 1982). Kohonen published a paper on **self-organised maps** (Kohonen, 1982). Barto, Sutton and Anderson published their work on **reinforcement learning** and its application in control (Barto *et al.*, 1983). But the real breakthrough came in 1986 when the **back-propagation learning algorithm**, first introduced by Bryson and Ho in 1969 (Bryson and Ho, 1969), was reinvented by Rumelhart and McClelland in *Parallel Distributed Processing: Explorations in the Microstructures of Cognition* (Rumelhart and McClelland, 1986). At the same time, back-propagation learning was also discovered by Parker (Parker, 1987) and LeCun (LeCun, 1988), and since then has become the most popular technique for training multilayer perceptrons. In 1988, Broomhead and Lowe found a procedure to design **layered feedforward networks** using radial basis functions, an alternative to multilayer perceptrons (Broomhead and Lowe, 1988).

ANNs have come a long way from the early models of McCulloch and Pitts to an interdisciplinary subject with roots in neuroscience, psychology, mathematics and engineering, and will continue to develop in both theory and practical applications. However, Hopfield's paper (Hopfield, 1982) and Rumelhart and McClelland's book (Rumelhart and McClelland, 1986) were the most significant and influential works responsible for the rebirth of neural networks in the 1980s.

1.2.6 Fuzzy logic, or computing with words (late 1980s onwards)

Classic expert systems are especially good for closed-system applications with precise inputs and logical outputs. They use expert knowledge in the form of rules and, if required, can interact with the user to establish a particular fact. A major drawback is that human experts cannot always express their knowledge in terms of rules or explain their line of reasoning. This can prevent the expert system from accumulating the necessary knowledge, and consequently lead to its failure. To overcome this limitation, expert systems must learn how to deal with incomplete and imprecise knowledge and data.

Most methods of handling imprecision in classic expert systems are based on the probability concept. MYCIN, for example, introduced certainty factors, while PROSPECTOR incorporated Bayes' rules to propagate uncertainties. However, experts typically do not think in terms of probability values; rather, they express their knowledge using qualifiers such as *often*, *generally*, *sometimes*, *occasionally* and *rarely*. To handle these qualitative terms effectively, another technology is required – the technology of **fuzzy logic**. Fuzzy logic is concerned with the use of fuzzy values that capture the meaning of words, human reasoning and decision making. As a method to encode and apply human knowledge in a form that accurately reflects an expert's understanding of difficult, complex problems, fuzzy logic provides the way to break through the computational bottlenecks of traditional expert systems.

At the heart of fuzzy logic lies the concept of a linguistic variable. The values of the linguistic variable are words rather than numbers. Similar to expert systems, fuzzy systems use IF-THEN rules to incorporate human knowledge, but these rules are fuzzy, such as

IF speed is high THEN stopping_distance is long

IF speed is low THEN stopping_distance is short

Fuzzy logic, or **fuzzy set theory**, was introduced by Professor Lotfi Zadeh, Berkeley's electrical engineering department chairman, in 1965 (Zadeh, 1965). It provided a means of computing with words. However, acceptance of fuzzy set theory by the technical community was slow and difficult. Part of the problem was the provocative name – 'fuzzy' – which seemed too light-hearted to be taken seriously. Eventually, fuzzy theory, ignored in the West, was taken seriously in the East – by the Japanese. It has been used successfully since 1987 in Japanese-designed dishwashers, washing machines, air conditioners, television sets, copiers and cars.

The introduction of fuzzy products gave rise to tremendous interest in this apparently 'new' technology first proposed in the mid-1960s. Hundreds of books and thousands of technical papers have been written on this topic. Some of the classics are *Fuzzy Sets, Neural Networks and Soft Computing* (Yager and Zadeh, eds, 1994); *The Fuzzy Systems Handbook* (Cox, 1999); *Fuzzy Engineering* (Kosko, 1997); *Expert Systems and Fuzzy Systems* (Negoita, 1985); and also the best-seller science book *Fuzzy Thinking* (Kosko, 1993), which popularised the field of fuzzy logic.

Most fuzzy logic applications have been in the area of control engineering. However, fuzzy control systems use only a small part of fuzzy logic's power of knowledge representation. Benefits derived from the application of fuzzy logic models in knowledge-based and decision-support systems can be summarised as follows (Cox, 1999; Pedrycz and Gomide, 2007; Turban *et al*., 2010):

- **Improved computational power.** Fuzzy rule-based systems perform faster than conventional expert systems and require fewer rules. A fuzzy expert system merges the rules, making them more powerful. Lotfi Zadeh believes that in a few years most expert systems will use fuzzy logic to solve highly non-linear and computationally difficult problems.
- **Improved cognitive modelling.** Fuzzy systems allow the encoding of knowledge in a form that reflects the way experts think about a complex problem. They usually think in such imprecise terms as *high* and *low*, *fast* and *slow*, *heavy* and *light*, and they also use such terms as *very often* and *almost never*, *usually* and *hardly ever*, *frequently* and *occasionally*. In order to build conventional rules, we need to define the crisp boundaries for these terms, thus breaking down the expertise into fragments. However, this fragmentation leads to the poor performance of conventional expert systems when they deal with highly complex problems. In contrast, fuzzy expert systems model imprecise information, capturing expertise much more closely to the way it is represented in the expert mind, and thus improve cognitive modelling of the problem.
- **The ability to represent multiple experts.** Conventional expert systems are built for a very narrow domain with clearly defined expertise. It makes the system's performance fully dependent on the right choice of experts. Although a common strategy is to find just one expert, when a more complex expert system is being built or when expertise is not well defined, *multiple* experts might be needed. Multiple experts can expand the domain, synthesise expertise and eliminate the need for a world-class expert, who is likely to be both very expensive and hard to access. However, multiple experts seldom reach close agreements; there are often differences in opinions and even conflicts. This is especially true in areas such as business and management where no simple solution exists and conflicting views should be taken into account. Fuzzy expert systems can help to represent the expertise of multiple experts when they have opposing views.

Although fuzzy systems allow expression of expert knowledge in a more natural way, they still depend on the rules extracted from the experts, and thus might be smart or dumb. Some experts can provide very clever fuzzy rules – but some just guess and may even get them wrong. Therefore, all rules must be tested and tuned, which can be a prolonged and tedious process. For example, it took Hitachi engineers several years to test and tune only 54 fuzzy rules to guide the Sendai Subway System.

Using fuzzy logic development tools, we can easily build a simple fuzzy system, but then we may spend days, weeks and even months trying out new rules and tuning our system. How do we make this process faster or, in other words, how do we generate good fuzzy rules automatically?

In recent years, several methods based on neural network technology have been used to search numerical data for fuzzy rules. Adaptive or neural fuzzy systems can find new fuzzy rules, or change and tune existing ones based on the data provided. In other words, data in – rules out, or experience in – common sense out.

1.2.7 Evolutionary computation, or learning by doing (early 1970s onwards)

Natural intelligence is a product of evolution. Therefore, by simulating biological evolution, we might expect to discover how living systems are propelled towards high-level intelligence. Nature learns by doing; biological systems are not told how to adapt to a specific environment – they simply compete for survival. The fittest species have a greater chance to reproduce, and thereby to pass their genetic material to the next generation.

The evolutionary approach to AI is based on the computational models of natural selection and genetics. Evolutionary computation works by simulating a population of individuals, evaluating their performance, generating a new population, and repeating this process a number of times.

Evolutionary computation combines three main techniques: genetic algorithms, genetic programming and evolutionary strategies.

The concept of **genetic algorithms** was introduced by John Holland in the early 1970s (Holland, 1975). He developed an algorithm for manipulating artificial 'chromosomes' (strings of binary digits), using such genetic operations as selection, crossover and mutation. Genetic algorithms are based on a solid theoretical foundation of the Schema Theorem (Holland, 1975; Goldberg, 1989).

Genetic programming represents an application of the genetic model of learning to programming. Its goal is to evolve not a coded representation of some problem but rather computer code that solves the problem. That is, genetic programming generates computer programs as the solution.

The interest in genetic programming was greatly stimulated by John Koza in the 1990s (Koza, 1992; 1994). He used genetic operations to manipulate symbolic code representing LISP programs. Genetic programming offers a solution to the main challenge of computer science – making computers solve problems without being explicitly programmed.

In the early 1960s, independently of Holland's genetic algorithms, Ingo Rechenberg and Hans-Paul Schwefel, students of the Technical University of Berlin, proposed a new optimisation method called **evolutionary strategies** (Rechenberg, 1965). Evolutionary strategies were designed specifically for solving parameter optimisation problems in engineering. Rechenberg and Schwefel suggested using random changes in the parameters, as happens in natural mutation. In fact, an evolutionary strategies approach can be considered as an alternative to the engineer's intuition. Evolutionary strategies use a numerical optimisation procedure, similar to a focused Monte Carlo search.

Both genetic algorithms and evolutionary strategies can solve a wide range of optimisation problems. They provide robust and reliable solutions for highly complex, nonlinear search and optimisation problems that previously could not be solved at all (Holland, 1995; Schwefel, 1995).

Another group of nature-inspired optimisation algorithms was introduced in the early 1990s. This group included ant colony optimisation and particle swarm optimisation. Both algorithms draw their inspiration from the collective behaviour of social organisms, and both involve multiple agents working together in exploring the solution space and iteratively improving the quality of solutions. In this sense, ant colony optimisation and particle swarm optimisation share the same fundamental properties with evolutionary computation.

The **ant colony optimisation** algorithm was originally proposed by Marco Dorigo in the early 1990s (Dorigo, 1992). He realised that ants have natural capabilities of optimising their routes and proposed the concept of artificial ants. Each artificial ant represents an agent that simulates the foraging behaviour of a real ant. Just like real ants deposit and follow pheromone trails, artificial ants deposit and update pheromone information.

Artificial ants make their decisions using pheromone information as well as problem-specific heuristics. The level of pheromone defines the probability of selecting a particular path. By moving through the problem space, artificial ants construct solutions. High-quality solutions receive higher performance levels. As a result, the algorithm tends to converge towards an optimal or near-optimal solution (Dorigo and Stützle, 2004).

Ant colony optimisation has an advantage over genetic algorithms in scenarios where the problem space can change dynamically. In such cases, the ant colony algorithm can run continuously, rapidly adapting to changes. This adaptability is particularly advantageous in network routing and transportation problems.

The **particle swarm optimisation** algorithm was proposed in the 1990s by Russell C. Eberhart and James Kennedy (Eberhart and Kennedy, 1995; Kennedy and Eberhart, 1995). They argued that sharing information among a group of individuals increases their chances for survival.

A swarm represents a group of particles or homogeneous agents such as bees, birds, ants or fish. These particles interact with the environment by moving through a multi-dimensional search space in order to find an optimal or near-optimal solution to the problem. There is no centralised control in a swarm.

The particle's position represents a candidate solution. By iteratively updating their positions, all particles collectively explore the search space and optimise the solutions.

Compared with genetic algorithms, particle swarm optimisation has fewer parameters to tune. However, in a high-dimensional search space, the particle swarm optimisation algorithm converges very slowly and often fails to discover the optimum solution.

Evolutionary computation techniques, along with ant colony optimisation and particle swarm optimisation, represent rapidly growing areas of AI and have great potential.

1.2.8 Deep neural networks, or the rise of a new era in AI (mid-2010s onwards)

ANNs with a few layers of computational neurons have been applied since the early 1990s. They serve as a universal model for representing non-linear functions and demonstrated effectiveness in solving many practical problems.

However, when confronted with more complex problems or larger data sets, traditional or *shallow* neural networks perform poorly.

Image recognition has proven to be a particularly challenging task for machine learning. To solve this task, we need to collect much larger data sets and use more powerful models. Before the 2000s, image data sets were relatively small (tens of thousands of images), and simple recognition tasks could be solved by neural networks with a small number of layers. But objects in a realistic environment vary greatly, so we need much larger data sets with millions of high-resolution images. And we need a model with a much larger learning capacity.

As a challenging problem, image recognition has attracted many AI researchers and even motivated a competition 'The ImageNet Large Scale Visual Recognition Challenge' (Russakovsky *et al.*, 2015). The ImageNet data set contains over 14 million photos categorised into more than 20,000 classes. The competition established a standardised evaluation of object recognition algorithms. The aim of the image recognition algorithm is to generate labels that identify the objects present in the images. This task is particularly difficult because many ImageNet photos contain multiple objects. For example, a dog chasing a soccer ball, or a bird's nest on a telephone pole.

In 2010, a team of researchers from the University of Toronto trained a neural network using 1.2 million training examples. The network was then tested using 150,000 images. It achieved an accuracy of 84.7 per cent, much higher than the next-best entry to the competition. How did the team from Toronto achieve such a high level of accuracy? They used a *deep* neural network with 650,000 neurons (Krizhevsky, Sutskever and Hinton, 2012).

A **deep neural network** is an ANN with multiple layers (Schmidhuber, 2015; Goodfellow *et al.*, 2016). For example, a network developed by the Microsoft team had 152 layers! Multiple layers allow deep neural networks to learn hierarchical representations of data, capturing complex patterns and features. Each layer in the network progressively extracts more abstract features, capturing complex patterns in the data.

Deep neural networks learn to perform tasks directly from images, sounds or text. While neural networks with a small number of layers perform poorly in image and speech recognition tasks, networks with multiple layers can generalise to tasks like these.

Although deep neural networks were initially introduced in the 1990s (LeCun *et al.*, 1989; LeCun *et al.*, 1998), they were regarded as an art rather than a practical AI technology. At that time deep neural networks faced several challenges that hindered their widespread adoption and acceptance as practical AI technology. The key reasons why deep neural networks were not adopted as a practical AI technology during that time include (Schmidhuber, 2015):

- **Limited computational resources.** Computers available in the 1990s were significantly less powerful compared to today's resources. Training neural networks with many layers required substantial computing power, and the limited resources available at that time made it difficult to train deep models.
- **Vanishing gradients.** Training deep neural networks involves back-propagation, where gradients are used to update the network's parameters. In deep networks, gradients could become extremely small (vanishing gradients)

during training, making it challenging to effectively update the parameters of multiple layers.

- **Lack of sufficient labelled data.** Deep neural networks, especially those used in image recognition, require large amounts of labelled data for training. In the 1990s, obtaining large labelled data sets represented a considerable challenge, limiting the ability to train deep models.
- **Difficulty in architecture design.** Designing the architecture of deep neural networks, including determining the number and structure of layers, was considered more of an art than a science. There was no guidelines for designing deep architectures.

However, over the years, advancements in hardware and the availability of large data sets have changed the situation and transformed deep neural networks into practical and powerful tools for various AI applications.

Google, for instance, decided to demonstrate the power of deep neural networks by applying them to the ancient board game Go – a game of strategy even more complex than chess. In October 2015, the program known as **AlphaGo** achieved a ground-breaking victory against Fan Hui, the European Go champion. Subsequently, in March 2016, it won in a tournament against Lee Sedol, the reigning world Go master. It is important to note that AlphaGo was not developed specifically to play Go, but rather designed as a generic program to learn how to win a game (Silver *et al.*, 2016).

Since the 2010s, many deep neural networks have been introduced and have already proved their effectiveness. The most popular architectures include **AlexNet** (Krizhevsky, Sutskever and Hinton, 2012), **GoogLeNet** (Szegedy *et al.*, 2015; Szegedy *et al.*, 2016), **ResNet** (He *et al.*, 2016a; He *et al.*, 2016b) and **DenseNet** (Huang, 2017).

Deep neural networks are used in driverless cars to enable safe driving, detect traffic lights and recognise road signs. They are also applied in voice-controlled consumer devices such as phones, tablets, TVs and hands-free speakers. Deep neural networks can recognise images and identify people and objects. They can automatically translate text from one language to another, learn to predict earthquakes and forecast energy market prices.

It is also important to mention here the concept of **transfer learning**, where pre-built and pre-trained deep neural networks are adapted to new tasks. The main idea behind transfer learning is to leverage knowledge captured in the pre-trained neural network, thereby significantly reducing the size of the training data set required to achieve satisfactory performance on a new but related task. This approach is particularly useful in image recognition when the number of available images could be limited.

Deep networks have demonstrated their astonishing abilities in both business and research. But, are they immune to problems? In reality, as the depth of the network increases, training becomes more challenging and demands substantial computational resources. Deep learning models require large amounts of labelled data for effective training, but obtaining and preparing such data sets is often time-consuming. On the other hand, when dealing with limited data, deep networks are prone to overfitting with a model performing well on the training data but struggling to generalise to new, unseen data.

Deep learning models can also amplify biases present in the training data. This can lead to biased predictions, especially in applications like image recognition and natural language processing (Zhao *et al.*, 2017; Foulds *et al.*, 2021; Zhao *et al.*, 2021). For instance, research studies have shown that facial recognition systems, when trained on data sets dominated by images of white males, tend to perform less accurately on darker-skinned individuals, especially females.

While the problem of biased predictions can be mitigated by training networks with diverse data sets, an even more serious issue arises. Deep neural networks are vulnerable to adversarial attacks, where small, carefully crafted perturbations to the input can lead to misclassification. Professor Geoffrey Hinton, regarded as the 'godfather' of deep learning, mentioned in his speech at the AAAI Conference in 2020 that by adding a tiny bit of noise to an image, a neural network could recognise it as something entirely different. In the field of AI, these manipulated images are known as adversarial examples (Szegedy *et al.*, 2014). Adversarial examples often cause deep neural networks to misclassify objects, even though to the human eye, they appear identical to the original images. Therefore, addressing the threat posed by adversarial examples is critical for the security of machine learning applications.

Despite these challenges, ongoing research and developments in the field make deep neural networks an increasingly powerful tool applicable to a wide range of problems. Deep neural networks have also paved the way to generative AI.

1.3 Generative AI

Generative AI can be defined as a type of AI that can produce various types of new content including text, images, videos and audio. Although generative AI has recently attracted a lot of attention as a new revolution in AI, it is *not* a new concept. In fact, generative AI was introduced in the 1960s in chatbots.

The first chatbot, called **ELIZA**, was developed at MIT by Professor Joseph Weizenbaum in the mid-1960s (Weizenbaum, 1976). The program used pattern matching and substitution techniques to mimic human conversations. The ELIZA chatbot worked by pairing words entered by the user to a list of possible scripted responses.

In the decades that followed, chatbots strived for more human-like interactions with the goal of passing the Turing test. However, none of them succeeded because the hardest part of the Turing test is that there is no limit on the topics of human conversations.

Of course, the development of chatbots continued. In 2010, Apple introduced **Siri**, an intelligent personal assistant that used a natural language interface. Siri paved the way for a new generation of AI bots.

In November 2014, Amazon presented **Alexa**, a smart speaker equipped with voice recognition capabilities. Since then, Alexa has been integrated in various Amazon devices, and has become a widely used voice-activated assistant in homes and businesses around the world.

In May 2016, Google launched **Google Assistant**, a virtual assistant designed to provide users with a natural way to interact with their devices. Google Assistant can perform a wide range of tasks including answering general knowledge

questions, setting reminders and alarms, managing calendar events, sending messages, making calls, checking the weather, providing directions, playing music and media, controlling smart home devices through voice commands, translating phrases and sentences into different languages, and more.

However, it wasn't until the introduction of **ChatGPT** in January 2022 that generative AI could truly capture the public's imagination. It took just two months for ChatGPT to reach 100 million users! For comparison, Google Translate took 78 months to reach 100 million users. According to OpenAI data, in December 2023, ChatGPT had more that 180 million users.

So, what makes ChatGPT so popular?

We will let ChatGPT answer this question.

"*ChatGPT has gained popularity for several reasons:*

1. ***Natural Language Understanding:*** *ChatGPT demonstrates a strong ability to understand and generate human-like text. It can comprehend context, engage in coherent conversations, and provide relevant responses, making interactions feel more natural.*
2. ***Large-Scale Training:*** *ChatGPT is built on the GPT (Generative Pre-trained Transformer) architecture, specifically GPT-3.5. It has been trained on an extensive and diverse dataset, enabling it to capture a broad range of language patterns and information.*
3. ***Versatility:*** *ChatGPT is versatile and can be used for a wide range of tasks. Users can leverage it for generating creative writing, answering questions, brainstorming ideas, learning new topics, and more. Its flexibility contributes to its widespread adoption.*
4. ***Accessibility:*** *OpenAI has made ChatGPT accessible to the public through various platforms, allowing users to interact with the model via API or through web-based interfaces. This accessibility has contributed to its popularity and widespread use.*
5. ***Innovative Use Cases:*** *Users have found innovative and creative ways to apply ChatGPT in different domains, from content creation to programming assistance. The model's general-purpose nature encourages experimentation and exploration of various applications.*
6. ***Continual Improvement:*** *OpenAI actively seeks user feedback and has a history of refining and improving its models based on community input. This iterative process helps enhance the capabilities of ChatGPT over time.*
7. ***Research and Development:*** *OpenAI's commitment to advancing the field of artificial intelligence and language models has contributed to the popularity of ChatGPT. The model is a result of ongoing research and development efforts.*
8. ***Public Interest and Hype:*** *The release of each new version of the GPT series, including ChatGPT, tends to generate public interest and excitement. This, along with media coverage, contributes to the model's popularity.*

It's important to note that while ChatGPT has garnered attention and popularity, it also has limitations, such as occasional generation of incorrect or nonsensical answers and sensitivity to input phrasing. Users are encouraged to use the model responsibly and be aware of its strengths and weaknesses."

What is the main technology behind ChatGPT?

One particular technology that played a critical role in the success of ChatGPT and similar AI technologies is called transformers.

In 2017, Google researchers published the seminal paper 'Attention is all you need' (Vaswani *et al.*, 2017), in which they introduced the concept of transformers. **Transformers** are a type of deep neural network architecture that consists of multiple layers or 'transformer blocks' (or just transformers). Each transformer contains sub-layers, including self-attention mechanisms and feedforward neural networks. The depth of the transformer architecture allows it to capture complex hierarchical relationships and dependencies in sequential data, making it particularly suitable for natural language processing tasks.

How deep is the transformer architecture?

The original transformer model, introduced in the paper 'Attention is all you need', had approximately 65 million trainable parameters for the base model and up to 215 million parameters for larger variants. Google researchers trained transformers to perform English-to-German and English-to-French translation tasks and demonstrated that transformers could outperform any other type of neural networks in both accuracy and the time required for training. More recent transformer models, such as OpenAI's GPT (Generative Pre-trained Transformer), are significantly larger. GPT-3 has 175 billion parameters in its full model. This is 10 million times larger than the number of trainable parameters in the neural networks of the 1980s. But transformers continue to grow. GPT-4 has about 1.8 trillion parameters, over 10 times larger than GPT-3!

However, there is no point in having such large neural networks unless we can train it with enough data. But, where can we get that training data from? Of course, from the World Wide Web. ChatGPT has been trained on around 500 billion words including both text-based and code-based data. Some speculations suggest that additional sources like Twitter, Reddit, YouTube and a large collection of textbooks have also been used.

Transformers have become a cornerstone in the development of **large language models** (LLMs), providing a foundation for understanding language, context and semantics.

What is a large language model? How does it work?

A large language model can be defined as a type of AI algorithm that uses deep learning and very large data sets to understand, summarise, generate and predict new content. The training of a LLM usually begins with unsupervised learning when transformers are exposed to unlabelled text from a wide range of sources including websites, books, articles and more. The model learns to predict the next word in a sequence. The self-attention mechanism in transformers enables the model to capture important dependencies and relationships between words.

Once the transformers have been pre-trained, the LLM is fine-tuned for specific tasks using labelled data sets. The fine-tuning approach is particularly powerful because the pre-trained model has already acquired a general understanding of language.

Large-scale pre-training followed by fine-tuning for specific applications has become a standard practice in the development of LLMs. Pre-trained transformers

has proven effective in text classification, sentiment analysis, machine translation, text summarisation and many more. They have excelled in question-answering tasks, where a chatbot must understand and respond to a user's question by extracting relevant information from a given context. Transformers have also been highly successful in text generation tasks, such as content creation and creative writing. Furthermore, transformers can represent relationships between different modes of data. For example, they are used to align images and speech with text, as well as to translate instructions given in natural language into images.

While transformers have excelled in natural language processing tasks by capturing complex relationships and patterns in sequential data, **generative adversarial networks** have demonstrated their abilities in generating new data, including images, videos and audio.

What are generative adversarial networks?

Generative adversarial networks, or GANs for short, represent a machine learning model where two deep neural networks compete with each other to make more accurate predictions. The concept of GANs was developed by Ian Goodfellow and his colleagues at the Université de Montréal in the mid-2010s (Goodfellow *et al.*, 2014). The two networks, called the generator and the discriminator, are designed to generate new data samples that closely resemble given data samples.

The generator produces new data samples by taking random noise as input. Its goal is to create artificial data samples, such as images, that are indistinguishable from real images.

The discriminator evaluates the input data samples, attempting to discriminate between real images in the given data set and artificial, or fake, images generated by the generator. The goal of the discriminator is to classify the presented images correctly.

Both the generator and the discriminator are trained simultaneously to enable competition. The discriminator provides feedback to the generator, and as a result of this feedback, the generator adjusts its parameters to improve the quality of the fake images. The discriminator also updates its parameters to enhance its own performance in discriminating between real and artificial images. The training process continues until the generator produces images that are indistinguishable from real images, and the discriminator cannot reliably discriminate between real and generated images.

GANs have demonstrated remarkable abilities in generating realistic data and have been successfully applied in various domains, including image generation, computer vision, medical imaging, text-to-image synthesis, drug discovery, anomaly detection and more.

Unfortunately, as is often the case with new AI tools, concerns are already arising, particularly regarding deepfakes – digitally forged images and videos – as well as the potential for malicious cyberattacks that realistically mimic real people.

Let us mention just a few notable deepfake examples that have attracted public attention due to their realism.

In 2018, comedian and filmmaker Jordan Peele, in collaboration with BuzzFeed, created a deepfake video featuring former President Barack Obama delivering a fabricated speech. The video aimed to raise awareness about potential dangers of deepfake technology.

In June 2019, a deepfake featuring Mark Zuckerberg giving a sinister speech about the power of Facebook was posted on Instagram.

During the 2020 presidential campaign in the USA, many deepfake videos surfaced showing Joe Biden falling asleep during an interview, getting lost and misspeaking – all bolstering rumours about his cognitive decline.

In 2021, Belgian visual effects artist Chris Umé created a series of realistic deepfake videos featuring actor Tom Cruise. The videos were released on TikTok and attracted tens of millions of views. They were so convincing that security experts expressed their concerns.

In May 2023, a deepfake video of Vice President Kamala Harris speaking gibberish went viral on social media.

It is important to notice that while some deepfake content is created for harmless entertainment purposes, there are real concerns about the potential misuse of this technology for malicious activities, including spreading misinformation, identity theft and creating convincing hoaxes. As technology advances, vigilance and awareness about the existence and implications of deepfakes are becoming increasingly important.

1.4 Summary

We live in the era of the knowledge revolution, when the power of a nation is determined not by the number of soldiers in its army but the knowledge it possesses. Science, medicine, engineering and business propel nations towards a higher quality of life, but they also require highly qualified and skilful people. We are now adopting intelligent machines that can capture the expertise of such knowledgeable people and reason in a manner similar to humans.

The desire for intelligent machines was just an elusive dream until the first computer was developed. The early computers could manipulate large databases effectively by following prescribed algorithms, but could not reason about the information provided. This gave rise to the question of whether computers could ever think. Alan Turing defined the intelligent behaviour of a computer as the ability to achieve human-level performance in a cognitive task. The Turing test provided a basis for the verification and validation of knowledge-based systems.

In 1956, a summer workshop at Dartmouth College brought together 10 researchers interested in the study of machine intelligence, and a new science – artificial intelligence – was born.

Since the early 1950s, AI technology has developed from the curiosity of a few researchers to a valuable tool to support humans making decisions. We have seen historical cycles of AI from the era of great ideas and great expectations in the 1960s to the disillusionment and funding cutbacks in the early 1970s; from the development of the first expert systems, such as DENDRAL, MYCIN and PROSPECTOR, in the 1970s to the maturity of expert system technology and its massive applications in different areas in the 1980s and 1990s; from a simple binary model of neurons proposed in the 1940s to a dramatic resurgence of the field of ANNs in the 1980s; from the introduction of fuzzy set theory and its being ignored by the West in the 1960s to numerous 'fuzzy' consumer products offered by the Japanese in the 1980s and world-wide acceptance of 'soft' computing and

computing with words in the 1990s. In the early 2010s, we witnessed the rise of deep neural networks, an increasingly powerful tool applicable to a wide range of problems. These networks paved the way for generative AI, truly capturing the public imagination.

Table 1.1 summarises the key events in the history of AI and knowledge engineering from the first work on AI by McCulloch and Pitts in 1943 to the recent trends in AI.

Table 1.1 A summary of the main events in the history of AI

Period	Key events
The birth of artificial intelligence (1943–1956)	McCulloch and Pitts, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, 1943
	Turing, *Computing Machinery and Intelligence*, 1950
	The Electronic Numerical Integrator and Calculator project (von Neumann)
	Shannon, *Programming a Computer for Playing Chess*, 1950
	The Dartmouth College summer workshop on machine intelligence, artificial neural nets and automata theory, 1956
The rise of artificial intelligence (1956–late 1960s)	*LISP* (McCarthy)
	The General Problem Solver (GPR) project (Newell and Simon)
	Newell and Simon, *Human Problem Solving*, 1972
	Minsky, *A Framework for Representing Knowledge*, 1975
The disillusionment in artificial intelligence (late 1960s–early 1970s)	Cook, *The Complexity of Theorem Proving Procedures*, 1971
	Karp, *Reducibility Among Combinatorial Problems*, 1972
	The Lighthill Report, 1971
The discovery of expert systems (early 1970s–mid-1980s)	DENDRAL (Feigenbaum, Buchanan and Lederberg, Stanford University)
	MYCIN (Feigenbaum and Shortliffe, Stanford University)
	PROSPECTOR (Stanford Research Institute)
	PROLOG – a Logic Programming Language (Colmerauer, Roussel and Kowalski, France)
	EMYCIN (Stanford University)
	Waterman, *A Guide to Expert Systems*, 1986
The rebirth of artificial neural networks (mid-1980s onwards)	Hopfield, *Neural Networks and Physical Systems with Emergent Collective Computational Abilities*, 1982
	Kohonen, *Self-Organized Formation of Topologically Correct Feature Maps*, 1982
	Rumelhart and McClelland, *Parallel Distributed Processing*, 1986
	The First IEEE International Conference on Neural Networks, 1987
	Haykin, *Neural Networks*, 1994
	Neural Network, MATLAB Application Toolbox (The MathWorks, Inc.)

Table 1.1 A summary of the main events in the history of AI (continued)

Period	Key events
Computing with words (late 1980s onwards)	Zadeh, *Fuzzy Sets*, 1965
	Zadeh, *Fuzzy Algorithms*, 1969
	Mamdani, *Application of Fuzzy Logic to Approximate Reasoning Using Linguistic Synthesis*, 1977
	Sugeno, *Fuzzy Theory*, 1983
	Japanese 'fuzzy' consumer products (dishwashers, washing machines, air conditioners, television sets)
	Sendai Subway System (Hitachi, Japan), 1986
	The First IEEE International Conference on Fuzzy Systems, 1992
	Kosko, *Fuzzy Thinking*, 1993
	Yager and Zadeh, *Fuzzy Sets, Neural Networks and Soft Computing*, 1994
	Cox, *The Fuzzy Systems Handbook*, 1994
	Zadeh, *Computing with Words – A Paradigm Shift*, 1996
	Fuzzy Logic, MATLAB Toolbox (The MathWorks, Inc.)
Evolutionary computation (early 1970s onwards)	Holland, *Adaptation in Natural and Artificial Systems*, 1975
	Koza, *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*, 1992
	Dorigo, *Optimization, Learning and Natural Algorithms*, 1992
	Kennedy and Eberhart, *Particle Swarm Optimization*, 1995
	Schwefel, *Evolution and Optimum Seeking*, 1995
	Fogel, *Evolutionary Computation – Towards a New Philosophy of Machine Intelligence*, 1995
Deep neural networks and generative AI (mid-2010s onwards)	The ImageNet Large Scale Visual Recognition Challenge, 2010
	Krizhevsky, Sutskever and Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012
	AlexNet, 2012
	Goodfellow *et al., Generative Adversarial Networks*, 2014
	GoogLeNet, 2015
	ResNet, 2016
	DenseNet, 2017
	Vaswani *et al*., Attention is all you need, 2017
	Deep Neural Networks, MATLAB Toolbox (The MathWorks, Inc.)
	ChatGPT, 2022
	GPT-4, 2023

The most important lessons learned in this chapter are as follows:

- Intelligence is the ability to learn and understand, to solve problems and to make decisions. AI is a science that has defined its goal as making machines do things that would require intelligence if done by humans.

- A machine is thought intelligent if it can achieve human-level performance in some cognitive task. To build an intelligent machine, we have to capture, organise and use human expert knowledge in some problem area.
- The realisation that the problem domain for intelligent machines had to be sufficiently restricted marked a major 'paradigm shift' in AI from general-purpose, knowledge-sparse, weak methods to domain-specific, knowledge-intensive methods. This led to the development of expert systems – computer programs capable of performing at a human-expert level in a narrow problem area. Expert systems use human knowledge and expertise in the form of specific rules, and are distinguished by the clean separation of the knowledge and the reasoning mechanism. They can also explain their reasoning procedures.
- One of the main difficulties in building intelligent machines, or in other words in knowledge engineering, is the 'knowledge acquisition bottleneck' – extracting knowledge from human experts.
- Fuzzy logic, or fuzzy set theory, provides a means to compute imprecise linguistic terms. It concentrates on the use of fuzzy values that capture the meaning of words, human reasoning and decision making, and provides a way of breaking through the computational burden of traditional expert systems.
- Expert systems can neither learn nor improve themselves through experience. They are individually created and demand large efforts for their development. It can take from five to ten person-years to build even a moderate expert system. Machine learning can accelerate this process significantly and enhance the quality of knowledge by adding new rules or changing incorrect ones.
- ANNs, inspired by biological neural networks, learn from historical cases and make it possible to model complex relationships within data. They can identify patterns and predict trends. These capabilities enables ANNs to excel in such tasks as pattern recognition, classification, time series analysis, forecasting and many more.
- The evolutionary approach to AI is based on the computational models of natural selection and genetics. Evolutionary computation works by simulating a population of individuals, evaluating their performance, generating a new population and repeating this process a number of times.
- Another group of nature-inspired optimisation algorithms includes ant colony optimisation and particle swarm optimisation. Both algorithms draw their inspiration from the collective behaviour of social organisms, and both involve multiple agents working together in exploring the solution space and iteratively improving the quality of solutions.
- ANNs, even with just a few layers of computational neurons, have been successfully employed since the early 1990s. However, as the complexity of the problem or the size of the data set increases, neural networks with only a few layers tend to perform poorly. Deeper networks become essential. Multiple layers enable deep neural networks to learn hierarchical representations of data, capturing complex patterns and features.

- Deep neural networks learn to perform tasks directly from images, sounds or text. While neural networks with a small number of layers may struggle with image and speech recognition tasks, networks with multiple layers can generalise to such tasks.
- Deep neural networks are now used in driverless cars and voice-controlled consumer devices. They can recognise images and identify people and objects. Deep neural networks automatically translate text from one language to another, and learn to predict earthquakes and forecast energy market prices.
- Deep neural networks have paved the way to generative AI, a type of AI that can produce various types of new content including text, images, videos and audio. However, it wasn't until the introduction of ChatGPT in January 2022 that generative AI could capture the public's imagination. It took just two months for ChatGPT to reach 100 million users.
- The technology that played a critical role in the success of ChatGPT and similar AI technologies is called transformers. Transformers are a type of deep neural network architecture that consists of multiple layers or 'transformer blocks'. The depth of the transformer architecture allows it to capture complex hierarchical relationships and dependencies in sequential data, making it particularly suitable for natural language processing tasks.
- While transformers have excelled in natural language processing tasks by capturing complex relationships and patterns in sequential data, generative adversarial networks have demonstrated their abilities in generating new data, including images, videos and audio.
- Unfortunately, as is often the case with new AI tools, concerns are already arising, particularly regarding the increasing risks of the dissemination of fake news, inaccurate and misleading information, digitally forged images and videos, as well as the growing threat of malicious cyberattacks. With advancing technology, vigilance and awareness about the existence and implications of deepfakes are becoming increasingly important.

Questions for Review

1 Define intelligence. What is the intelligent behaviour of a machine?

2 Describe the Turing test for AI and justify its validity from a modern standpoint.

3 Define AI as a science. When was AI born?

4 What are weak methods? Identify the main difficulties that led to the disillusion with AI in the early 1970s.

5 Define expert systems. List the common characteristics of early expert systems such as DENDRAL, MYCIN and PROSPECTOR. What are the limitations of expert systems?

6 What are the differences between expert systems and ANNs? Why was the field of ANNs reborn in the 1980s?

7 What are the premises on which fuzzy logic is based? When was fuzzy set theory introduced? What are the main advantages of applying fuzzy logic in knowledge-based systems?

8 What is evolutionary computation? What are the main tools of evolutionary computation?

9 In what scenarios does ant colony optimisation have an advantage over genetic algorithms? What is the primary advantage of particle swarm optimisation in terms of parameter tuning when compared to genetic algorithms?

10 What is a deep neural network? What are the main differences between deep neural networks and shallow neural networks?

11 What is generative AI? What makes ChatGPT so popular? What are the main technologies behind generative AI? What risks are associated with generative AI?

12 What are transformers? What are generative adversarial networks? Which tasks are more suitable for transformers, and which tasks are more suitable for generative adversarial networks?

References

Barto, A.G., Sutton, R.S. and Anderson C.W. (1983). Neurolike adaptive elements that can solve difficult learning control problems, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13, 834–846.

Boden, M.A. (1977). *Artificial Intelligence and Natural Man*. Basic Books, New York.

Broomhead, D.S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks, *Complex Systems*, 2, 321–355.

Bryson, A.E. and Ho, Y.-C. (1969). *Applied Optimal Control*. Blaisdell, New York.

Buchanan, B.G., Sutherland, G.L. and Feigenbaum, E.A. (1969). Heuristic DENDRAL: a program for generating explanatory hypotheses in organic chemistry, *Machine Intelligence 4*, B. Meltzer, D. Michie and M. Swann, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 209–254.

Cook, S.A. (1971). The complexity of theorem proving procedures, *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, New York, pp. 151–158.

Cowan, J.D. (1990). Neural networks: the early days, *Advances in Neural Information Processing Systems 2*, D.S. Tourefzky, ed., Morgan Kaufman, San Mateo, CA, pp. 828–842.

Cox, E. (1999). *The Fuzzy Systems Handbook: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems*, 2nd edn. Academic Press, San Diego, CA.

Dorigo, M. (1992). Optimization, Learning and Natural Algorithms (in Italian), PhD dissertation, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

Dorigo M. and Stützle, T. (2004). Ant Colony Optimization, MIT Press, Cambridge, MA.

Duda, R., Gaschnig, J. and Hart, P. (1979). Model design in the PROSPECTOR consultant system for mineral exploration, *Expert Systems in the Microelectronic Age*, D. Michie, ed., Edinburgh University Press, Edinburgh, Scotland, pp. 153–167.

Durkin, J. (1994). *Expert Systems: Design and Development*. Prentice Hall, Englewood Cliffs, NJ.

Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory, *Proceedings of the 6th International Symposium on Micro Machine and Human Science* (MHS), Nagoya, Japan, 4–6 October 1995, pp. 39–43.

Feigenbaum, E.A., Buchanan, B.G. and Lederberg, J. (1971). On generality and problem solving: a case study using the DENDRAL program, *Machine Intelligence 6*, B. Meltzer and D. Michie, eds, Edinburgh University Press, Edinburgh, Scotland, pp. 165–190.

Fogel, D.B. (1995). *Evolutionary Computation – Towards a New Philosophy of Machine Intelligence*. IEEE Press, Piscataway, NJ.

Foulds, J., Islam, R., Keya, K.N. and Pan, S. (2020). An intersectional definition of fairness, *Proceedings of the 36th International Conference on Data Engineering* (ICDE), Dallas, Texas, 20–22 April 2020, pp. 1918–1921.

Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning*. Addison-Wesley, Reading, MA.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial networks. *Proceedings of the 27th International Conference on Neural Information Processing Systems* (NIPS'14), 8–13 December 2014, Montréal Canada, pp. 2672–2680.

Goodfellow, I., Yoshua Bengio, Y. and Courville, A. (2016). *Deep Learning*, The MIT Press, Cambridge, MA.

Greenblatt, R.D., Eastlake, D.E. and Crocker, S.D. (1967). The Greenblatt Chess Program, *Proceedings of the Fall Joint Computer Conference*, pp. 801–810.

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review*, 87, 1–51.

He, K., Zhang, X., Ren, S. and Sun, J. (2016a). Deep residual learning for image recognition, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27–30 June 2016, pp. 770–778.

He, K., Zhang, X., Ren, S. and Sun, J. (2016b). Identity mappings in deep residual networks, *Proceedings of the European Conference on Computer Vision*, Springer: Cham, Switzerland, pp. 630–645.

Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

Holland, J.H. (1995). *Hidden Order: How Adaptation Builds Complexity*. Perseus Books, New York.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, *Proceedings of the National Academy of Sciences of the USA*, 79, 2554–2558.

Huang, G., Liu, Z., van der Maaten, L. and Weinberger, K.Q. (2017). Densely connected convolutional networks, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 21–26 July 2017, pp. 4700–4708.

Karp, R.M. (1972). Reducibility among combinatorial problems, *Complexity of Computer Computations*, R.E. Miller and J.W. Thatcher, eds, Plenum, New York, pp. 85–103.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization, *Proceedings of the International Conference on Neural Networks* (ICNN), Perth, Australia, 27 November–1 December 1995; Vol. 4, pp. 1942–1948.

Kohonen, T. (1982). Self-organized formation of topologically correct feature maps, *Biological Cybernetics*, 43, 59–69.

Kosko, B. (1993). *Fuzzy Thinking: The New Science of Fuzzy Logic*. Hyperion, New York.

Kosko, B. (1997). *Fuzzy Engineering*. Prentice Hall, Upper Saddle River, NJ.

Koza, J.R. (1992). *Genetic Programming: On the Programming of the Computers by Means of Natural Selection*. MIT Press, Cambridge, MA.

Koza, J.R. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge, MA.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet classification with deep convolutional neural networks, Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012, pp. 1106–1114.

LeCun, Y. (1988). A theoretical framework for back-propagation, *Proceedings of the 1988 Connectionist Models Summer School*, D. Touretzky, G. Hilton and T. Sejnowski, eds, Morgan Kaufmann, San Mateo, CA, pp. 21–28.

LeCun, Y., Jackel, L. D., Boser, B., Denker, J. S., Graf, H. P., Guyon, I., Henderson, D., Howard R. E. and Hubbard W. (1989). Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning, *IEEE Communication*, pp. 41–46.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998). Gradient-based learning applied to document recognition, *Proceedings IEEE*, 86, pp. 2278–2324.

Lighthill, J. (1973). Artificial intelligence: a general survey, *Artificial Intelligence: A Paper Symposium*. J. Lighthill, N.S. Sutherland, R.M. Needham, H.C. Longuest-Higgins and D. Michie, eds, Science Research Council of Great Britain, London.

McCarthy, J. (1958). Programs with common sense, *Proceedings of the Symposium on Mechanisation of Thought Processes*, vol. 1, London, pp. 77–84.

McCulloch, W.S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, 5, 115–137.

Minsky, M.L. (1975). A framework for representing knowledge, *The Psychology of Computer Vision*, P. Winston, ed., McGraw-Hill, New York, pp. 211–277.

Minsky, M.L. and Papert, S.A. (1969). *Perceptrons*. MIT Press, Cambridge, MA.

Negoita, C.V. (1985). *Expert Systems and Fuzzy Systems*. Benjamin/Cummings, Menlo Park, CA.

Newell, A. and Simon, H.A. (1961). GPS, a program that simulates human thought, *Lernende Automatten*, H. Billing, ed., R. Oldenbourg, Munich, pp. 109–124.

Newell, A. and Simon, H.A. (1972). *Human Problem Solving*. Prentice Hall, Englewood Cliffs, NJ.

Parker, D.B. (1987). Optimal algorithms for adaptive networks: second order back propagation, second order direct propagation, and second order Hebbian learning, *Proceedings of the IEEE 1st International Conference on Neural Networks*, San Diego, CA, vol. 2, pp. 593–600.

Pedrycz, W. and Gomide, F. (2007). *Fuzzy Systems Engineering: Toward Human-Centric Computing*. John Wiley, Hoboken, NJ.

Rechenberg, I. (1965). *Cybernetic Solution Path of an Experimental Problem*. Ministry of Aviation, Royal Aircraft Establishment, Library Translation No. 1122, August.

Rechenberg, I. (1973). *Evolutionsstrategien – Optimierung Technischer Systeme Nach Prinzipien der Biologischen Information*. Friedrich Frommann Verlag (Günther Holzboog K.G.), Stuttgart – Bad Cannstatt.

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan, Chicago.

Rumelhart, D.E. and McClelland, J.L., eds (1986). *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, 2 vols. MIT Press, Cambridge, MA.

Samuel, A.L. (1959). Some studies in machine learning using the game of checkers, *IBM Journal of Research and Development*, 3(3), 210–229.

Samuel, A.L. (1967). Some studies in machine learning using the game of checkers II – recent progress, *IBM Journal of Research and Development*, 11(6), 601–617.

Schmidhuber, J. (2015), Deep Learning in Neural Networks: An Overview, Neural Networks, 6, pp. 85–117.

Schwefel, H.P. (1995). *Evolution and Optimum Seeking*. John Wiley, New York.

Shannon, C.E. (1950). Programming a computer for playing chess, *Philosophical Magazine*, 41(4), 256–275.

Shortliffe, E.H. (1976). *MYCIN: Computer-Based Medical Consultations*. Elsevier Press, New York.

Silver, D. et al. (2016). Mastering the Game of Go with deep neural networks and tree search, *Nature*, 529, pp. 484–489.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. (2014). Intriguing properties of neural networks, *Proceedings of the 2nd*

International Conference on Learning Representations, ICLR 2014, Banff, Canada, April 14-16, 2014, pp. 1–10.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A. (2015). Going deeper with convolutions, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, 7–12 June 2015, pp. 1–9.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J. and Wojna, Z. (2016). Rethinking the inception architecture for computer vision, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 27–30 June 2016, pp. 2818–2826.

Turban, E., Sharda, R. and Delen, D. (2010). *Decision Support and Business Intelligent Systems*, 9th edn. Prentice Hall, Englewood Cliffs, NJ.

Turing, A.M. (1950). Computing machinery and intelligence, *Mind*, 59, 433–460.

van Melle, W. (1979). A domain independent production-rule system for consultation programs, *Proceedings of the IJCAI 6*, pp. 923–925.

van Melle, W., Shortliffe, E.H. and Buchanan B.G. (1981). EMYCIN: a domain-independent system that aids in constructing knowledge-based consultation programs, *Machine Intelligence, Infotech State of the Art Report 9*, no. 3.

Vaswani, A. et al. (2017). Attention is all you need, *Proceedings of the 31st Conference on Neural Information Processing Systems* (NIPS 2017), Long Beach, CA, USA, pp. 1–11.

Waterman, D.A. (1986). *A Guide to Expert Systems*. Addison-Wesley, Reading, MA.

Weizenbaum, J. (1976). Computer Power and Human Reason: From Judgment to Calculation. New York: W. H. Freeman and Company.

Yager, R.R. and Zadeh, L.A., eds (1994). *Fuzzy Sets, Neural Networks and Soft Computing*. Van Nostrand Reinhold, New York.

Zadeh, L. (1965). Fuzzy sets, *Information and Control*, 8(3), 338–353.

Zhao, D., Wang, A. and Russakovsky, O. (2021). Understanding and evaluating racial biases in image captioning, *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops* (ICCVW), 11–17 October 2021, Montreal, BC, Canada, pp. 14830–14840.

Zhao, J., Wang, T., Yatskar, M., Ordonez, V. and Chang, K.-W. (2017). Men also like shopping: reducing gender bias amplification using corpus-level constraints, *Proceedings of the Conference on Empirical Methods in Natural Language Processing* (EMNLP), 9–11 September 2017, Copenhagen, Denmark, pp. 2979–2989.