Understanding Artificial Intelligence

From the creation of the first Artificial Intelligence, to creating your own

Finn de Horde, Luke de Jeu, Tony Munzer and Milan La Rivière
Supervisor: R. de Vries
31-1-2020

Table of contents

Introduction

In recent years, the subject of Artificial Intelligence (AI) has received a lot of attention. This relatively new field of research pushes both scientists and ordinary people, to view the world from a different perspective. AI-research teaches us that machine intelligence is both possible and plausible.

With the ultimate goal of AI-research being the creation of AI equivalent to, or even surpassing human intelligence, implications of AI-related research stretch far beyond computer science. The field of Artificial Intelligence overlaps with various research fields such as philosophy, ethics, biology, economics and mathematics, each regarding AI from a different perspective. Furthermore, Artificial Intelligence has applications in countless other fields of research, making almost all AI-related work interdisciplinary in nature.

The goal of this paper is to provide readers with a rudimentary understanding of the concept and workings of Artificial Intelligence. This paper will prove useful to those wishing to include Artificial Intelligence in their work, but are lacking a foundational understanding of the various divisions within the field of AI. The paper also guides readers through the process of creating one’s first AI, while at the same time explaining the basic elements as well as the turbulent history of Artificial Intelligence. So as not to limit the usability of the paper, the scope of the research has been kept as broad as possible. To maximise the project feasibility, the practical component of this paper has been limited to the programming language Python and the libraries Tensorflow and Keras. The information presented in this paper has been compiled from literary research. Because there was no predefined scope to the paper, no literature selection protocol was followed.

In the first chapter, the history of AI is explained from birth up to the present. Reading chapter two provides the reader with a basic understanding of AI in general. Chapter three explains how AI and other research fields are interconnected. Finally, in chapter four, the reader is guided through the process of building an AI, using the writers’ own project as an example.

History of AI

The Golden Age of Science Fiction (1938-1946)

Before the birth of Artificial Intelligence, science fiction was the driving factor in inspiring, but also warning people about machines that can think like humans. Many consider 1938-1946 to be the Golden Age of Science Fiction (Nicholls, 1981)[1]. In this era, science fiction gained widespread public attention and many classic science fiction stories were published. One of the most influential authors from this period was Isaac Asimov. Isaac Asimov (1920-1992) wrote or edited more than 500 books and about 90,000 letters (Asimov, 1996)[2]. Many of his books include some form of AI. Artificial Intelligence as described by Asimov, almost 80 years ago, continues to astonish both science fiction readers and scientists as we grow closer and closer to the AI as portrayed in his books.

Aside from his success as an author, Asimov is also known for creating the Laws of Robotics. These laws are intended as a fundamental framework for robots with any form of autonomous artificial intelligence. The famous *Three Laws of Robotics* (Asimov, 1942)[3] published in 1942, are:

1. A robot may not injure a human being or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as said protection does not conflict with the First or Second Laws.

In 1986 a zeroth law was introduced:

0. A robot may not injure humanity, or, by inaction, allow humanity to come to harm. (Asimov, 1986)[4]

From the time that these laws were introduced by Asimov, scientists have kept them in the back of their minds, as many agreed this would ensure that new forms of robotics do no harm to humans. These laws were some of the few lines taken from literature which would also be taken seriously in science.

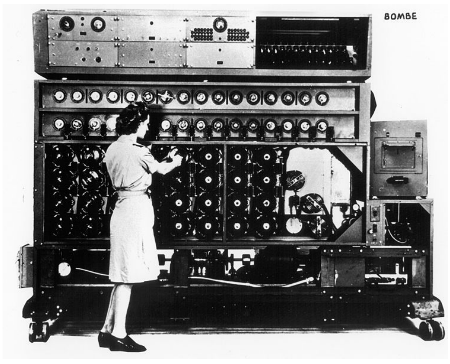


Image 1.1: Machine called 'Bombe' by Turing

Alan Turing (1950)

Alan Turing (1912-1954) was an extraordinarily talented and intelligent English mathematician, computer scientist, logician, cryptanalyst, philosopher and theoretical biologist (Oxford University Press, 2017)[5]. Turing is widely considered to be the father of theoretical computer science and artificial intelligence (Beavers, 2013)[6]. He is known for his instrumental work in breaking the German Enigma code leading to the Allied victory over Nazi Germany. Estimates say that Turing's contribution shortened the war by two years, saving almost 14 million lives (Copeland, 2012)[7]. Turing managed this by creating an electromechanical machine he named the Bombe in 1940 (see Image 1.1 (NSA, 1940)[8]). This machine was able to break the daily changing German Enigma code in a matter of minutes instead of a matter of weeks/months. This highly complex form of electronic problem solving kick-started the further development of computer advancement.

Turing Test

In 1950, Alan Turing developed the Turing Test. This is a test of a machine's ability to display intelligent behaviour similar to, or indistinguishable from, that of a human. A human evaluator (C) would judge natural language conversations between a human (B) and a machine designed to generate human-like responses (A) (see Image 1.2 (Margallo, 2017)[9]). If the evaluator is unable to reliably tell the machine from the human, the machine has passed the test. For the machine to pass the test, it is not a matter of giving correct answers to the questions, only how much the answer resembles an answer a human would give. Turing introduced this test in his 1950 paper; "Computing Machinery and Intelligence" (Turing, 1950)[10]. It opens with the words: "I propose to consider the question, 'Can machines think?'" This is, of course, one of the most relevant questions being asked in the field of AI.

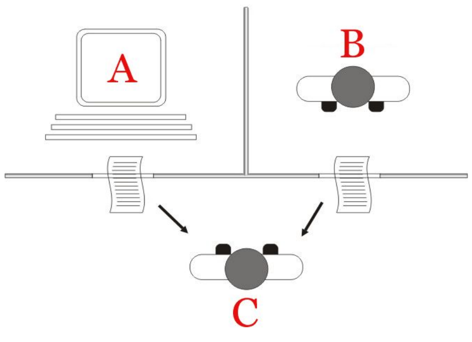


Image 1.2: Turing test with 3 participants.

Because "thinking" is a complex word to define, Turing came up with another version of the Turing Test, The Imitation Game. Originally, this game is played with three human players. Player A is a man, player B is a woman and player C (who plays the role of of the interrogator) is of either sex (see Image 1.3 (Férée, 1950)[11]). In the imitation game, player C is unable to see either player A or player B, and can communicate with them only through written notes. By asking questions of player A and player B, player C tries to determine which of the two is the man and which is the woman. Player A's role is to trick the interrogator into making the wrong decision, while player B tries to assist the interrogator in making the right one (Saygin, 2000)[12].

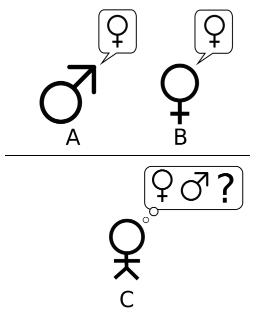


Image 1.3: The Imitation Game with people

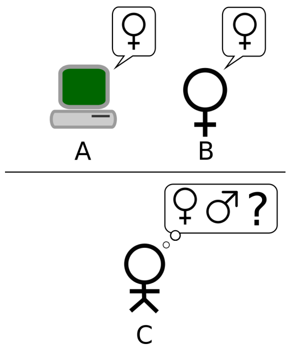


Image 1.4: The Imitation Game with a computer (Férée, 2011)

Turing then asks: "What will happen when a machine takes the part of A in this game (see Image 1.4 (Férée, 2011)[13])? Will the interrogator decide wrongly just as often when the game is played like this, as he does when the game is played between a man and a woman?". This question thus replaces the less concrete question: "Can machines think?"

Birth of Artificial intelligence (1952-1956)

Game-Playing Program

In 1952 Arthur Samuel wrote the first game-playing program. This program was able to play checkers (draughts) at an amateur skill level. Three years later, he created a version that learned to play. Looking back, both of these programs qualify to be called AI. However, at that time, the field of artificial intelligence did not yet exist. Even the term itself ("artificial intelligence") would not be created until the following summer during the Dartmouth Conference.

Dartmouth Conference

The Dartmouth Conference of 1956 was organized by Marvin Minsky, John McCarthy and two senior scientists: Claude Shannon and Nathan Rochester (McCorduck 2004)[14] (See image 1.5). The proposal for the conference included the following statement: "Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it." (McCarty, Minsky, Rochester & Shannon, 1955)[15]. Ray Solomonoff, Oliver Selfridge, Trenchard More, Arthur Samuel, Allen Newell and Herbert A. Simon all participated in this conference. All of them would go on to create important programs during the first decades of AI research and become important contributors in their respective fields (McCorduck, 2004)[16]. Newell and Simon displayed the Logic Theorist for the first time at this conference. The Logic Theorist was deliberately engineered to mimic the problem-solving techniques of a human being and has been called "the first artificial intelligence program" (Crevier, 1993)[17]. The use of this novel phrase, together with the persuasion of McCarthy to accept "Artificial Intelligence" as the name of the field, settled the naming of this new technology. At the 1956 Dartmouth conference, AI was named and its mission became clear. The conference also marked the first AI success and brought together the people instrumental in AI development. Consequently, the conference is widely considered to be the birth of artificial intelligence (Crevier, 1993)[18].

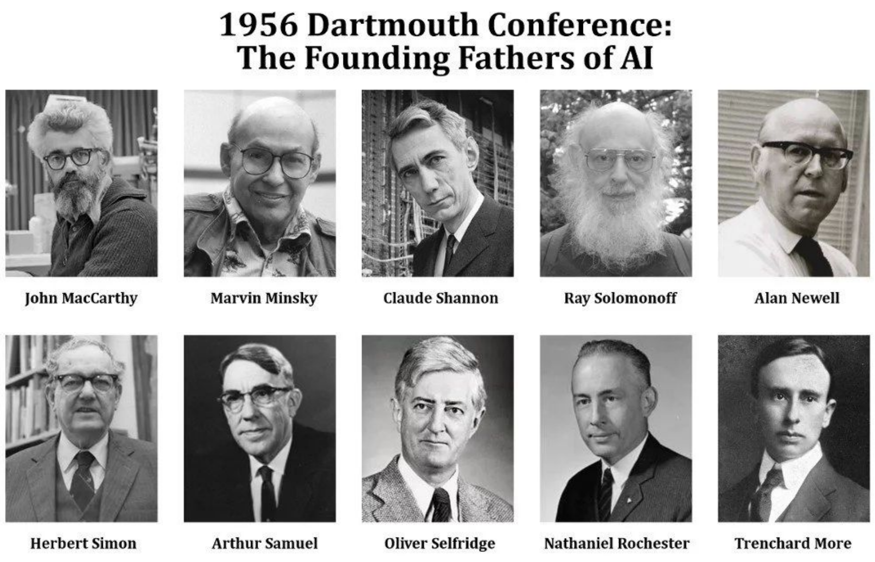


Image 1.5: Founders of AI

Golden Years for AI (1956-1974)

After the field of AI had officially been kicked off, a wave of all kinds of computer and cognition experts dove into the development of new programs and techniques. This surge of interest and activity in the field resulted in this period being called "The Golden Years for AI". Programs were written to solve algebra word problems, to prove theorems in geometry and to speak English. There was extreme excitement and optimism, convincing many researchers that a fully intelligent machine would be built in less than 20 years (McCorduck, 2004)[19]. Highly respected researchers of AI predicted the following:

- Simon & Newell (1958)[20]: "within ten years, a digital computer will be the world's chess champion" and "within ten years a digital computer will discover and prove an important new mathematical theorem."
- Simon (1965)[21]: "machines will be capable, within twenty years, of doing any work a man can do."
- Minsky (1967)[22]: "Within a generation ... the problem of creating 'artificial intelligence' will substantially be solved."
- Minsky (1970)[23]: "In from three to eight years, we will have a machine with the general intelligence of an average human being."

The first of these predictions did eventually happen. However, it took almost 40 years after this prediction for an AI to beat the world's best chess player at the time in 1997. The other predictions stating that an AGI (AI with the same cognitive capabilities as a human) would be created, have not yet come true.

Even though predictions by the leading experts of AI were not met in the timeframes they proposed, the years between 1956 and 1974 still proved very innovative. The most influential programs and new developments during these Golden Years were: Reasoning as search, Natural language and Micro Worlds.

Reasoning as search

Most of the first AIs all used the same algorithm. These AI programs use the same approach for all problems, using a method similar to how a human would walk through a maze. At each intersection (decision/option) arbitrarily picking a direction in which to continue. If the chosen direction leads to a dead end, it goes back and chooses a different direction. This way of problem solving was called "reasoning as search" (McCorduck, 2004)[24]. However, constantly pushing AI to its limits and giving it increasingly more difficult problems, revealed a bottleneck. In some cases (for example, an AI that plays checkers), the maze would be so large, with so many junctions, that this type of brute force method was highly impractical. Researchers would try to reduce the search space (make the maze smaller) by using heuristics or "rules of thumb". This method would eliminate paths that were unlikely to lead to a solution. As time went on, these methods became increasingly reliable and efficient. Simon and Newell attempted to create a general program of this

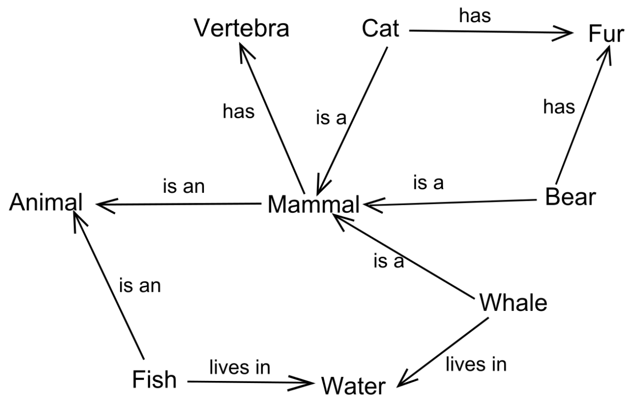


Image 1.6: Example of a Semantic net

algorithm called the "General Problem Solver" (McCorduck, 2004)[25](GPS). This was the first computer program which separated its *knowledge* of problems from its *strategy* of how to solve problems. However, even though the technique used by the GPS was revolutionary, due to the lack of computational power at the time, the GPS itself never did solve any real world problems.

Natural language

From the first day of AI’s existence, researchers have had the goal to make AI communicate in natural languages like English. The first real success in this area was a program called STUDENT by Daniel Bobrow. This program was able to solve high school algebra word problems (Russell & Norvig, 2003)[26]. Bobrow was able to accomplish this by using something called a semantic net (see Image 1.6). A semantic net, as first used in a program written by Ross Quillian (Crevier, 1993)[27], represents relations between concepts in a network. In this example (Anonymous, 2007)[28] "is an" and "has" links are used to define "Mammal" in relation to other concepts shown in the image.

Two years later, Joseph Weizenbaum created a program called ELIZA. ELIZA could carry out conversations that were so realistic that users were occasionally fooled into thinking they were communicating with a human being instead of a program. In reality though, ELIZA did not actually know what she was saying. She mainly gave predetermined responses to common questions or repeated what was said to her, rephrasing her response with a few grammar rules. ELIZA inspired the development of a new type of AI called the chatterbot, of which ELIZA was the first (McCorduck, 2004)[29].

Micro-worlds

In other sciences, for example physics, many principles are often best understood using simplified models. For example, frictionless objects or perfect constant speeds. These simplified models of the world are known as micro-worlds. In the late 1960s, Marvin Minsky and Seymour Papert of the MIT AI Laboratory, decided to apply this technique to AI. In some cases, doing so allowed AI to train much faster and inspired a new perspective towards training an AI. Their research focused on a "Blocks World" (Crevier, 1993)[30], consisting of colored blocks of various shapes and sizes on a flat surface. Blocks World was usually used for training AI in planning, as seen in Image 1.7. Gerald Sussman used Bocks World to make huge leaps in machine vision, making AI that could "see" (recognise images) far better than anything made before. Another big leap at the time, making use of Blocks World, was a robot arm capable of stacking blocks. This advancement brought the Blocks World into "the real world". The greatest achievement at the time, using micro-world structures, was SHRDLU by Terry Winograd. SHRDLU was able to communicate in ordinary English sentences, construct plans and execute them (Crevier, 1993)[31].

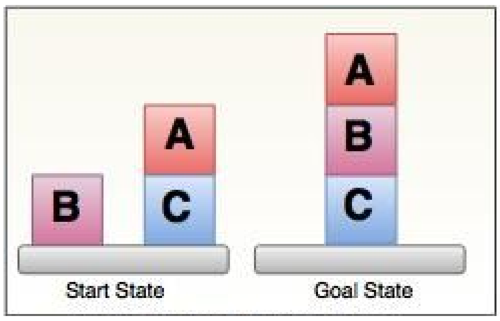


Image 1.7: Blocks-World planning problem

The first AI winter (1974-1980)

The complexity of the field of AI was the cause of a six year stagnation in all AI research. The researchers underestimated its complexity and were unable to achieve nearly impossible goals that were set in the past (for example the predictions/goals mentioned in "Golden Years for AI"). This resulted in almost all funding being cut, as investors were unable to understand why the progress had been slow (Russell & Norvig, 2003)[32]. Many of the unachieved goals were not a result of the wrong code, or flawed reasoning, but rather the result of a lack of computing power at that time. Drawing nearer to something comparable to the human brain required immense computing power. For example, due to the maximum memory capabilities at the time (Crevier, 1993)[33], the highly successful work on natural language by Ross Quillian, was only able to run with a small vocabulary consisting of just 20 words. Scientists were exploring new concepts and objectively making progress, but there was no real application for anything created.

Return of interest in AI (1980-1987)

In 1980, the development of "Expert Systems", the first truly useful AI, attracted the attention of many corporations around the world. This shifted the focus of AI research towards Knowledge Representation and Reasoning (KR&R). The revival of connectionism inspired many researchers to return to the field. The Japanese government started to pour money into an AI initiative called the "Fifth Generation Computer". All of these factors led to a new era within the AI timeline, a short but successful boom in development, where interest had finally returned.

Expert Systems

The first successful and useful AI was an Expert System called XCON (eXpert CONfigurer). XCON had many defining features. As many AI programs at the time tried to come close to what a human was capable of in a specific field, for example basic speech or being able to play a board game at amateur level, XCON had a different goal. XCON was developed to outperform humans at a specific task. Achieving this goal is also what ultimately made Expert Systems such a success, as companies would benefit greatly from this. XCON was made possible by the groundwork which was laid down by Edward Feigenbaum and his students. They created simpler Expert Systems like Dendral (1965, identified compounds from spectrometer readings) and MYCIN (1972, diagnosed infectious blood diseases). These programs would prove the effectiveness and possibilities that Expert Systems were capable of (Newquist, 1994)[34]. Expert Systems relied on two main components. First of all, they would be restricted to a very small domain of specific knowledge. This would allow for faster computing and simpler calculations, as no general understanding (which could be made by semantic nets) was needed. Secondly, ‘Expert Knowledge’ was condensed into if-then rules which formed the basis of the program. At the time, procedural code was much more common as a program structure. Procedural code was simply a series of computational steps to be carried out. Rule-based systems would be able to solve seemingly much more complicated problems. For example, helping a doctor choose the correct diagnosis based on any number of symptoms. Here the rule-based system would underscore a program that would follow a set of rules to derive an answer. Conversely, programs using procedural code would less efficiently, try to find a single calculation that factors in all the variables to arrive at an answer.

Using rule-based systems to solve complex problems was the start of AI taking over jobs. Expert Systems proved to perform better, faster, longer and most importantly, cheaper. XCON, developed for the Digital Equipment Corporation, was saving the company 40 million dollars annually by 1986 (Crevier, 1993)[35]. By 1985, corporations around the world were spending a total of over a billion dollars developing Expert Systems (McCorduck, 2004)[36].

Knowledge Representation and Reasoning

KR&R is the field dedicated to representing knowledge or information in a way that a computer can understand. From the beginning of the 1980s KR&R regained the focus of research. This allowed for huge improvements in Knowledge-Based Systems (KBS). KBS are computer programs that use a knowledge base (KB) to solve complex problems. A KB is used to store two types of information/data: structured data and unstructured data. Structured data is highly organized and is often well understood by a computer, such as a spreadsheet with names and dates. Unstructured data is more complex for computers to read and understand, such as audio recordings or images (Pickell, 2018)[37]. The key to KR&R is to have all data in a KB function properly. A KB is applied in many types of AI. It is also one of the two components of an Expert System. The other component being Inference Engine. The Inference Engine is a system that applies logical rules to the information in a KB to deduce new information.

Connectionism

Connectionism became increasingly popular in the late 1980s. This was the result of two main victories in this field at the time. John Hopperfield proved that a form of neural network could learn and process information in an entirely new way. Also, David Rumelhart and Geoffrey Hinton popularized a method for training neural networks called "Back Propagation (BP)". BP is an algorithm used for training AI, which would later form a crucial component of Deep Learning. The field of connectionism focuses on trying to explain mental phenomena using Artificial Neural Networks (ANN) (Garson, 2018)[38]. While ANN had been used many times in the past, the 1980s proved to be a landmark period for the evolution of this field. The development of the Metal–Oxide–Semiconductor (MOS) and Very-Large-Scale Integration (VLSI), enabled the development of the first practical ANNs.

Fifth Generation Computer

After the first AI winter, the Japanese government was one of the first to start investing large amounts of money into research in AI. The Fifth Generation Computer System (FGCS) project, launched in 1982, is what caused the Japanese Ministry of International Trade and Industry to invest $850 million in the field of AI research. The goal of this project was to create a supercomputer for its time, able to carry out conversations, translate languages, interpret pictures, and reason like human beings (Newquist, 1994)[39]. The Japanese believed that this could only be achieved by combining state-of-the-art hardware and software. They hoped to achieve this by using many CPUs for parallel computing, allowing all CPUs to work/compute together, and programming the computer in Prolog, as opposed to the mainstream LISP, which the Americans used at the time (Crevier, 1993)[40]. However, these "innovations" were also part of what caused the eventual failure at the time. The parallel computing proved to be inefficient and the use of Prolog resulted in only a small number of expert level programmers who were able to work on the project. Thankfully, the failure of the FGCS still brought its advantages. After seeing Japan invest $850 million, attempting to become the leading county in AI development, many other countries responded with investments of their own. The UK invested £350 million in the Alvey project, which had a similar goal to that of FGCS. The Defense Advanced Research Projects Agency (DARPA), invested as well. DARPA is an agency of the United States Department of Defense responsible for the development of emerging technologies for use by the military. They started the Strategic Computing Initiative (SCI), designed to support various projects in the field of AI. DARPA tripled its investments in AI between 1984 and 1988 (National

Academies of Sciences, Engineering, and Medicine, 1999)[41], spending a total of $1 billion on the SCI from 1983 to 1993 (Roland & Shiman, 2002)[42].

The second AI winter (1987-1993)

Once again, the abundance of investments meant that expectations were on the rise. As the investments were increasing, the researchers who survived the budget cuts at the start of the first winter started to suspect expectations would rise above reality. These researchers were correct (Crevier, 1993)[43]. Expert Systems, such as XCON, proved to be too expensive to maintain. And as Expert Systems like these definitely proved useful in very specific circumstances, they also proved to be challenging to maintain. Also, all Expert Systems at the time struggled greatly with unusual inputs and lacked adaptability. DARPA cut its SCI and decided that AI was not "the next wave" (McCorduck, 2004)[44]. By the end of 1993, over 300 companies had shut down or gone bankrupt. This marked the end of the first commercial wave of AI (Newquist, 1994)[45].

Continued AI development (1993-2019)

Due to rapidly increasing computer hardware technology, like memory and processors, combined with the steady continuation of AI research, AI was once again gaining momentum. Finally, in 1997, an AI had beat the world chess champion (McCorduck, 2004)[46]. This AI, Deep Blue, which was developed by IBM, had finally fulfilled the prediction made in 1958. And even though this event had been predicted to happen almost 30 years earlier, it had eventually come true. Milestones like these continued to happen throughout the following decades. More powerful computers resulted in more milestones being accomplished. This resulted in media coverage and more attention to AI. Ultimately, this resulted in more funding of AI research and more researchers contributing to AI development. This amplified the cycle by increasing the number of new developments, and boosting AI's popularity. Deep learning evolved from Machine Learning when combined with big data. Big data was the result of the exponential growth of the internet and the number of computer users. This ubiquitous computer use allowed large amounts of data to be collected which could then be used to train Machine Learning AI. Deep Learning became the subset of Machine Learning which uses big data to train the AI. In recent decades, researchers have found many new ways of creating better, faster and smarter AI.

Summary: History of AI

Before AI existed, imaginative and speculative authors wrote stories about computers which were able to learn, and sometimes even become smarter than humans and would ultimately form a threat. However, once real AI started to be created, it was still quite a bit less spectacular. In 1950, Alan Turing permanently changed the way people looked at computers by asking the question "Can machines think?" and proposing to answer that question with the Turing Test. Turing's revolutionary research would later earn him the title of father of theoretical computer science and artificial intelligence. Six years later, during the Dartmouth Conference of 1956, this new field of research would officially gain the name Artificial Intelligence. AI was born. This conference was the kick-off for almost two decades of growth of the field. Many new investors were excited about the potential of AI, and many researchers were interested in making new discoveries in this young field of research. However, these Golden Years would come to a stop in 1974, when the first AI winter would strike. Investors had lost hope since many expectations had not been met. In 1980, interest would return as the first AI was created which was available (and useful) for public use. This would mark the first AI which would take over human jobs. However, in 1987, these AI would prove not yet ready for implementation in corporate life. At the time, AI were still too expensive, difficult to keep up to date, and not flexible enough. This disappointment would send the development of AI into another winter, lasing until 1993. Rapidly increasing computer capabilities would result in AI finally being able to achieve goals which were set decades before. In these last decades of AI, development of many new techniques were developed. To understand the current state of affairs concerning AI development, The Basics of AI will explain how all AI work today, and what methods are used to create an AI.

The Basics of AI

To understand how to create or use AI, you need to have an understanding of the terminology and basics behind Artificial Intelligence. These basics are divided over four subquestions:

1. What is an AI?
2. What are the divisions of AI?
3. How does an AI work?
4. What do you need to create an AI?

The information gathered by answering these four questions give a basic understanding of the field of AI research and development.

Subquestion 1: What is an AI?

AI is often referred to, in a broad sense as, “A machine that simulates human intelligence processes”. These processes include: learning, problem solving, reasoning and self-correction. The unclear lines of what defines an AI are the result of increasingly smarter and more complex computer systems. A program that generates a heads or tails result based on a randomly generated number is not an AI. But how about a program that can play the game Snake? And what about handwritten text-recognition or self-driving cars?

There are also two ways to define an AI. You can look at it either from a Function Perspective or a Code Perspective.

- **Function perspective**: The function, what a program does, is decisive in this classification. If a program does something that is considered intelligent, then it simulates human intelligence and is an AI. This perspective is rather subjective.

- **Code perspective**: It does not matter what the program does, the code the program consists of determines whether it is considered an AI.

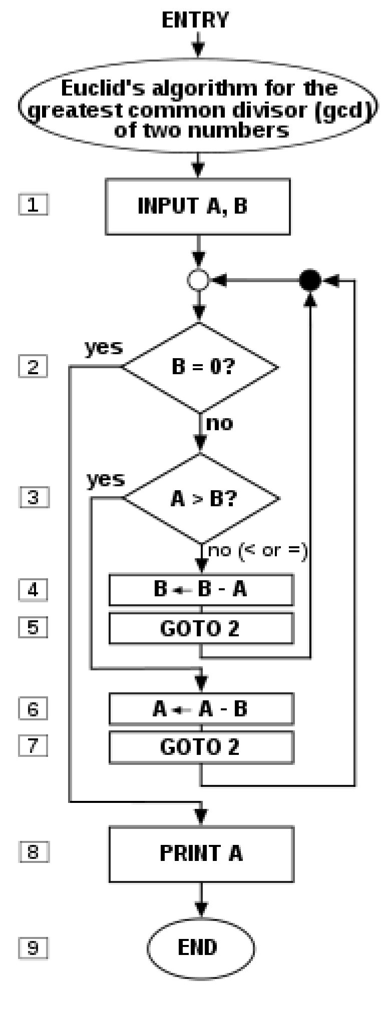


Image 2.1: Euclid's Algorithm

Algorithms

Algorithms are a crucial part in coding. Almost every computer program uses them. However, algorithms themselves do not require a computer. Greek mathematicians used algorithms for finding prime numbers, and used the most well-known algorithm, the Euclidean Algorithm, to find the greatest common divisor of two numbers.

To explain how an algorithm works and what it could look like, the Euclidean Algorithm (see Image 2.1) is a good example.

1. An algorithm starts with an input. In this case, the input (variable A and B) consists of two positive integer numbers.
2. The input is tested to see whether variable B is 0. If that would be the case, the algorithm would jump to step 8 (follow the arrow) and print (display) variable A. If you would put this to the test and take the numbers A=4 and B=0, the greatest common divisor would be A, meaning 4.
3. If B does not equal 0, the algorithm would continue to step 3. Here it would check to see if A is larger than B. If that is true, it would continue towards step 6 and 7. If the statement is false (so no) it would continue to 4.
4. At this step, variable B is changed. The result of B-A is stored as the new B.
5. The algorithm is sent back to step 2 to repeat the process until B=0.
6. This step is similar to step 4. However, step 6 is for when A is larger than B. Variable A is changed in this step and the new A is A-B
7. The algorithm is sent back to step 2 to repeat the process until B=0.
8. The alterations of A and B have resulted in B=0 at step 2. The algorithm can now continue to step 8 and print A
9. The algorithm has successfully completed the process and has found the answer to the question is was created to answer.

Euclid's Algorithm is a very basic algorithm. It demonstrates how a set of "If this then that" statements constructed in the correct manner can compute a complex mathematical question, while each step in the calculation is very simple and rudimentary. Algorithms allow for endless possibilities and are therefore the building blocks that almost all programs rest on, including AI.

A good example of what an AI is and what is just an algorithm is two different parts of YouTube: The monetization system and the recommendation system. The monetization system works with an AI. The AI scans all YouTube videos and thumbnails to decide whether the video can be monetized. It does this based on the knowledge it has, which is trained by trainers who manually review videos. With the data from these reviews, the AI determines what has monetization potential and what doesn't and uses this data to perform the automatic video reviews.

The recommendation system is an algorithm that YouTube uses to recommend videos to users. This is not an AI because it's an algorithm that has multiple variables. After entering the variables (previously watched videos, current trends, demographic data of the user, etc.), the algorithm suggests multiple videos to put in the recommendations. This system doesn't think for itself, it simply follows instructions. Therefore, it is not considered an AI.

Difference Algorithm and AI

To create a clear image of the difference between an algorithm and an AI, a social media example can be used. Imagine a platform like Instagram. This is a platform that allows users to upload and "like" images. The goal of the platform is to keep the user entertained and drawn to it for as long as possible. If a user has "liked" an image of a cat, thus displaying an interest in that image, the platform will keep this logged. However, if the log is only "user X likes image Y" then the platform has no further useful instructions or information to try to keep the user on the platform because Y is not defined.

Now imagine a "tag" system. When users upload an image of a cat, a tag called "cat" is added. If a user likes an image with the tag "cat", the platform can store useful information. A user who likes an image with a cat tag has a larger chance of liking another image with a cat tag than a random image.Therefore, when the platform is capable of recommending appealing videos, the user is more likely to stay entertained on the platform for longer. The information provided by the tag allows the platform to introduce an algorithm which can make the appropriate recommendations.

This algorithm will function in the same sense as Euclid's Algorithm. Depending on the input, an output is generated. In this case, the input is the tag "cat". The algorithm would then be programmed to display more images with this tag. This process has already been pre-programmed. The basic structure which has then already been made is: If tag[X] is liked, then display more images with tag[X]. This results in continuing to show the user images of subjects that the user is interested in.

However, this setup poses a problem. The user might get bored. The way this algorithm is set up is relatively "closed". A way to introduce users to new tags with a high chance of interest is by using an AI. Especially in a constantly changing environment like social media, AI can be a very powerful tool. AI could recognize patterns and act according to algorithms that haven't been programmed. New trends and unprogrammed correlations could be detected and could be used to recommend new content (in this case, different tags) which could lower the risk of the user being uninterested in the recommendation.

The AI effect

The AI effect is when people discount the behavior of an AI by arguing it is not *real intelligence*. A lot of people see AI as a difficult principal. This results in the perception that when a computer solves a difficult problem, it is often seen as the work of an AI. This is the same the other way around. When a computer does something that is seen as common and is done regularly nowadays, it is often seen as the result of algorithms, although it was done by an AI. This means that things are only seen as an AI when they are solving difficult problems. Otherwise "Intelligence is whatever machines haven't done yet." (Maloof, 2017)[47] What counts as AI is difficult to define, because many different scientists and researchers have differing opinions on it.

Conclusion: What is an AI?

An AI is a program or machine that exhibits human-like aspects, like decision-making and communication. An AI consists of complex algorithms containing a series of formulas and checks. The algorithm gets an input and gives back a response, an output. Yet, an algorithm on its own is not an AI. It is the use of complex algorithms in a system or program which performs a human-like function that is defined as an AI.

Subquestion 2: What are the divisions of AI?

AI divided in stages

Some AI are more capable than others. An AI can be placed in a stage based on the number of tasks that an AI can fulfil and the level at which it can perform those tasks.

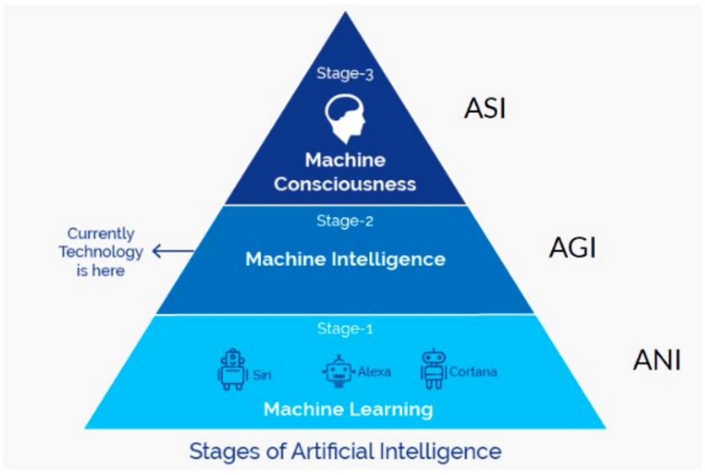


Image 2.2: Stages of AI

As can be seen above in Image 2.1, AI are separated into three stages; Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI). Current technology makes it possible to create AI up to an AGI (see Image 2.1). Many ANI already exist as do parts of an AGI. However, a fully functional AGI has yet to be made.

Artificial Narrow Intelligence (ANI)

All AI that are used nowadays are ANI. They are made to perform a specific function quickly. These ANI are often optimized in these functions as it is their only task. However, ANI are still only able to perform specific tasks. They have a narrow specialization and often do not have a lot of parameters. They are not necessarily slower than AGI and ASI, they just can't perform as many tasks.

Artificial General Intelligence (AGI)

AGI is often seen as the ultimate target for AI. AGI can perform all the human intellectual tasks. It has no consciousness but can think and act logically. It will be able to perform basic human intelligence processes, for example learning, reasoning, planning and problem solving. Humans will still be necessary when AGI is accomplished because elements of the human consciousness that an AGI misses are still necessary in society (mostly in the form of creativity).

Artificial Super Intelligence (ASI)

ASI can do anything that a human can do and can do it better, faster and more consistently It also reaches a state of consciousness. The moment that ASI is achieved, humans won't be necessary anymore. That is because every task that a human can do, can also be achieved by an ASI.

Below is a brief overview for clarity.

Stages	ANI	AGI	ASI
Name	Narrow	General	Super
Also referred to as	Weak	Strong	Conscious
Compared to Human	Ability to mimic human intelligence and/or behavior in a narrow range of parameters	Ability to mimic human intelligence and/or behavior indistinguishable from that of a human	Doesn't mimic human intelligence and/or behavior but surpasses it
Progress	Has already been done	Does not exist but is being developed	Does not exist and is only a concept
Uses (or expected uses)	Siri, Alexa, Cortana	Jarvis (fictive, Avengers movie)	Skynet, Ultron (fictive, Terminator/ Avengers movie)
Effect	Jobs enhanced	Jobs at risk	Humanity at risk
Timing	Today	About 2060 (AIMultiple, 2019)[48]	Unknown

AI in relation to Machine Learning and Deep Learning

AI consists of different subsets which each have different ways of functioning. The subset in which an AI is made depends on what it will be used for and what type of data it is fed. Machine Learning (ML) being a subset of AI, and Deep Learning (DL) being a subset of ML (as can be seen in image 2.2) (Garbade, 2018)[49].

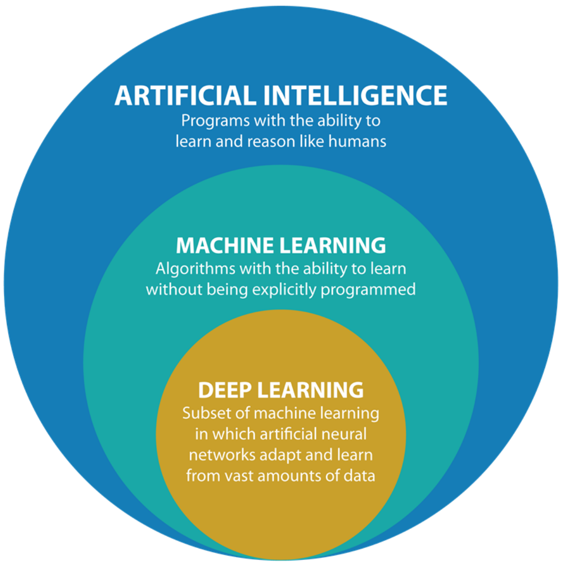

ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Image 2.3: AI, ML and DL

Using ML, you can create a constantly changing and refining program that teaches itself to improve (Martin, 2019)[50]. Around 1980, 30 years after the start of AI, ML created possibilities that had never been before. ML starts out as a relatively useless and empty program, not knowing what to do. Once a goal is defined, the program strives to accomplish that goal, or in most cases, to get as close as possible. Once you let it run (and fail because it doesn't know how to accomplish the goal yet), it analyzes how close it was to the goal. Then it tweaks its own parameters which causes the second run to yield a different result. If it gets closer to the goal, then the tweaks were positive and mostly the same parameters will be tweaked in the same direction. If getting to the goal was less successful, different parameters will be tweaked. More runs will result in a better trained program. If you look at an example of an ML program made to play Snake, the goal is to become as large as possible. During its first run, it will just go into the wall in a straight line. After a couple of tries, it might (accidentally) run into a piece of food. The "food score" increases and the program recognizes that the previous run was significantly better (closer to the goal) than previous runs. After many runs, it will learn to steer clear of walls, watch out for itself and focus on the food. None of these actions have been pre-programed into the program, yet it knows the rules of the game. This way of programming a Snake playing program does not make much difference from hard-programming like: if the distance to a wall is closer than in the previous frame and is smaller than 2, then turn left. However, many systems, for example programs that recognize spam emails, do not have defined values like that. You cannot say that all emails with a word like "free" are spam. In many cases, it depends on the context. It is impossible to pre-program each different email with a classification of "spam or no spam". Here ML is almost unmissable. Training and perfecting itself along the way, it defines its own rules depending on whether the flagged spam email is kept as spam, and whether emails that are not flagged as spam are marked as spam by the user.

DL is a subset of ML which was first used around 2010. It uses the same principles as ML. These principles are Neural Networks (NN). NN are the layers of parameters and calculations that are tweaked in ML. DL is known to use deep NN in contrast to ML using standard NN. Deep NN is usually an NN which uses three or more "hidden" layers(Samson, 2017)[51]. In general, DL is trained by a large dataset at the start and only improves a little after that, whereas ML trains along the way with each separate run. A visualization of how a deep neural network sees is shown in Image 2.4.

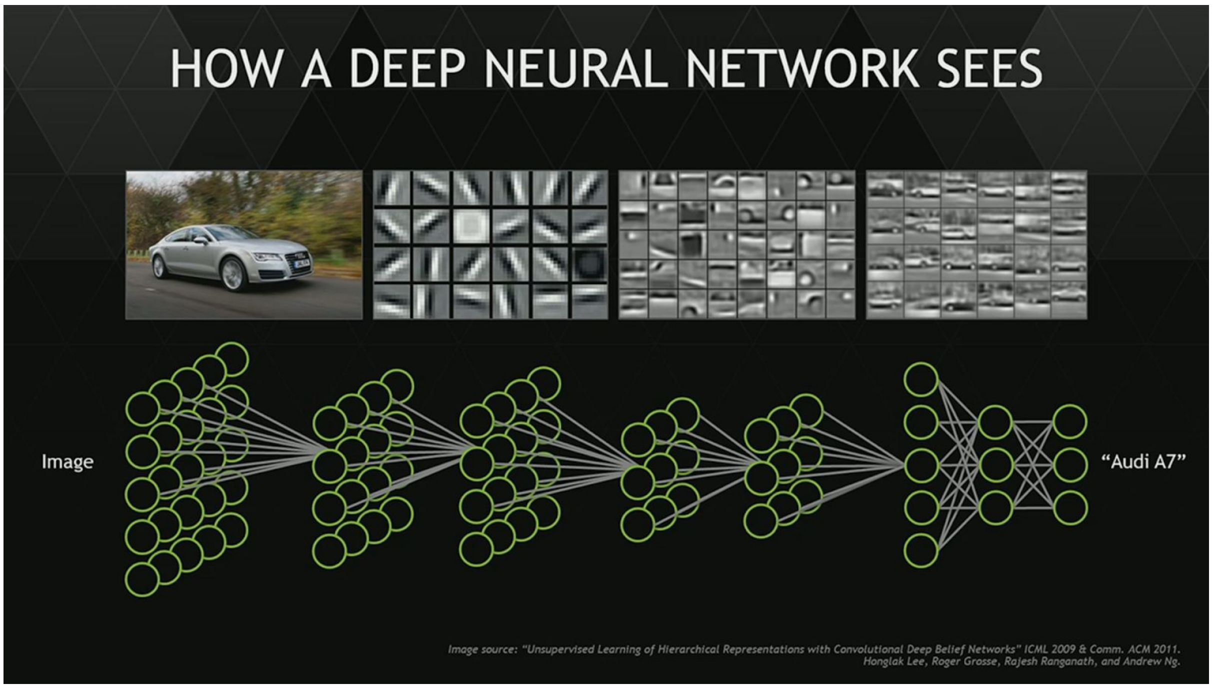

HOW A DEEP NEURAL NETWORK SEES
Image
"Audi A7"
Image source: "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks" ICML 2009 & Comm. ACM 2011.
Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Ng.

Image 2.4: Neural Network Example

Learning Methods

There are multiple approaches that can be used when training an AI. The approach that you want to use depends on which input data you have, which output data you want, and the type of problem that you want to solve. These three criteria need to correspond with each other to achieve the expected result.

Supervised Learning

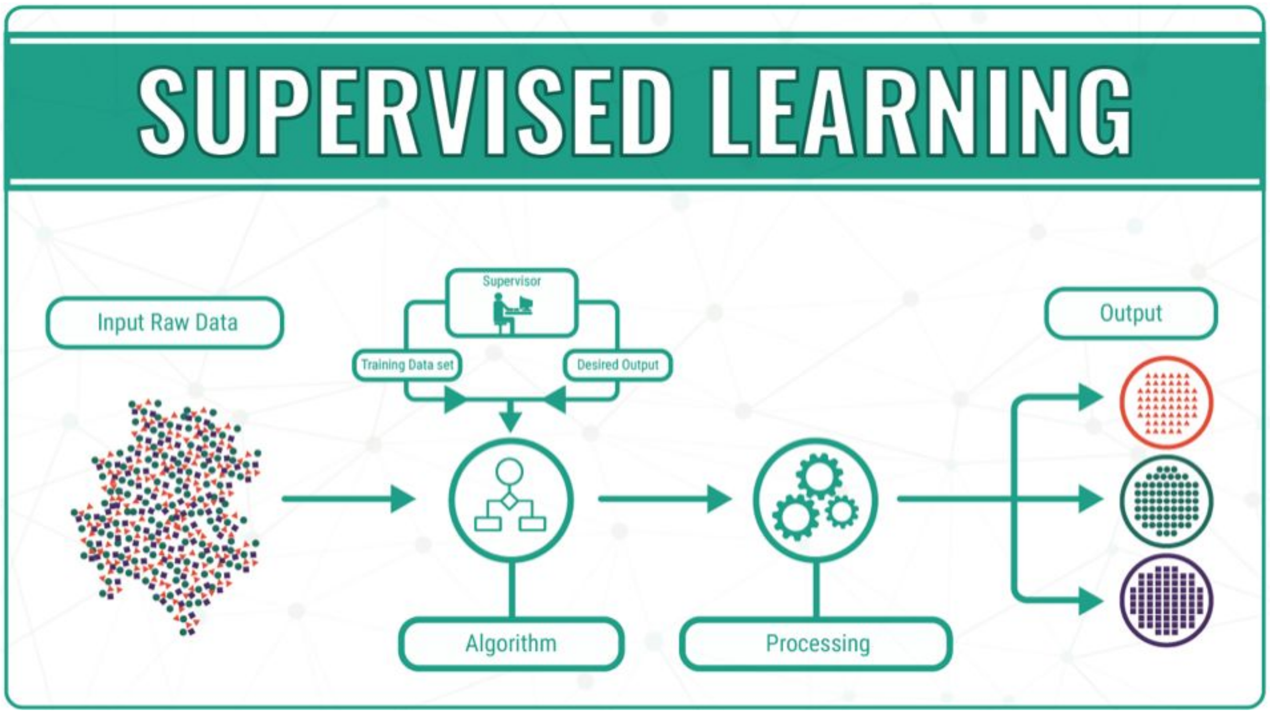


Image 2.5: Supervised Learning

With supervised learning, there is a corrector or supervisor that tells the algorithm what the desired answer is; this can be seen above in Image 2.5. The input data will be processed into output data by the AI, the AI will learn from the difference between the generated output data and the correct output data. If those two differ too much, then the AI will change the way that it processes the data with the hope of achieving a result closer to the correct output data. This will be done many times to increase the chance of generating the correct output data. In the image above, the supervisor is represented as a human but in reality, it is mostly done by feeding the AI a labeled data model. This is essentially the raw data as input as well as the desired output. The AI will try to generate the output. Once the initial output is generated, it will correct itself with the desired output. It is based on the same principle as when a human does the correction, but in this way, it does not require human action and can learn much faster (Russel, 2010)[52].

This technique is mostly used when there is a clear view of the desired output. For example, with digitizing handwriting, it is clear what the desired output is; the same letters in digital text as there are in written text. In these cases, it is useful to use supervised learning. This is because it is fairly easy to give the AI a handwritten text to digitize together with the desired digital text. In this scenario, the desired text is typed by a human so the AI can learn from this correct output.

Unsupervised Learning

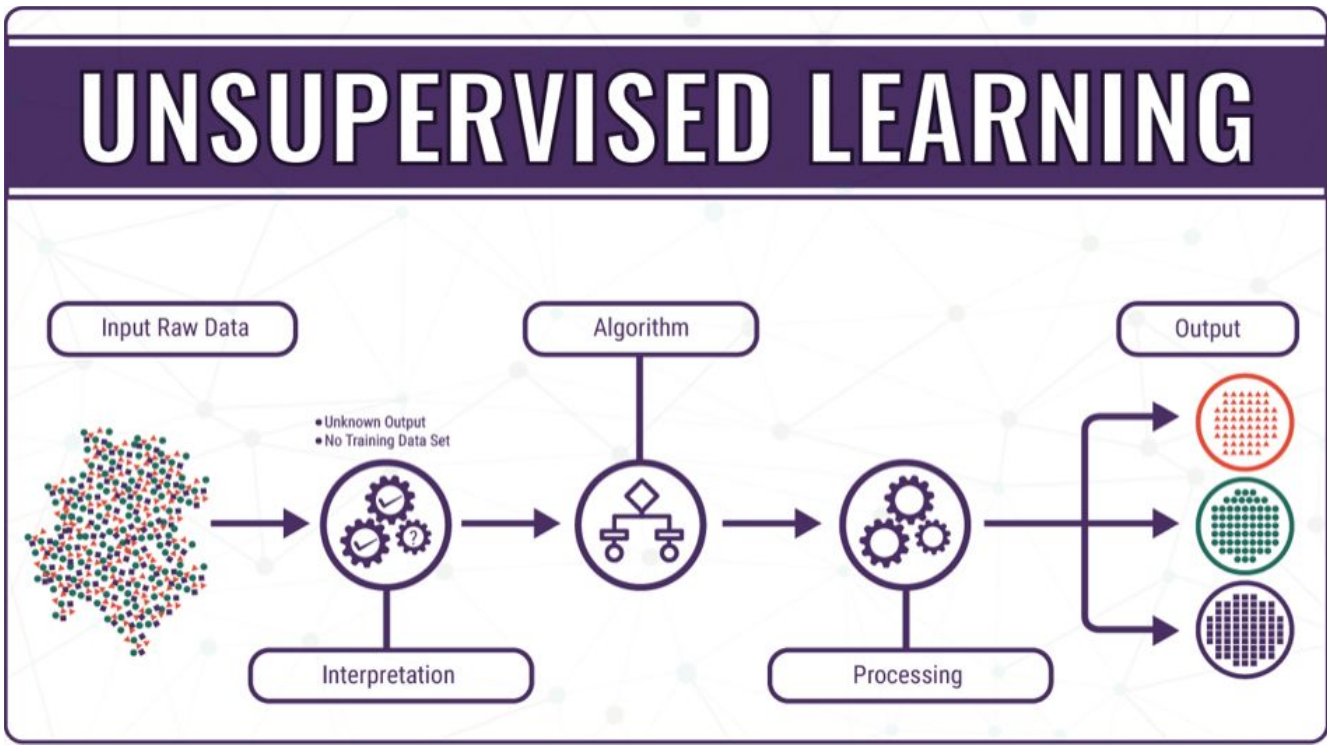


Image 2.6: Unsupervised Learning

As can be seen in Image 2.6, with unsupervised learning, there is no corrector of any kind. The AI gets a data set as input and has no specific output to give. The only output it should give is finding structure in the data sets, which is done by clustering or grouping data points. The algorithms identify similarities and react to the absence or presence of these similarities in new data sets. In this way, the AI finds correlations in the data sets (Russel, 2010)[53].

This technique is often used in fraud detection (Horan, 2018)[54]. Fraudulent transactions are often different than normal transactions (in terms of time and amount). The AI finds the abnormal behavior by looking for patterns.

Reinforced Learning

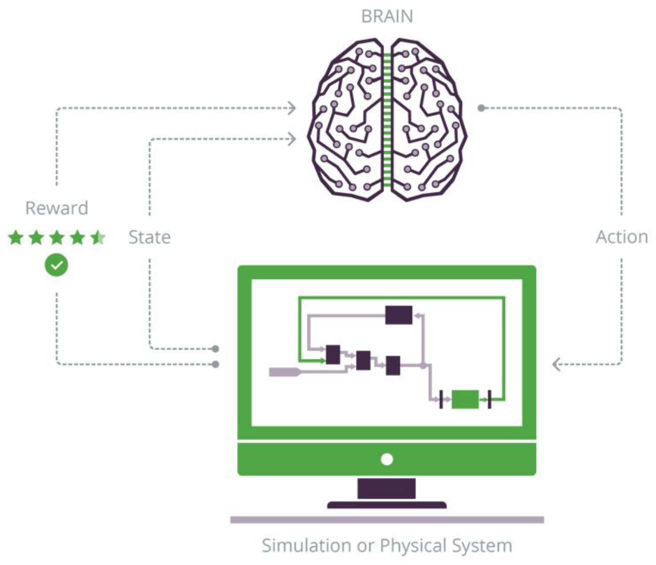


Image 2.7: Reinforcement Learning

With reinforced learning, the AI learns from reinforcements or punishments (the rewards are can be seen as the number of stars in Image 2.7 above). It only knows that it must do something differently when punished and to do the same when rewarded. In this way, it will learn every time it is punished or rewarded, so that it does better the next time. It must (more or less) repeat the process when rewarded and it must change its process when punished (Russel, 2010)[55].

This technique is used in many ways, such as in games. It is, for example, used for an AI that must battle in a game against a human opponent. The AI is rewarded when it wins a match and punished when it loses. In this way, the AI learns from the previous matches how to play better in the next one.

Conclusion: What are the divisions of AI?

AI can be categorized in two ways: based on its level of intelligence, or based on the applied learning approach. When categorized by intelligence, there are three types of AI: ANI, AGI, and ASI. When AI is categorized by the learning approach used, the three types are: Supervised learning, unsupervised learning, and reinforcement learning.

The levels of of AI intelligence represent how advanced the AI is and how closely it resembles human intelligence. The first and simplest type of AI is ANI. ANI is an AI used for specific functions, like detecting an object in an image. All AIs existing today are ANIs. The second and more complicated AI, which does not yet exist, is AGI. AGI is able to mimic human intelligence and/or behavior indistinguishable from that of a human. The last and most complex type is ASI. ASI is able to surpass human intelligence and possesses consciousness. This level has also not been achieved yet and brings with it a danger to humanity if it is ever attained.

The three types of learning methods used for AI each have their own advantages and disadvantages. The appropriate method is selected based on the desired function of the AI once it is done learning. Supervised learning is useful when working with a clear output, for example when digitizing handwriting. Unsupervised learning is used when patterns have to be found within a large dataset. This technique is used, for example, when trying to find fraudulent transactions on bank accounts. And reinforcement learning is useful when the input is on a broader scale than a few numbers, like when a bot plays a game and there are multiple possible inputs.

Subquestion 3: How does an AI work?

Approaches for AI

Because there is no established theory that guides AI research, different researchers decide to approach AI from different fields of research or theories. The question “How *does* an AI work?” is often phrased as “How *should* an AI work?”. Different researchers think differently about this question and seek to answer it in different ways. They ask questions like: Should artificial intelligence simulate natural intelligence? Or is human biology irrelevant to AI and should we therefore not try to connect the two fields of research at all? This causes many different AI approaches to be used in AI research and development.

Some of the most used approaches are:

Symbolic

Symbolic AI is the term used for all approaches in artificial intelligence research that are based on symbolic representations of problems. This means that the problems are able to be read by human beings, like certain numbers, names, words or icons. The input is definite and certain. (Haugeland, 1985)[56]

Sub-symbolic

The sub-symbolic approach uses not only symbolic ways of representing knowledge and ways for the AI to process tasks, but also uses other approaches that are not symbolic. This is because researchers realized that symbolic AI wouldn’t be able to imitate all types of human cognition, like perception, learning and pattern recognition. (Nilsson, 1998)[57]

Cognitive Simulation

An approach developed by using the results of psychological experiments to develop programs that simulate the techniques that humans use to solve problems. (McCorduck, 2004)[58]

Logic-based

Logic-based is an approach that claims that a machine should think by using logic. This means that the AI is able to do complex and abstract thinking without using the same method used by humans. The AI should use logical thinking to solve illogical problems and complex situations.

Anti-logic

Anti-logic differs from other approaches because it doesn’t use one concept or principle for every situation/problem. Anti-logic is based on researching and building each part of the machine to handle a different type of problem solving.

Knowledge-based

Knowledge-based uses the increased memories of computers to build knowledge into AI systems. The AI built with this approach uses the knowledge they possess to solve problems.

Embodied intelligence

Embodied intelligence is used in the field of robotics and can be taken literally as intelligence in the body of the machine. It is the way that a machine can recognize and interact with its surrounding. For example, with movement, perception and visualization.

Computational intelligence

Computational intelligence uses artificial neural networks to solve a problem which cannot be solved with logical certainty, the system therefore makes an estimate as an approximate solution.

Statistical learning

With statistical learning, scientists try to solve more complex issues with AI by using a more complex mathematical language and more computing power. Statistical learning led to higher accuracy in results in fields like data mining and probability calculation. This step also turned AI into a more scientific field, with more calculations and experiments. Sometimes it is so complex that the results are not reproducible or understood by humans.

Development Methods used for AI

Different methods can be used to develop an AI. the decision of which method to use depends on the function of the AI and the approach to AI (Luger & Stubblefield, 2004)[59]. Some of the most used development methods are:

Search and optimization

Search means that the AI will search for an answer to a problem by looking at the different options that are available and choose the answer that most fits the desired answer. The number of options that the AI can choose from is too many for it to efficiently find the answer. Therefore, methods are used to optimize the field that the AI searches in. By continuously narrowing down the field, the AI is optimized to find answers faster.

Logic

Logic is a broad term in AI, it refers to all uses of methods that simulate human logic in different ways. For example: when approaching a turn at high speed, slow down before the turn. This can be done with methods like ‘fuzzy set theory’ which attributes a “degree of truth” between 0 and 1 to vague statements, telling the AI how “true” something is.

Probabilistic methods for uncertain reasoning

Probability methods are methods created for the AI to make a guess or statement with incomplete or uncertain information. This method is used for a wide range of functions. For example: filtering, making predictions and analyzing data.

Classifiers and statistical learning methods
Classifiers use pattern matching to determine a closest match. They are tuned with examples which are then learned as patterns. Each pattern belongs to a certain class. Every class is a different action or decision that needs to be made. All the observations together with the class labels are a data set. After multiple data sets have been created, a new observation can be classified according to these data sets.

Artificial neural networks
Neural networks are based on the way that neurons connect in the human brain. A neuron in a neural network accepts input from multiple other neurons surrounding it, which give a weighted vote on whether the neuron should activate or not. It is up to an algorithm to adjust the weight of these votes in order to learn. The weight is adjusted based on the training data. This makes it so that a network can also connect to a subnetwork. For example, connecting the word tree with the word leaves.

Deep feedforward neural networks
With deep feedforward neural networks, the signal only passes in one direction, ensuring that it is a straightforward process.

Deep recurrent neural networks
In deep recurrent neural networks, the signal can be scattered and varied. This allows the system to be capable of feedback and the creation of short-term memories of previous inputs.

NEAT (NeuroEvolution of Augmenting Topologies)
A NEAT-network does not have a fixed set of nodes and layers but creates them when needed (Stanley, 2004)[60]. The NEAT method was developed by Kenneth O. Stanley in August 2004.

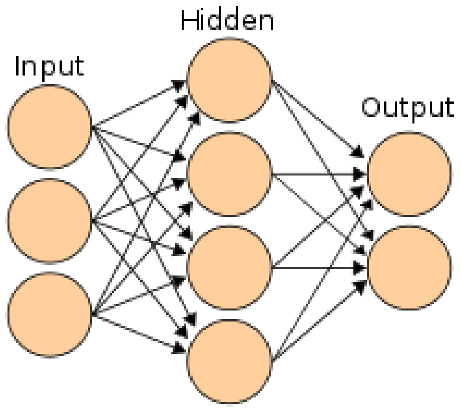


Image 2.8: A Simple Neural Network

Neural Networks

A neural network consists of multiple layers (see Image 2.8). Each of these layers has nodes which can be presented as circles. An output can be produced from each of these nodes when an input is given to them. The values of the nodes are set in such a way that at the end, the right results will be achieved. During the training, the AI tries to set the values of the nodes. After setting these values, the real output is compared with the output set by the AI and the values of these nodes are tweaked to make the right prediction next time. When this is done on a larger scale rather than a one time training, the neural network “picks up” the pattern and can predict data it has not seen before. The exact steps of what happens inside of some of these layers is not always visible, as these steps are done in a so-called hidden layer.

Functions in ML models

Loss

A loss function is one of the parameters required to compile a model. It is the difference between the output the AI gives and the expected output. It determines how much the value of the nodes needs to change to match the expected output. When the loss value is big, it means that the AI is not predicting accurately. In that case, the nodes have to change to give a better prediction next time.

Metrics

Metrics are also used when compiling a model and are similar to a loss function. The difference is that Metrics do not affect the training of the model. They are used to get a good understanding of how well the AI is performing during training, but they do not influence the training in any way. Metrics are useful to keep an eye on other values in addition to the outcome.

Activation

Activation functions are really important for neural networks to learn and make sense of complicated and non-linear functional mappings between input and response variables (Walia, 2017)[61]. Simply said, activation functions are functions which determine the output of a node. Each function has an input, an output and a variable which can be tweaked by the AI. If no activation function is applied, it will be set to the default linear activation function. Linear functions are easy to solve and limited in complexity. They also have less power to learn complex functional mappings from data. So if you would make a model without activation functions, it will not learn and perform optimally. A simple example of a linear activation function is `y = a*x . x` is the input the node receives and `a` is a variable which can be tweaked by the AI. `y` is the output of this node after receiving the input. An example: The input is 5 and the AI is expected to give an output of 10. The AI will change the value `a` to be 2, since the output will be correct given this variable. This process happens with all of the nodes within a single neural network.

Neural Network Layers

A neural network consists of layers. All of these layers have a purpose which influences the model. Keras, a high-level API for building and training deep learning models, has a few Core Layers which are the most common and easy to use (Keras, 2019)[62].

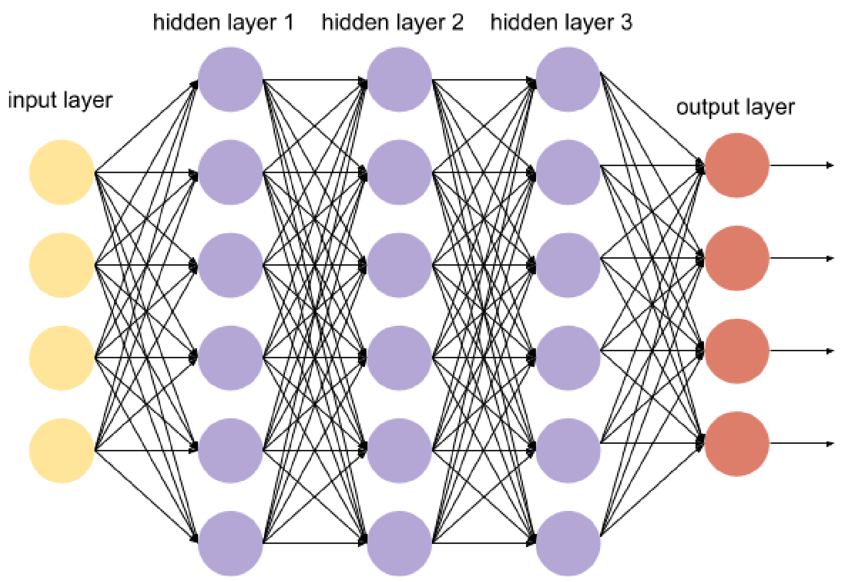


Image 2.9: Example of a Neural Network Layout

Image 2.9 shows how a neural network could look like if it consisted of 5 layers. At the beginning, there is an input layer. This input layer defines how many values there will be as input and how these will be given (these will be given in an array) during the training. There are multiple hidden layers behind the input layers. These have a predifined number of nodes (referenced as units within Keras) and these nodes will process the data given in the input layer. When a layer is hidden, it means that you cannot see what data is being used and how. The output layer is the last layer. This layer will define how the output is being processed. When one value is expected from the neural network, there will be one node in this last layer. Below are some of Keras' core layers explained.

Dense Layers

The first Keras core layer is the Dense layer. Each node in a layer receives an input from all of the nodes present in the previous layer. This means that they are densely connected with each other. Image 2.9 above is an example of a regular densely-connected Neural Network layer with hidden layers. Here is an example of how a Dense layer would be implemented:

```
keras.layers.Dense(units, activation=None, input_shape=(16,),
output_shape(8,))
```

`Units` refers to the number of nodes the layer should have. An activation function is a mathematical function that defines what the output of a node will be, given a specific input. An activation function can be defined in the `activation` argument to be used for this layer. The `input` and `output` arguments define in which format the data will be given to that node and in which format it should output its data.

Activation Layers

This layer applies an activation function (explained in Functions in ML models, activation) to an output. Activations can either be used through an `activation` layer, or through the `activation` argument, which is supported by all forward layers. Examples of both ways to use activation layers can be seen below. The first example adds a second layer, which is an activation layer and the second example adds an activation using the `activation` argument to a Dense layer.

```
# Example 1: Adding an activation to a dense layer
model.add(Dense(64))
model.add(Activation('tanh'))
```

Example 1 is equivalent to Example 2 below.

```
# Example 2
model.add(Dense(64, activation='tanh'))
```

Dropout Layers

Dropout randomly sets a fraction of the inputs to 0. This fraction is entered in the `rate` argument and is a number between 0 and 1. The `noise_shape` and `seed` arguments influence the process of randomization of the values, which will be set as a fraction of the inputs. Dropout layers are often used to prevent overfitting, which is explained below.

```
keras.layers.Dropout(rate, noise_shape=None, seed=None)
```

The Input Layer

An input layer determines in which format the data is given. It is a little similar to the input arguments used in Dense layers, but the structure is different. As shown below, it contains `Input()`, which is used to create an instance of a Keras tensor. A Keras tensor is a tensor object from the underlying backend (the Tensorflow software) that allows us to build a Keras model just by knowing the inputs and outputs of a model.

```
keras.engine.input_layer.Input()
```

So if `a`, `b` and `c` are Keras tensors, it becomes possible to do:

```
model = Model(input=[a, b], output=c)
```

Reshape Layers

Reshape is a layer used to reshape the output array. This is mostly used when you want to change the shape of input values, so it can be used with another layer. The reshape layer has two different notations, one when used as the first layer in a model and the second when used as an intermediate layer in a model. Both are shown in the example below.

```
# Example 1: Used as the first layer in a model
model = Sequential()
model.add(Reshape((3, 4), input_shape=(12,)))
```

Now the `model.output_shape` is set to `(None, 3, 4)` from `(12,)`, where `None` is the batch dimension.

```
# Example 2: Used as an intermediate layer in a model
model.add(Reshape((6, 2)))
```

Now the `model.output_shape` is set to `(None, 6, 2)`. The `input_shape` argument is not needed at the moment as it is used after a layer with an already specified shape and not as the first layer.

There are a lot of other types of layers (for example Permute, RepeatVector, Lambda, and Masking Layers), but those are often used in advanced models and we will not be using and analyzing them in this report.

Overfitting

Overfitting happens when a model learns the detail and the noise in the training data to the extent that it negatively impacts the performance of the model on new data. There are two definitions, the signal and the noise, which are often used with overfitting. The ‘signal’ can be seen as the true underlying pattern that you wish to learn from the data. The ‘noise’ on the other hand is the irrelevant information and randomness in a dataset. Overfitting means that the noise in the training data is partly picked up and seen as ‘signals’ by the model. This makes the accuracy of the training drop as it is not learning relevant data only, but random data as well (Keras, 2019)[63]. A visualization can be seen in image 2.10 on the right.

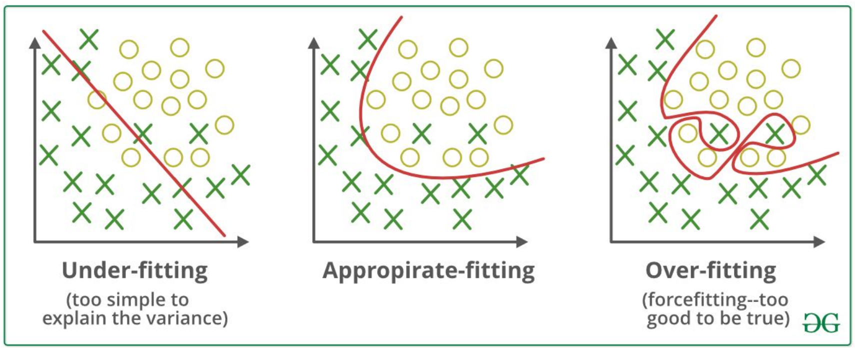

Under-fitting
(too simple to explain the variance)
Appropirate-fitting
Over-fitting
(forcefitting--too good to be true)

Image 2.10: The 3 types of fitting

Overfitting can be detected by the training of your model with the dataset. Normally you would split your dataset into a ‘training set’ and a ‘test set’. The training set contains the data used to train and tune your models and the test set is used after training to see if it can predict values. If the model performs much better on the training set (for example 80% accuracy) than the test set (for example 55% accuracy), then you can almost be sure it’s overfitting.

To avoid overfitting, you can do some of the following:

- Use cross-validation. Use the initial training data to generate multiple mini train-test batches, which will be used to fine-tune the model.
- Train with more data. This does not work every time, but training with more data can help algorithms detect the signal better as it learns from various examples.
- Remove features. Sometimes you can improve the accuracy by removing irrelevant input features, which could have a bad influence on the training.

Batches

Batches are groups of samples which are fed into the AI at the same time. They are used to make the training process go faster and more efficiently instead of feeding every sample to the AI separately. The batch sizes are always to the power of 2, which makes the training easier for the CPU. The batches must also fit within the memory of the CPU. A normal batch size is 32. This is because most modern CPUs have enough memory for this amount and the training will be executed smoothly most of the time.

Epochs

Epochs define how often the AI trains on the same dataset. 100 epochs means the AI will loop 100 times over the same given dataset. For every time the AI loops over the same dataset, changes are made to the nodes within the AI. However, these nodes can only change slightly with every loop. Therefore, a higher epoch count will result in larger adjustments to the nodes. But, when the amount of epochs is set too high, the AI will start to overfit. This has a negative effect on the training process.

Recurrent Neural Networks

Recurrent Neural Networks are neural networks with memory, which means it could remember sequential ups and downs of data in order to make more informed predictions (Yiu, 2019)[64]. Sequential data is data where the order matters. Examples of sequential data include:

- Stock prices or the average temperature for each day, which are both ordered by time.
- The words in an online article, where the order of the words convey context and meaning.

Thus, the volatility of the market can also be seen as sequential data.

You might ask what it means to add memory to a model and why you should want to do that? Memory is the ability to use information from relevant past experiences/training in order to assist during the decision making step. This means that the model will be dynamic. Long Short-Term Memory (LSTM) is one of the options used to add memory to a model. This will be explained below.

LSTM Networks

The problem with short-term memory is that if a sequence is long enough, the AI will have a hard time carrying information from earlier time steps to later ones. This means that it may leave out important information from the beginning of the training, when making predictions. On the other hand, the problem with long-term memory is that the gap between relevant information and the point where it is needed can sometimes become too big. This means that the recurrent neural network is unable to learn to connect the information.

To make the previous training information persistent and to avoid both of the problems mentioned above, Long Short-Term Memory networks are used (Olah, 2015)[65]. LSTMs are explicitly designed to avoid the long-term memory problem and remembering information for long periods of time is their default behavior and not something they struggle to learn. In short, it means that the AI is able to learn to keep only relevant information to make predictions, and forget non relevant data, instead of using that as well to predict.

You can implement LSTM layers with the code example below. However, adding only one LSTM layer to your model will probably not work very well without configuration.

```
model.add(LSTM(hidden_size, return_sequences=True))
```

To start developing machine learning models, we will need software and a backend 'engine'. This is further described in '[Subquestion 4: What languages and programs are used for AI?]'.

Mathematical theory behind AI

Math is not only used in the programs that are part of the AI, it is also used for the performance analysis of the AI. This way, it can be calculated if a used method or tool is successful or not (Russel, 2018)[66].

Asymptotic analysis

In the case that the effectiveness of an algorithm can be calculated with a simple sum and another with an asymptote, then it is calculated with asymptotic analysis if the algorithm has an asymptote. The calculation will then reach infinity and will therefore have a different effectiveness.

Vectors

Vectors are used to turn the results of an algorithm into a value that is visible or understood. For example, as a dot on a graph. Vectors can also be used to calculate two separate vectors, for example into x and y vectors.

Matrices

A matrix is a rectangular array of values which are arranged into rows and columns. One of the uses for matrices is matrix multiplication where one matrix is multiplied by another. This can be used to multiply certain specific sequences of numbers or data.

Probability distribution

Probability calculations and probability distributions are used to define certain probabilities for certain values. This makes it so that by using data, an algorithm can calculate the change to get a certain result from a certain value. In a probabilistic model, there is a sample space of exclusive possible outcomes together with a probability measure for each outcome.

Gradient descent (Dimitri, 1999)[67]

Gradient descent is an iterative optimization algorithm for finding the minimum of a function. This means that the algorithm starts with a guess and gets more and more accurate with each step. It does this by calculating if the gradient at that point is positive or negative. If it's negative, it's approaching the minimum. Within AI, this type of algorithm is used to calculate the most exact minimum value based on a function.

Conclusion: How does an AI work?

An AI works by using different learning and development methods. The methods are chosen based on the approach used for the AI. The choice of which learning and development method is dependant on the AI creator's intended function for the AI. Almost all modern AI uses neural networks. A neural network has multiple layers, each containing multiple nodes. Each node makes a calculation depending on an input. The mathematical function behind this calculation is called an activation. The actions that the AI performs are based on multiple mathematical concepts which can also be used to evaluate the AI's performance.

Subquestion 4: What do you need to create an AI?

There are a lot of different tools and libraries which people use to make machine learning models and to test them. As an answer for this subquestion we will describe some of the most used tools and programming languages which (in combination) can be used to build and train an AI.

Keras

Keras is a useful Python library used for creating neural networks. It is an API (application which can communicate with other applications), written in Python and capable of running on top of different machine learning frameworks. Keras is used to develop the model, which will then be integrated in TensorFlow. Programming in Keras can be seen as building with (pre-coded) lego bricks, as it is easy to build up and take apart different layers in your model. Without Keras it would be a lot harder, because we would need to build the model from scratch, which would be a lot harder to debug.

TensorFlow

TensorFlow is a collection of tools, libraries and community-shared resources that helps to train Machine Learning models. It is used to power the learning process of the earlier created Machine Learning models with Keras. We will be using Tensorflow Keras, which is Google's implementation of the Keras Python library. TensorFlow Keras is easier to use and more user-friendly compared to TensorFlow alone.

In short, Keras is an abstraction layer that builds up an underlying graphic model. TensorFlow is the engine that does all the heavy lifting and "runs" the model created in Keras.

Python

Python is a programming language often used for creating an AI. It is easy to learn and the developers of libraries (these are collections of functions and methods in code) often leave a documentation with their code, which can be used while troubleshooting during the programming phase. Python is also an open-source language and therefore it is possible to use the libraries for your own projects and contribute your code and updates to it if you want to. Python is widely known for this which is why Python is used in a lot of different programming situations. Below are some of the libraries, we used, explained.

Pandas library

Pandas is a common Python library. Pandas is useful for writing and reading CSV-files and analysing data. In this project, it will only be used to read the data from the file.

Pytorch library

Pytorch is a free and open source Python library (PyTorch, 2019)[68]. It is a machine learning library based on another Python library named Torch. It got the name Pytorch because it has optimization algorithms to build neural networks.

Numpy library

When it comes to arrays within Python, Numpy is the way to go. Numpy can “be used as an efficient multi-dimensional container of generic data” (Numpy, 2019)[69]. When working with large amounts of data, Numpy can be used to store this data in multi-dimensional arrays. Multi-dimensional arrays are used when setting up a dataset for the AI. With a normal dataset, 2 dimensions are mostly enough because there are values and time stamps. Datasets for an AI need this data in batches as well and are therefore mostly in 3 dimensions.

Matplotlib library

Matplotlib is a Python 2D plotting library which makes it possible to generate a lot of different charts and plots using just a few lines of code (Matplotlib, 2019)[70]. You can generate plots, histograms, power spectra, bar charts, scatterplots and even more. This library will be very useful to visualize a lot of points after the training of the AI as it will make the results understandable and useful while documenting the code.

Time library

There is a standard library included in Python called Time (Python, 2019)[71]. This library can be used for various time related functions. When working on building an AI, is it useful to know how much time it takes to train the AI. With this information, it is possible to estimate how much tweaking can be done and how often the AI can be reshaped.

Version control with Git

Git is a free and open source control system designed to handle everything from small to very large projects with speed and efficiency (Git, 2019)[72]. With Git, it is possible to work together on code and share this code easily without any inconvenience. Git also allows multiple developers to work on the same piece of code without the possibility of overwriting code from other developers while saving. This is a really important feature, as it allows a large group of people to collaborate effectively on code development.

GitHub

GitHub is a development platform where code can be stored and reviewed. Adaptations can be viewed and it can be determined which ones should be kept.

Google Collab

Collab is a great online code development platform by Google Research. It allows anybody to write and execute Python code through the browser and is especially well suited to Machine Learning, data analysis. At its core, Colab is a hosted Jupyter notebook service that requires no setup, while providing free access to computing resources including GPUs. There is also no maintenance needed when making use of Collab, so it can be a great environment if you don’t have any hardware and need to work on Machine Learning projects.

(Optional) Hardware

Of course you will need some hardware to run the training process of the AI, but we have made this part optional, because there are a lot of different ways to use online resources. You could use Google Collab for example, which lets you use resources from Google, such as RAM and disk space, for the training period. When you are done, the resources get flushed, but you will have the results of the training.

It is also possible to develop your AI locally if you are planning on setting up your own hardware to use for the training. As stated here (Stackoverflow, 2017)[73], there are no static minimum requirements to run Tensorflow on your own hardware. However, when you want to do larger projects you will need to take into consideration how much data you are trying to put into memory (RAM) as it will reduce your performance if you are flooding it with data. To avoid this problem, you could do the following:

- Store fewer training examples in memory at one time.
- Use smaller batch sizes.

Conclusion: What do you need to create an AI?

The first step of the development process is to make a list of all of your requirements. You should start by selecting your hardware by choosing to use Google's resources or your own. In case you are using Google's resources, the setup process is already done for you, but if you choose to use your own hardware, you will need to create a training environment. After preparing your environment, you can start with importing the required libraries. These are: Tensorflow, Tensorflow Keras, Pandas, Pytorch, Numpy, Matplotlib and Time. A detailed guide on how to prepare your environment and how to import and use the libraries can be found under Guide for Building an AI, Selecting your coding environment and software.

Summary: The Basics of AI

An AI is a program or machine that exhibits human-like aspects, like decision-making and communication. It achieves this by using complex algorithms. AI can be categorized in two ways: based on its level of intelligence, or based on the applied learning approach. When categorizing by level of intelligence, there are three types. The first type is the weakest, ANI, which has a very narrow expertise and can only perform a single (type of) task. At the moment, all existing AI is ANI. The second type, AGI, which has yet to be created, shows a general intelligence. This means that this type of AI is able to mimic all types of human intelligence and behaviour. The final type, ASI, surpases human intelligence and possesses consciousness. When categorizing AI based on the applied learning approach, AI is categorized in three methods. Supervised Learning is applied when working with a clear output, Unsupervised Learning is applied when patterns have to be found, and Reinforcement Learning is used when the input is more complex than numbers in a database. Many AI are reliant on Neural Networks which are made up of layers. Each layer consists of many nodes which have their own activation function. When training these Neural Networks, batches are used which ease the CPU workload. To fully train a Neural Network, the data must be fed through the Neural Network multiple times. Circling over the Neural Network is done by using epochs. For the development of an AI, the coding environment is crucial. This environment consists of the hardware and software which are applied. As an alternative to using personal hardware, Google offers online access to all the resources needed to build an AI. After preparing your environment, you can start importing the required libraries. Many libraries are necessary for building an AI, and some help with more complicated processes.

Now that it has been made clear what an AI is, what its divisions are, how it works and what you would need to create an AI, it's time to look at what AI can actually do. The capabilities of AI are virtually endless. In the last few years, AI has been applied to many different disciplines. To gain a good understanding of the applications of modern day AI, the next chapter will discuss how AI connects to other subjects and how AI is currently used.

How AI Connects To Other Subjects

Philosophy

With the way that AI thinks resembling how humans think, we are constantly making machines more and more like humans. But what makes us human and how do you simulate this through AI technology? Is it even possible to simulate human thinking? By answering these questions we not only understand better how to develop AI, but we also learn more about humanity through answering these philosophical questions. Philosophy regarding AI also shares concepts and discussion points with philosophy in general. For example: intelligence, action, consciousness, epistemology and free will (McCarthy, 2006)[74].

General Idea

By using the study of philosophy to answer difficult questions regarding the field of AI, we will be able to understand and develop AI easier. Because AI “thinking” aims to simulate human thinking, certain questions regarding humanity are also applicable to AI. For example: the morality in decision-making, the “correct way” to solve problems and how we would define human traits like emotion, thinking and experiencing life.

There are, however, a number of different propositions in the philosophy of AI regarding the intelligence of machines:

- Turing's "polite convention*": If a machine behaves as intelligently as a human being, then it is as intelligent as a human being.* (Russel, 2003)[75]

If a machine makes decisions and acts just as intelligently as a human being, then its intelligence is equal to a human’s intelligence.

- The Dartmouth proposal*: "Every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it."* (McCarthy, 1955)[76]

If we are capable of explaining exactly how we think, learn and display other aspects of intelligence, then it can also be described in such a way that machines can simulate it.

- Newell and Simon's physical symbol system hypothesis: *"A physical symbol system has the necessary and sufficient means of general intelligent action."* (Newell & Simon, 1976)[77]

A physical symbol system means that a physical pattern (symbols) combined with structures (expressions) and manipulated by using specific processes will result in new expressions. This is a description of how we think and process information. It is used in the symbolic approach for AI. Human thinking can be described as symbol processing and machines can do this as well, therefore AI intelligence can be similar to equal to human intelligence.

- Searle's strong AI hypothesis: *"The appropriately programmed computer with the right inputs and outputs would thereby have a mind in exactly the same sense human beings have minds."* (Searle, 1999)[78]

Searle says that we don’t have to simulate the brain to stimulate the mind. To give AI an intelligence similar to us we don’t need to look at how we think but at how we can make an AI think differently and get the same results as our way of thinking. Implementation is unimportant, so the only thing that matters is the results that the system gives. A computer cannot be shown to have a "mind".

- Hobbes' mechanism: *"For 'reason' ... is nothing but 'reckoning,' that is adding and subtracting, of the consequences of general names agreed upon for the 'marking' and 'signifying' of our thoughts..."* (Hobbes, 1651)[79]

Reasoning is nothing more than reckoning: our intelligence is nothing more than a form of calculation. Therefore, if thinking can be boiled down to calculations, a machine should be able to do those same calculations as well.

The question of how machines think and if it's similar to human thinking is the main question in the philosophy of AI. Other important questions are:

- Can a machine have emotions?
- Can a machine be self-aware?
- Can a machine be hostile or turn on its creator?

Questions like these are not only important for us to learn how we should use AI, but in the case of questions like "Can a machine display general intelligence" it is also relevant to the development of AI. These questions help to determine how the AI should be developed in order to reach the intended result of showing general intelligence.

First Connection

As described earlier, one of the first ideas regarding the intelligence of machines came from Thomas Hobbes. Thomas Hobbes was an English philosopher who was one of the founders of political philosophy. In addition to philosophy, he also studied in the fields of geometry, physics and ethics. This explains his interest in looking at thinking as a logical process. His opinion on intelligence was written in his book called *Leviathan or The Matter, Forme and Power of a Common-Wealth Ecclesiasticall and Civil* in 1651 (Hobbes, 1982)[80]. The book tackled a broad number of subjects, for example: religion, taxation, governments and politics. He also described his idea of mechanism, or mechanist philosophy. He stated that everything can be explained with logical thinking about what actually exists in the here and now. This way of thinking can be used as a basis for the way an AI thinks, namely using logical thinking about what actually exists. Furthermore, the view that all things can be tackled using calculations, correlates with Hobbes' theory that human intelligence, at its core, is also just calculations.

Example

Multiple researchers including Professor John McCarthy have tackled difficult questions regarding AI from the perspective of philosophy. These questions are regarding topics like self-consciousness, how we think and other existential questions. By understanding how these concepts work or should work, AI-researchers are able to make an appropriate plan on how to program an AI . Some philosophers also argue that the role of philosophy in AI is underrated (Bringsjord, Selmer and Govindarajulu, Naveen Sundar, 2019)[81]. For example, different people can have a different idea of how an AI is supposed to act. Do we want an AI to make choices and take actions like a human being, or do we want AI to do this rationally? Rational thinking would lead to a very “by the numbers” thinking type AI, while an AI that acts more human could have some errors or unclear reasoning. The ambivalence surrounding questions like these hinders the development of AI.

Ethics

AI ethics is the study of ethical situations concerning AI, for example, liability for the actions taken by the AI, weaponization of AI, and unintended consequences of AI usage. AI ethics is a combination of roboethics and machine ethics, because AI technically falls under both categories.

General Idea

AI-technology is one of the most important technologies being developed at the moment. This development needs to happen in a controlled environment however, because AI is a very powerful tool. The nature of AI and the possibilities of what can be done with it lead to a number of interesting ethical questions regarding society, power and existence. AI-researchers therefore also want politics to get involved with AI; politicians and lawmakers should make clear rules on the use and development of AI. Different countries already have different ideas regarding this, which have led to multiple new ethical questions.

Subjects in the field of Ethics of AI

Some of the most discussed ethical topics regarding AI are:

Biases

AI can learn certain biases depending on the data given and the method used to train it. In a way, the biases of the person or machine training the AI are picked up by the AI. This has caused situations such as certain ethnicities not being recognized by a facial recognition AI. But when AI is used on a larger scale, even more issues can come up. Biases will inevitably prevail when teaching an AI what is right or wrong. This means that if an AI that is meant to check for inappropriate content gets wrongly taught what’s right and wrong, this could lead to incorrect decisions. Standards of right or wrong are of course subjective, every culture has their own ideas and beliefs. This makes it difficult to say what type of activity is right or wrong.

Liability

An AI makes its own decisions, but is developed by humans. Therefore, when an AI makes a decision that is punishable by law, it needs to be determined who is liable for the action. This is an interesting subject for politicians and law-enforcement. If an AI were to be used to commit a crime, then who is to blame for the crime? The general consensus seems to be that the one who trained and activated (used) the AI is guilty for its actions. This is because it is understood that AI is programmed by humans. But what happens when an AI programs another AI?

Weaponization of AI

Because of its efficiency in simulating human thinking skills, AI has been an interesting technology for military use. The use of AI as a weapon is quite controversial. If the weaponization of AI is perfected, it could cause more casualties and damage than the weapons used in regular warfare. There is also the risk of AI weapons becoming autonomous weapons, thinking and acting for themselves. If AI weapons are not kept under control, there could be severe consequences.

Unintended Consequences

AI can act autonomously, this means that the decisions it makes can be unpredictable if not developed right. These actions can cause consequences which were originally not intended.

Unintended consequences can also arise on a larger scale. For example, if AI is used in combination with robotics to automate certain factory processes, some jobs will be lost. However, by automating the process it becomes more profitable. This, of course, is a motivation for more atomization. Eventually, a lot of jobs will be lost, the jobless workers will have a hard time finding a new job and the poverty in the country will increase. This is an unintended consequence of automating a process using AI.

Control and Robot Rights

The subject of control and robot rights is a bit more existential in nature. The question is if robots deserve rights and if we have the right to control and use robots. This is because we have to wonder if at some point a robot can be seen as a sentient being, somewhat equal to us, and if having control over a being like that is not similar to having a slave. Another question is if robots and therefore sentient AI even deserves rights, as it is not a natural organism but a completely different form of life.

First Connection

There doesn't seem to be an exact point where the first connection between ethics and AI was made. Discussions about ethical subjects have been going on since the concept of using AI as a tool was taken seriously. Around the time that major developments in AI started, discussions regarding the ethical use of it also began, for example, with the discoveries of Alan Turing using the Turing Test in 1950. Scientists and philosophers have always first thought about how AI should work and what it should be used for and then asked the question if it is ethical or not.

Example

China has been using AI for a while now to monitor its citizens. The intensity of this differs per region, but the government also keeps an eye on their citizens in general. The most controversial use of AI by the government is facial recognition in public places. This has been used in combination with the monitoring system to find citizens who the government is after. Once located, these citizens will then receive sanctions, or get taken to "re-education camps". The way that China uses technologies like AI is highly controversial (Mozur, 2018)[82], and acts as an example to other countries of how *not* to use it.

On the other side regarding ethics, there are companies/organizations like OpenAI. OpenAI is a company whose mission it is to ensure that AI benefits everyone. The benefits should be for everyone, not just the elite. The new developments should also be safe. Not just in a broad sense of not harming anyone, but also to not infringe on human rights (OpenAI, 2018).[83]

Technology

Over the years, AI has been implemented in more and more existing technology. Because AI can perform human-like tasks, it is used to enhance existing processes and technology. AI can be found on your phone in the form of assistants like Siri or on a larger scale in the contentID system of Google.

AI performs the tasks that we do not want to do, because they are tedious or repetitive.

General Idea

AI is mostly used in the same way that other technologies have been used, to do things that we cannot or to simplify a certain process. Because AI is capable of performing human-like tasks, it is used in more complex processes. These processes often require a lot of decision-making or data analysis. The tasks are too repetitive and/or hard for humans to do, but an AI with its focus and intelligence is capable of doing it efficiently. AI is used, or can be used, in almost all sectors: healthcare, transport, agriculture, etc.

First connection[84]

The first uses of AI were machines that were able to play checkers or chess. The first of these was built by Christopher Strachey in 1951. The first AI used for a productive task was the Logic Theorist, developed in 1956 by Allen Newell, J.C. Shaw and Herbert A. Simon. This program was able to perform automated reasoning and would end up proving 38 of the first 52 theorems from the Principia Mathematica, a book written by Alfred North Whitehead and Bertrand Russel in 1910. It is a three-volume work with the basics of mathematics. The AI also eventually found new proof for more theorems.

The creation of the Logical Theorist was proof that machines are able to perform tasks considered intelligent and creative. It convinced people that AI was not only possible, but also an effective tool in problem-solving.

Logical Theorist also introduced three new concepts to AI:

- Reasoning as search: Logical Theorist used a search tree to come to a certain conclusion. The root was the hypothesis and each branch was a deduction based on logic. If the AI was able to make a path from the root to the end-result, the theory was proven.
- Heuristics: In order to make sure that the AI did not have to go through exponentially huge amounts of branches, they used a technique they called heuristics. Heuristics would “trim” the branches which were decided to be unlikely to lead to a solution.
- List processing: To implement the AI on a computer, the researchers developed IPL, a programming language that was capable of symbolic list processing.

Examples

There are very many examples of AI being implemented in current technology. Some of the most important ones are:

Medical

In the medical world, AI has been used to assist or in some cases even replace doctors or nurses. In India, over 72 million diabetics need to be screened for diabetic retinopathy, which can cause blindness. However, the shortage of over 100,000 eye doctors means that only 6 million patients can be screened. Google and its sister organization Verily have created an AI which can almost instantly detect diabetic retinopathy in images of retinas. This brings two main advantages. First of all, scanning becomes faster and therefore more accessible for people who cannot wait months for a screening. Secondly, the shortage of eye doctors is greatly reduced, as they can focus on treating patients who have been marked positive during the screening, instead of having to screen all of the patents themselves (Miliard, 2019)[85].

Transport

The use of computers is not new in the transport industry. Cruise control, for example, has been around since 1948 when it was invented by Ralph Teetor. Now, with the use of AI, even the driver can be replaced. Self-driving cars use GPS and AI to automatically bring a passenger from point A to B. The AI has multiple functions such as face recognition, route-planning, positioning, and driving. All of these functions also have sub-tasks, driving, for example, includes regulating speed, turning, paying attention to other drivers, etc. Tusimple has made an AI which can control a truck without any help from the driver. The driver should still be present in case the AI does not take the right actions but he/she does not have to interact with the driving mechanics. This could prove to be far less tiring for truck drivers. It can be especially useful for trucking industry since truck drivers can only drive for 8 hours a day and need to take regular breaks. Therefore, the potential for self-driving technology to transform this industry is great. The input for the AI that controls vehicles is mostly cameras to see the surrounding traffic and roads but can also contain light detection and ranging sensors. The automated trucks have a 360° view so it can detect every car in the vicinity (tusimple, 2020)[86].

Assistance

AI-assistants can perform a wide array of functions. They can plan events, answer questions, help find a route to where you need to be, remind you of appointments, etc. AI-assistants can be only a program, for example Siri or Google Assistant, but they can also come in the form of a robot for example Pepper and Asimo.

Agriculture

The university of Wageningen is developing an AI to grow peppers without any human interaction. The AI has cameras to make 3D models of the plants and conclude how well each plant is growing. Based on these models, the AI makes decisions regarding the watering, lighting and temperature of the plants.

Disaster prediction

There are over 2000 satellites surrounding the earth and most of them constantly collect images of the earth. This enormous amount of data can be used by AI as there is a sufficient amount of data to learn from and base predictions on. This is exactly what Descartes Labs does[87]. They use satellite photos to predict crop health and land fertility among other things. With this information, agricultural planning can be arranged in advance, and more accurately, resulting in improved food supply and less hunger.

It is predicted that a huge earthquake will take place in California within the next 30 years. When exactly is not sure. To ensure that as few people as possible get hurt when it strikes it is crucial to make a more accurate prediction of when it will occur. Sensors have been placed all over California to detect sounds that can indicate an earthquake. These signals have, in addition to the useful data, also distortion signals, like a train coming by. These signals are not useful when predicting an earthquake. This is where the AI comes into the picture. The AI filters out the useless signals. It compares the incoming sounds with a large dataset of known sounds and filters these out.

Literature (Fiction)

Writers of Science Fiction quickly took notice of the ideas of artificial intelligence. Machines with human-like intelligence weren't exactly new, but speculating and imagining how these machines would work and interact with humans have also shaped the public's view and ideas of AI. Some works of fiction even helped visualize certain ideas within AI.

General Idea

Artificial intelligence is an interesting but also controversial topic. Just like how philosophers and psychologists try to figure out and explore the subject from their viewpoint, so do writers. Fiction is used to explore the different effects that AI development and other technological developments have on our lives. These stories are almost always under the genre of Science Fiction, fictional stories that explore and speculate on where technological advancements will bring human society. This fictional future is often one of two things, it is either a utopia or a dystopia. Which means that life either becomes near perfect or it will become a disaster. This reflects the ideas that the author has on these technological advancements.

This exploration of ideas not only gives us a visualization of what such a future might look like. But it is also able to make us question the direction that technology is heading and whether it is actually beneficial to humanity. Some Science Fiction novels are therefore also considered philosophical fiction.

First Connection

One of the first works of fiction that has a machine with human-like intelligence is the novel 'Erewhon' (see Image 3.1), written by Samuel Butler in 1872[88]. In the beginning, the novel seems to be about a utopian society, but it soon becomes apparent that it's a satirical take on the Victorian society of the time. Butler was one of the first writers to write about the possibility that machines might develop consciousness by a process similar to selective evolution as described by Charles Darwin.

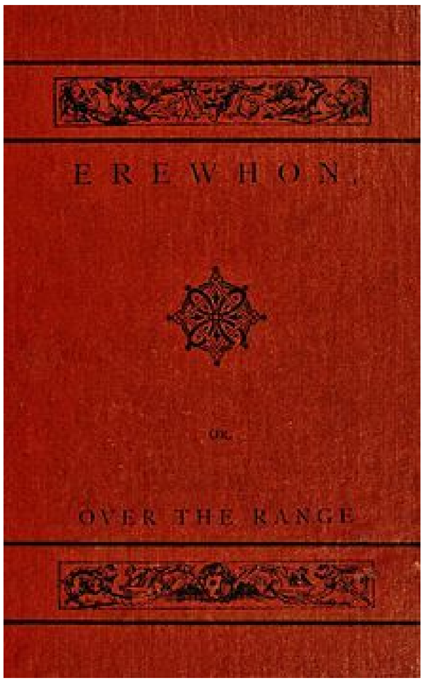


Image 3.1: Erewhon by Samuel Butler

Butler had already written about machines being, or being able to become, more human than we think. He had already published a number of articles about this idea, including the article 'Darwin among the Machines' on June 13, 1863 (Butler, 2004)[89]. This article not only became quite influential for fiction about machines gaining consciousness or human-like attributes, but also for the idea of AI in general. It described how machines are a kind of "mechanical life" which just like all other forms of life is constantly evolving, eventually surpassing humans and becoming the dominant species. Butler says in the article that if we don't stop this evolution, humanity will be in danger of being replaced. He urges that humanity should go on a war to the death against machines to prevent this.

This idea of “Darwin among the Machines” was quite influential. Not only for fiction about intelligent machines, but also for discussions and ideas around artificial life and intelligence. For example, technological historian, George Dyson built upon Butler’s ideas when he wrote the book, ‘Darwin Among the Machines: The Evolution of Global intelligence’ in 1998[90]. Dyson combines Butler’s idea with the ideas of artificial life and intelligence as described by Alan Turing and suggests that the internet is a living sentient being.

Examples

Do Androids Dream of Electric sheep? - Philip K. Dick (1968):

Synopsis: In a world where most animals have become extinct or endangered and most natural life is dead, a bounty hunter needs to kill six escaped androids while another man protects and helps these fugitive androids.

Themes: Humanity in machines, can a machine feel empathy?, what makes a human being ‘human’?

The Matrix - The Wachowskis (1999):

Synopsis: The main character realizes that his entire life is a simulation and joins a group of freedom fighters to get out of the simulation, master it and defeat the AI that controls the system and this simulation.

Themes: Humanity in conflict with machines, reality inside a simulation, evolution of machines, sentient machines.

Summary: How AI Connects To Other Subjects

AI is connected to many different disciplines. Philosophy, ethics, technology and literature are only a few examples of fields where AI is relevant . Philosophy connects to AI as both searching for a deeper understanding of how humans think. Ethics have always been a part of human thinking. Everyone has a different opinion about ethics, yet an AI must have defined rules for how to act at specific crossroads. Every day AI pushes new technologies to its limits, replacing or enhancing functions previously performed by humans. Finally, speculating about the future of AI will always be a popular subject in literature, ranging from disasters to utopias. Now that an overview of AI has been presented, and many applications of AI have been explored, it is time to look at what is involved in making an AI. To fully understand the process of making an AI, it is best to learn how to make one yourself.

Guide for building an AI

This section of the report is to guide the reader who wishes to build his/her own AI (from here on referred to as: you). This guide is divided into five phases: Orientation, Selection, Examination, Creation and Training & Refining. To complete a phase, all steps within this phase must be completed (except the indicated optional steps). It is not recommended to continue to the next phase without fully completing and understanding the previous phase, as it could leave your AI with gaps. With each step, an example will show how the step could be executed. The examples (marked by the header + “Example”) are drawn from a specific project. The example project is the creation of an AI to predict the server load of a company dependent on the previously measured traffic (also referred to as network load) and volatility. Volatility is a rate at which the price of an asset increases or decreases for a given set of returns.

1. Orientation

This phase is focussed on getting a good general idea of what your goal is, what your limitations are and how your goal will be achieved. This is an important first step for every project.

1.1 Deciding what the AI should do

Having a good general idea of what you would like the AI to do is an important first step. After you visualize this, you have a goal to work towards which will keep you on the right path. It is important to be very concrete and detailed in the functional description of your AI, as this will be the foundation of the project and it’s goal. If the function is not defined carefully, the goal could become hard to complete.

1.1 Example

The example project is described as follows: Create an AI to predict the server load of a stock trading company based on the volatility of the stock market. Here, the function of the AI is to find a correlation between the volatility and the server load. It is important to note that the server load and volatility will be used to train the AI and that after the training is completed, it should predict the server load given a certain amount of time stamps related to both volatility and traffic. The training processes will be dependent on the provided data in the dataset; the time and date will be important factors in the training process as well.

1.2 Determining your limitations

During this step, you determine and inventorize all limitations you will face in your project. This is an important step as it provides information about the amount of progress you could make during the project. One of the biggest limitations on most projects is time. This is because there is often not enough time to learn absolutely everything about building an AI (even if you already have some experience in this field).

Another significant limitation could be the availability and amount of data which could be used for the training. This limitation has a bigger impact on smaller projects and companies compared to projects conducted by large companies like Google or Facebook. This could be because bigger companies often have a lot of customers or users, who generate data when using the company’s services. Smaller companies lack the amount of customer activity to generate ample amounts of data. A small dataset could be a pivotal limitation because the model may therefore overlook some patterns, which would have a conclusive influence on the predictions.

2. Selection

2.1 Selecting the learning approach of your AI

The data structure, required data, language and environment are all dependent on what type of AI you are planning to develop and what the AI's function is. For detailed information on the specific qualities of each type of AI and what forms of training there are, please look back to "Subquestion 2: What are the divisions of AI?" where different AI learning methods are discussed.

It is important to consider how you want to approach learning. Keep in mind whether or not you're going to use Machine Learning, or the more specific Deep Learning. It is recommended to search online for AI projects similar to what you want to make. This will not only give you a clearer view of what is achievable as the final result, but it will also inform you of how this result could be reached. If you have no idea what type of Machine Learning you should apply, or if you are uncertain and can't find a definitive answer online, it is best to use standard Machine Learning and train the AI using Supervised Learning.

2.1 Example

The function of the AI that we are developing is to predict a future value. The only way to accurately predict something which will happen in the future is to analyze a lot of situations in the past. The more examples from the past, the more reliable the prediction can be. Therefore, it can be expected that a large amount of data from your dataset will be used to train the AI. After contacting our client, he confirmed that two large datasets would be supplied to us for the training.

The first dataset will contain the traffic of the company's servers corresponding with a date and time. The second dataset will contain the volatility, which is also corresponding with a date and time. Each dataset contains data points with measurements separated by one minute. Based on this knowledge, Deep Learning, the form of Machine Learning which is better suited for large amounts of data, seemed the most appropriate form of learning to apply to this project. Furthermore, Supervised Learning seems like the best way to train in this case. During Supervised Learning, the AI will receive training data and will be asked to calculate the following value. After this, the predicted value will be compared to the measurement of the same time and it will continue to predict the next point and so forth.

2.2 Selecting the required Python libraries

After setting up a detailed plan of the AI that you want to make, it is important to select the libraries you will use to develop the AI. Two of the often-used libraries for configuring a neural network are Pytorch and Keras. Keras makes use of other libraries as back-end. For more information about the Python libraries used in this project, please go back to Subquestion 4: What do you need to create an AI?.

3. Preparation and examination

During this step you will prepare your working environment and your datasets for training and the setup process of the AI.

3.1 Choosing the location of your environment

You will need to choose a location for your project files and your ‘engine’. If you want your files and project to be accessible to more people, you could upload your files to Google Drive and use Google Collab for code editing and running the AI. This way you make use of the resources provided by Google and you do not need to pay for hardware. However, if you want to start a bigger project, you will need more disk space for the datasets. A solution could be buying more disk space in Google Drive or setting up your own training environment on your own hardware.

Big companies often use their own hardware for training, as it has some other advantages over Google’s environment. One of the biggest advantages is privacy, as some companies are training with datasets containing confidential information. This information could be harmful to the company (or its clients) if it is leaked and should therefore stay inside company walls. Another advantage is the flexibility as you can build your own environment and thereby choose what software and hardware you will be using. For example, if you have a lot of big datasets, you could invest in more storage. This way you could have an environment which is set up to suit your needs better than the Google Collab environment.

Using your own environment also has some disadvantages. For example, you will need to invest and buy your own hardware, which will need to be capable of running Machine Learning software. Maintaining the servers is also a disadvantage, as it takes time to set up and keep the environment secure. You also wouldn’t want your environment to break down as it could cause a significant amount of damage to the progress of your project. To avoid some of these disadvantages, you could rent server space from another company where you will have access to resources online. This way you make use of the hardware of another company and keep the flexibility of setting up your own personal environment.

The setup process of both online and offline environments are explained below. For our project, we first tried to make it work locally on our own hardware, but then started to make use of the Google Collab environment. We made that choice because it was a lot harder to maintain and keep the local environment running without errors than we thought.

If you choose to set up your own environment using Jupyter, read further about ‘3.1.1 Setting up your local environment with Jupyter’ and skip the section about Google Colab. If you think that a Google Drive & Colab environment is a better fit for you, skip 3.1.1 and read further at ‘3.1.2 Setting up your online environment with Google Drive and Collab’.

3.1.1 Setting up your local environment with Jupyter (Optional)

As mentioned above, you will need to have your own hardware and software ready to build a local environment. Below are some of the minimum requirements. You will need:

- **Hardware** (often a **Server** or **PC**). This server will run a Tensorflow Docker container (more about this below), which will do the processing and training of the AI. For this project, an Intel NUC barebone pc will be used, but as you can read here[91], there are no minimum hardware requirements for the Tensorflow container. We were using the following hardware specifications: Intel Core i7, 16GB RAM, SSD storage, but you could use lower specifications as well, although low specifications are not recommended, as the performance will be worse.
- A **PC/Laptop** for the programming work. Using a pc/laptop, you will be able to log into the Jupyter environment.
- An **Internet connection** to download a clean copy of an operating system, the Docker software, different Python libraries, the Tensorflow container and your datasets.
- (Optional) If you want to be able to work from anywhere in the world and leave your server at home, you will need **the ability to port forward**. We won't cover that, but you can Google the terms 'docker container port forward' to read more about container networking.

It's also possible to install the Tensorflow docker container on your laptop, so you can take the neural network with you. A disadvantage of this method is that the training speed of your AI will be limited by the hardware of your laptop.

For our project environment, we have decided to use Docker because it allows the Tensorflow virtual environment to be isolated from other applications and only a part of it to be accessed. This means that you will have a ‘light’ working environment, so you don't lose performance on unnecessary resources on your server.

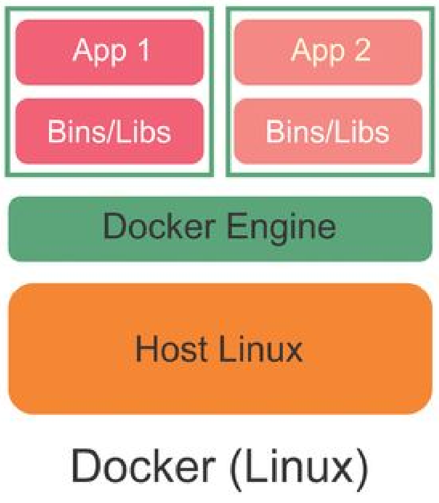


Image 4.1: Docker container structure

An example of how Docker works can be seen in the picture on the right (Image 4.1). The first step is to install a host operating system on an empty server. Linux is often chosen as it is light and has a lot of support. After that, install the Docker engine in the empty Linux operating system. The advantage of using Docker is that you can create a separate ‘box’ where your AI is separated from other applications and only has the tools it needs installed. This way you don’t have unnecessary processes and applications installed and running.

It is also easy to update/remove/reinstall a container without affecting your files as you are not supposed to store information inside, but outside a container. This way, if something breaks or stops working, you just delete the container, install a fresh one and continue working.

We have chosen to use the official Tensorflow Docker container called `tensorflow/tensorflow:latest-py3-jupyter` because it has all of the software required by Tensorflow and Keras to work. The Tensorflow container also uses Jupyter which is a GUI

(Graphical User Interface), which can be accessed from the browser. It is often used by developers to develop open-source software and services. It also has support for dozens of programming languages, which means that you can use the environment while developing other applications as well. The most useful features of Jupyter are:

- The ability to create Python notebooks, where snippets of code can be run immediately and separately from each other.
- You have terminal access to the Docker container and it lets you install new libraries and run Python files.
- Easy file management with folders as well as upload, download, move and rename functions.

To install Docker and the container on a Linux server, type the following in the command prompt:

- `sudo apt-get update && sudo apt-get upgrade`. This will install the latest updates on your system.
- `sudo apt-get install docker.io` will install the latest version of Docker.
- `sudo systemctl start docker` will start up the Docker service in the background.
- (optional) `sudo systemctl enable docker` enables Docker to run at the startup of your computer/server.

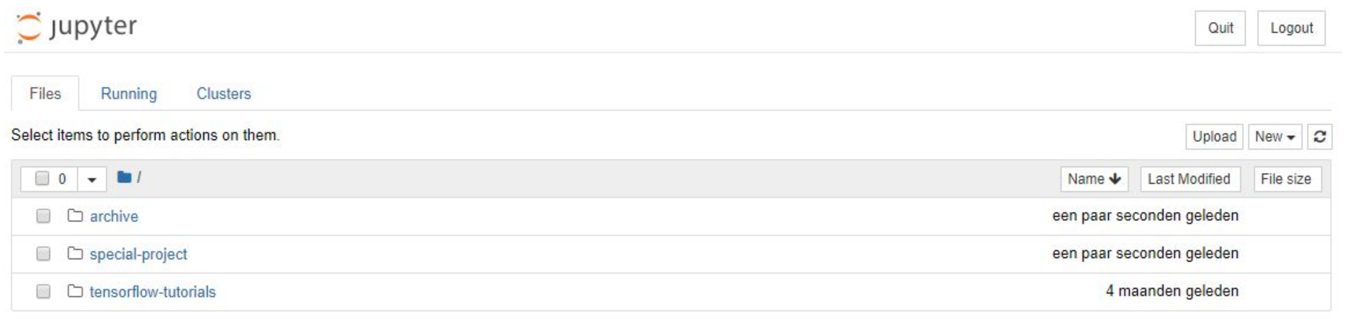
After installing Docker, you can issue the `docker --version` command in the command prompt to verify the installed Docker version number. Now we are going to download, install and set up the Tensorflow container which runs in the Docker application. To do this, open a terminal (as root user) and use the command:

```
docker run -it --name tensorflowAI
-v $(realpath ~/tensorflow/archive):/tf/archive
-v $(realpath ~/tensorflow/special-project):/tf/special-project
-p 2000:8888 tensorflow/tensorflow:latest-py3-jupyter
```

The `--name` option in the first line lets you give a name to the container, the `-v` (volume) option lets you add a location to save your files, as you should not keep data inside the Docker container because you could lose it if the container crashes. The `-p` option lets you specify a port from which Tensorflow will be available outside the container (in this example it's available outside on port 2000. Port 8888 is the default port used inside the container, which is mapped to port 2000). `tensorflow/tensorflow:latest-py3-jupyter` is the version container, which will be used from Tensorflow. There are a lot of different Tensorflow container types you could use,(for example `.../tensorflow:latest`, which is the latest version with new updates or `.../tensorflow:nightly`, which is updated at regular periods of time) but we will be using the one with Python (version 3) and Jupyter pre-installed.

The container starts running after you issue the command. To start using this environment, you will need to go to your web browser and type the IP-address of the computer/server, from which you are running the container. To find out what the IP-address of your server is, issue the `ifconfig` command in the terminal and search for an IP-address which starts with `192.168...`. You will need to add: 2000 after that in your browser to access the Jupyter online environment. The IP-address you will need to provide in your browser could look something like this: `http://192.168.1.60:2000` or if you are running the container on the same laptop/pc, it will look something like this: `http://localhost:2000`.

Because this is the first time you try to log into this environment, you will need to set a password and provide the token. If you look at your console, you will find a token, which will be needed to change the master password to your Jupyter environment. Insert this token into your browser (token field) and enter a new master password. After logging in, you can see three folders: archive, special-project and Tensorflow-tutorials (as can be seen below in Image 4.2).



Select items to perform actions on them.

archive	een paar seconden geleden	
special-project	een paar seconden geleden	
tensorflow-tutorials	4 maanden geleden	

Image 4.2: Post installation Jupyter homepage

To check out the tutorials posted by Tensorflow, check the Tensorflow-tutorials folder. Now you can upload the datasets to this environment and start developing your AI. It's important to keep all of your files in the archive or special-project folders as they are saved outside the container. This ensures that if the container somehow gets corrupted, your files will not be lost.

3.1.2 Setting up your online environment with Google Drive and Collab

If you do not want to use your own hardware and go through the difficult setup process, you can use Google Drive and Google Collab instead. To start, you simply go to your Google Drive (drive.google.com) and log in using your Google account. It's recommended to create a new folder where you will be storing all of your project files. You should always save your files in one particular place, as they will be easier to find. After uploading your datasets, it is time to start the development of your AI. This can be done by creating a new Google Collab file (click 'New', then 'More', then Google Colaboratory in Google Drive).

3.2 Preparing the datasets

Every AI needs some sort of dataset to be able to learn. These datasets can range from 'RGB colors in every pixel of a collection of images' to 'the price of stocks on the market measured at every time stamp'. These 'out of the box' datasets are almost never in the right format to be properly used by the AI. So the datasets have to be reshaped in order to be useful. A very common file format to store datasets is the `.csv` format. It is also standardized nowadays. The values in a CSV file are separated by commas and every data record is written on a new line. The data often does not have the right format to be directly used for training. When this is the case, the data needs to be reshaped. Libraries like Pandas and Numpy have very useful tools for situations like these but it can also be done manually by writing a function to loop over the code.

3.2.1 Normalization

In most cases, you will want to normalize the data. The AI learns by slightly changing values in the nodes every time. This will result in an ineffective training if the input values have a wide range since the AI needs to make a lot of small changes. If the data is normalized to a range within 0 and 1, the underlying structure will be preserved, and the changes that need to be made are smaller, thus the training is more effective.

3.2 Example

The datasets provided for this project are two CSV-files. Both cover a time period of six months and have measurements with a one minute time interval. One dataset contains the data about the volatility (for each time interval) and one dataset contains the server load (also for each time interval). The server load does not have measurements on every timestamp however. The code below shows how to reshape these datasets using Python so it will filter out the useful data and store the data in a way it is readable by the model. This data will be stored in .npy files which can be read using the Numpy library.

Both of these datasets have around 290,000 time-stamps with the measurements: traffic and volatility. That is an array of 290,000 by two points. Which will be noted as an array of (290,000, 2) (290,000 along the x-axis and 2 along the y-axis). This AI needs a large number of points to base its predictions on. The AI will be trained in batches with each batch containing a few “examples” of input and output. For example: if 1000 points are used to make predictions (the reason why multiple points are used, is explained here), there is an array of (1000, 2). But the AI needs to be trained with more of these arrays. If in total 64 (the reason why 64 is chosen as the batch size is explained here) of these arrays are used to train the AI, there are 64 arrays of (1000, 2) which can be made into one three-dimensional array of (64, 1000, 2) (64 along the z-axis, 1000 along the x-axis and 2 along the y-axis).

Start by importing your folder with the datasets using the code below.

```
from google.colab import drive
drive.mount('/content/drive')
%cd drive/My Drive/AI-project/datasets
```

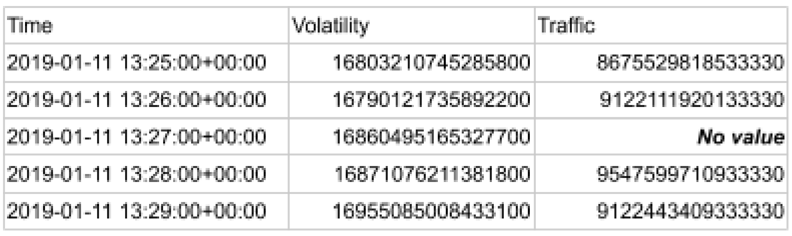
In the first line, import the drive function from Google Collab, which will be used to access the shared drive folder in which the datasets are stored. The second line specifies the location where the folder will be mounted within the workspace and the third line sets the folder, which will be mounted at the location mentioned in the second line.

Then import the `pandas` library (set the name of the library to `pd`, so you can call up the library using a shortcut) and both of the `.csv` dataset files:

```
import pandas as pd
volat = pd.read_csv("enchanted_volatility.csv", header=0)
traffic = pd.read_csv("enchanted_traffic_load.csv", header=0)
```

The Pandas library is being used because it has a `read_csv` function, which can read/import CSV-files. After this, we will start to extract useful training data from the datasets. An important note is that the datasets sometimes have gaps (as can be seen in Image 4.3) where it looks like no data was collected.

These gaps cannot be used while training, as they do not provide a complete data point. The following code looks in both of the datasets and compares if there are timestamps where both the volatility and traffic are present.



Time	Volatility	Traffic
2019-01-11 13:25:00+00:00	16803210745285800	8675529818533330
2019-01-11 13:26:00+00:00	16790121735892200	9122111920133330
2019-01-11 13:27:00+00:00	16860495165327700	***No value***
2019-01-11 13:28:00+00:00	16871076211381800	9547599710933330
2019-01-11 13:29:00+00:00	16955085008433100	9122443409333330

Image 4.3: Missing traffic values

The code below starts with defining a function. When this function is called, all the code below is executed at once. After this, the needed libraries are imported and a ‘stopwatch’ to measure the time of the reshaping process is started.

```
def collect_data_point(start_point, amount_of_points):
    import pandas as pd
    import time
    import numpy as np
    start_time = time.time()
```

The code below sets a starting point for the program to start collecting data from the original file and makes two arrays to store the useful data. The first array of ‘trainingdata’ stores the input points that the AI bases its prediction on. The second array, called ‘expected’ stores the output data that is expected from the AI.

```
    current_row = start_point
    current_row_data = 0
    trainingdata = np.zeros((amount_of_points, 2))
    expected = np.zeros(2)
```

The code below loops over the dataset starting from ‘start_point’ until the array is filled and looks for timestamps where both values are present (if there is a volatility present at the same time a traffic is present, add it to the array).

```
while (current_row_data < amount_of_points):
    traf = traffic.loc[traffic["indexnumbers"].isin([str(current_row)]),
["time", "traffic"]]
    vol = volat.loc[volat["time"].isin([str(traf.values[0][0])]), "volit"]

    if (len(vol.values) > 0):
        trainingdata[current_row_data, 0] = float(traf.values[0][1])
        trainingdata[current_row_data, 1]= float(vol.values[0])
        current_row_data += 1
    current_row += 1
```

The next part of the code looks for the volatility with the same time and takes the traffic and time. Then it takes the next minute after as the ‘expected’ result. At the end, it returns the arrays as output of the function collect_data_points as mentioned above.

```
voll = volat.loc[volat["time"].isin([str(traf.values[0][0])]), "volit"]
while len(voll.values == 0):
    traff = traffic.loc[traffic["indexnumbers"].isin([str(current_row)]),
["time", "traffic"]]
    voll = volat.loc[volat["time"].isin([str(traff.values[0][0])]),
"volit"]
    current_row += 1

expected[0] = float(traf.values[0][1])
expected[1] = float(vol.values[0])

return (trainingdata, expected)
print("collect_data_point took {} seconds
---".format(time.time()-start_time))
```

This is one of the ways to reshape a dataset. Of course there are other ways of reshaping data using Python, but for now we will be sticking to the code we used and explained above.

4. Creation

After preparing your environment and datasets, you are ready to begin the development process of your AI.

4.1 Building a simplified AI (skeleton)

To start the building process, it is advisable to make a very simple AI with just one layer. This AI will probably not predict any accurate data but it is used to make sure that the dataset has the right structure, the output has the right shape and that everything is working as it should. When this is set up and running, most of the necessary code has already been written. From here, the only things that need to be changed are the layers, optimization and loss (and sometimes metrics too). In this step, not only will the model be created but also the AI prediction part will be constructed.

Start by importing all of the libraries which are used to create the model and can be seen below. Renaming Tensorflow and Numpy isn't necessary, but using abbreviations makes the development process a bit easier as you use those libraries often.

```
import tensorflow as tf
from google.colab import output
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
import numpy as np
```

Next, you want to import the datasets with the correct extension (it should be `.npy` as it is a Numpy array). Notice that the Numpy files are located within a folder called npy in this instance, so it's important to add the directory location as well. The batch size variable and the amount_of_points variable should also be set as they will be used when training the models.

```
trainX = np.load("npy/newdata.npy")
trainY = np.load("npy/newdataoutput.npy")
batch_size = 32
amount_of_points = 100
```

After importing the datasets you need to specify the model layers and their activation. We will be using two LSTM layers and one Dense layer for this model as this is sufficient for our example.

```
model = Sequential()
model.add(LSTM(64, input_shape=(2, amount_of_points),
return_sequences=True, activation=activation))
model.add(LSTM(32, activation=activation))
model.add(Dense(1, activation=activation))
```

The next step is to set an `optimizer`, which is one of the two arguments required for compiling a Keras model. The `tensorboard_callback` will allow Tensorboard to display information about the nodes, loss and metrics during the training. This is useful for checking if the training is going according to plan and can be done as follows.

```
optimizer = tf.keras.optimizers.Adam(learning_rate=learning)
model.compile(loss=loss, optimizer=optimizer, metrics=["mae"]) # other
metrics

tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir="logs/",
histogram_freq=1)
```

The basic code for your ‘AI skeleton’ should look something like the four code above combined. However, your final AI is not done yet. You will still need to do the fitting and optimization of your model. Optionally, you could add the `model.summary()` function at the end, which will show you the layer layout, so you can see if everything is working correctly. This function will output the following:

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_4 (LSTM)                (None, 2, 128)            117248
_________________________________________________________________
lstm_5 (LSTM)                (None, 96)                86400
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 97
=================================================================
Total params: 203,745
Trainable params: 203,745
Non-trainable params: 0
```

5. Training & Refining

5.1 Running the dataset through the AI

The AI will be trained with the prepared data from 3.2 Preparing the datasets. The data is divided into two arrays: one with the input data (often referred to as trainX) and one with the expected output (often referred to as trainY). Both are needed to train the AI.

```
import time
start_time = time.time()
history = model.fit(np.array(trainX), np.array(trainY), epochs=200,
batch_size=batch_size, verbose=2, shuffle=False)

print("--- fitting model took {} seconds / {} minutes
---".format(int(time.time()-start_time), round((time.time()-start_time)/60,
2))) # print the time it took to execute
```

This piece of code trains (fits) the AI with the datasets `trainX` and `trainY`. The training session is stored to a variable called `history` in which can later be seen how well the training went. The amount of `epochs` states how often the AI trains with the same given dataset. If the AI is trained too often with the same dataset, it will overfit and won't be precise in predicting new data. The `batch_size` defines the amount of examples it trains with at the same time. `verbose` tells how much information of this training session is being displayed as output. A value of 0 means that nothing is shown, 1 means a loading bar per epoch and 2 means that the loss and metrics are displayed every epoch. It is recommended to start with `verbose=2` when starting with machine learning since it will show useful information.

5.2 Making small adjustments and checking for improvement with visualizations

5.2.1 Loss and Metrics

When the AI trains, its goal is to get the `loss` as low as possible. With verbose=2 on during the training, it can be seen how high the loss is and how it is developing between epochs. This development can be shown in a graph to visualize how well the training is going. The desired graph should be somewhat similar to the graph in Image 4.4. This image was generated from our training and is the same as other examples where the loss should be descending.

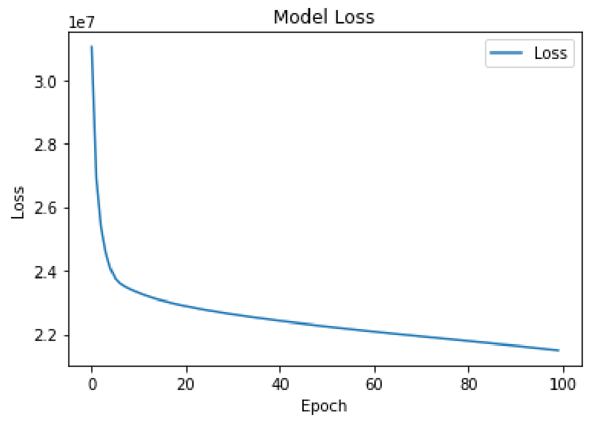


Image 4.4: Correct loss graph after training

During the first few epochs, you can see that the AI is guessing. Quickly after that, the AI begins to understand the general idea and huge progress is made. This allows the AI to refine its predictions. However, because of this, it will not learn that much per epoch. The AI gradually improves, but because the loss gets smaller, the predictions are fewer and the graph flattens out. This is the desired structure when training an AI. The loss should always go down and flatten out at some point.

The graph on the right (Image 4.5) is an example of what the training should **not** look like. When the graph looks like this, there is a big chance that the loss will still be high at the end. When the loss fluctuates, it is hard to get a stable training session since the loss can really differ between two aligned epochs. If you would take 201 epochs instead of 200 for example, the loss can change a lot.

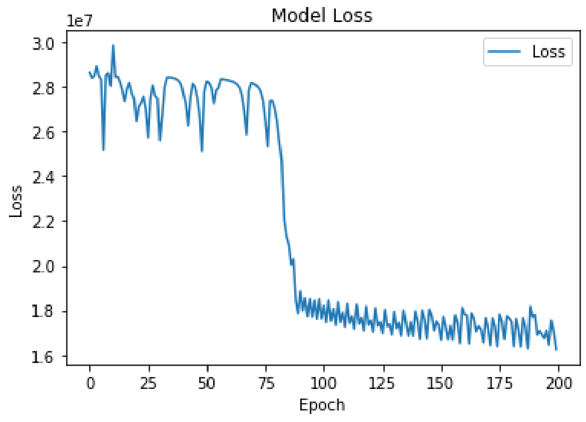


Image 4.5: Bad fluctuation of the loss

These graphs can easily be drawn when making use of Matplotlib as shown with the code below.

```
import matplotlib.pyplot as plt

plt.plot(history.history['loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['MAE'], loc='upper right')
plt.show()
```

The Matplotlib library is imported in the first line. The `plt.plot(history.history['loss'])` tells to draw the loss function of the previous data (defined with `history`). The titles/labels of the graph, line and axis are defined below. And on the last line, a Matplotlib function called `show()` is used, which displays the created graph.

5.2.2 Predictions

Visualizing the loss will make clear how well the training is going but that does not necessarily mean that the predictions are accurate as well. The best way to visualize the data is to make a few predictions and draw those in a graph along with the actual data at these points. With this as a visualization, you can see how well the predictions of the AI actually are.

In Keras, a function is included to make predictions using the trained model. This function is called upon using the following code.

```
prediction = model.predict(np.array(test_input), batch_size=batch_size)
```

The trained model will try to predict an outcome given the `test_input`. This `test_input` is formatted in the same way as the `trainX` array. The only difference is that this array only has one thing to calculate and not whole batches. Additionally, in this function there will not be a check if the AI was right. The output will be stored in `prediction` and available to be used to fulfil a function the AI was intended for.

5.2.2 Example

The AI predicts based on 100 previous points of volatility and traffic and we are trying to predict 10 points. Therefore, a dataset is created with 110 points. The AI will loop over these points, each time using 100 points to predict the consecutive point.

5.2.3 Making adjustments to the datasets

In some cases, the dataset needs to be changed. This is because in some datasets, there is invalid or missing data. The AI would be able to make better predictions when training if it were based on more accurate or complete datasets. Therefore, in these situations, it is sometimes useful to make multiple datasets. One would have the original data and one would have the edited data. Multiple AI can be trained with each dataset. From the predictions of these AI, it can be concluded which dataset allows for the best predictions.

5.3 Saving the final AI

The process of training and refining will continue until the best solution is found. The best solution is often the best one found within the time limit. The trained AI can be saved at this point with the code below.

```
from keras.models import load_model

model.save('my_model.h5')  # creates a HDF5 file
del model  # deletes the existing model
model = load_model('my_model.h5') # loads the saved model
```

The model can be saved in a directory defined by a string. This should be an HDF5 file with the extension `.h5`. This means that `model_file` should be something like `"model_5.p5"`.

To load the saved model again, `keras.models.load_model(model_file)` is used. When the model is loaded, it can be used again in the same way as before. The reason why saving a model is useful is that it does not have to be stored in the RAM but on your hard drive instead. It can also be accessed at any time without having to retrain the AI.

Summary: Guide for Building an AI

This chapter has given brief, yet detailed instructions for building a simple AI. This is meant to inspire and intrigue the reader to explore the actual construction of an artificial intelligence program. It also opens a window into the complexity of the workings of AI applications that people use every day.

Epilogue

First of all we would like to thank Amie de Jeu and Tobias Meij for checking the final report for spelling and grammar mistakes. Secondly, we would like to thank R. de Vries for helping us with the structure of this report and our planning. Finally, we would like to thank Stephen Helms for guiding us through the process of making an AI and helping us at moments when we were struggling to solve a problem.

During this project, we learned a lot about the history of AI, as well as the fundamental elements and how to make an AI by ourselves. The scope of the report was broad which gave us the opportunity to explore many aspects of AI. However, as a result, we did not go as in depth as we would have wanted. Additionally, given more time, we would have wanted to do more research on the scientific aspect. Everyone in our team had their own strengths and they fit together perfectly. Some of us became even more interested in AI as a result of working on this report, and plan to continue studying this subject in our free time.

Sources

1. Nicholls, P. (1981) The Encyclopedia of Science Fiction, Granada, p. 258.

2. Asimov, Stanley (1996). Yours, Isaac Asimov. My estimate is that Isaac received about 100,000 letters in his professional career. And with the compulsiveness that has to be a character trait of a writer of almost 500 books, he answered 90 percent of them. He answered more than half with postcards and didn't make carbons of them. But with the 100,000 letters he received, there are carbons of about 45,000 that he wrote.

3. Asimov, I. (1942). Runaround. Retrieved from http://ekladata.com/-Byix64G_NtE0xI4A6PA1--o1Hc/Asimov-Isaac-I-Robot.pdf.

4. Asimov, I. (1986). Foundation and Earth. Retrieved from https://openlibrary.org/works/OL46347W/Foundation_and_Earth.

5. Who's Who. ukwhoswho.com (online Oxford University Press ed.). (2017). Turing, Alan Mathison. an imprint of Bloomsbury Publishing plc. doi: https://doi.org/10.1093/ww/9780199540884.013.U243891 (subscription required).

6. Beavers, A. (2013) Alan Turing: Mathematical Mechanist. In Cooper, S. Barry; van Leeuwen, Jan (eds.). Alan Turing: His Work and Impact. Waltham: Elsevier. p. 481–485. ISBN 978-0-12-386980-7.

7. Copeland, J. (18 June 2012). Alan Turing: The codebreaker who saved 'millions of lives'. BBC News Technology. Retrieved version from archive.org of date 11 October 2014.

8. Central Security Service, National Security Agency. (n.d.). WAVE Demonstrating Bombe [Photograph]. Retrieved from https://www.nsa.gov/Resources/Everyone/Digital-Media-Center/Image-Galleries/Cryptologic-Museum/Machines/igphoto/2002138771/.

9. Margallo, J. A. S. (2017). File:Turing test diagram.png - Wikimedia Commons [Graph]. Retrieved from https://commons.wikimedia.org/w/index.php?curid=57298943.

10. Turing, A. (1950). Alan Turing Scrapbook - Turing Test. Retrieved from https://www.turing.org.uk/scrapbook/test.html

11. Férée, H. (2011). File:The Imitation Game.svg - Wikimedia Commons [Graph]. Retrieved from https://commons.wikimedia.org/w/index.php?curid=17059778.

12. Saygin, A. P.; Cicekli, I.; Akman, V. (2000), Turing Test: 50 Years Later (PDF), Minds and Machines.

13. Férée, H. (2011). File:Turing Test Version 1.svg - Wikimedia Commons [Graph]. Retrieved from https://commons.wikimedia.org/w/index.php?curid=17059705.

14. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627

15. McCarthy, J.; Minsky, M.; Rochester, N.; Shannon, C. (31 August 1955), A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence

16. McCorduck, P. (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627.

17. Crevier, D. (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 44–46.

18. Crevier, D. (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, p. 49.

19. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627

20. Simon, H. A.; Newell, Allen (1958), "Heuristic Problem Solving: The Next Advance in Operations Research", Operations Research, 6: 1, doi:10.1287/opre.6.1.1

21. Simon, H. A. (1965), The Shape of Automation for Men and Management, New York: Harper & Row p. 96

22. Minsky, M. (1967), Computation: Finite and Infinite Machines, Englewood Cliffs, N.J.: Prentice-Hall.

23. Minsky strongly believes he was misquoted. See McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627.

24. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627 pp. 247–248.

25. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627 pp. 245–250

26. Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2 p. 19

27. Crevier, D. (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 79–83.

28. Public Domain, Anonymous author. (2007). File:Semantic Net.svg - Wikimedia Commons [Graph]. Retrieved from https://commons.wikimedia.org/w/index.php?curid=1353062.

29. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627 pp. 291–296

30. Crevier, Daniel (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 83–102

31. Crevier, Daniel (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 84–102

32. Russell, Stuart J.; Norvig, Peter (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2, pp. 21–22

33. Crevier, Daniel (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 146–148

34. Newquist, HP (1994), The Brain Makers: Genius, Ego, And Greed In The Quest For Machines That Think, New York: Macmillan/SAMS, ISBN 978-0-9885937-1-8, p. 271

35. Crevier, D. (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3.

36. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627, pp. 434–435

37. Pickell, D. (2018, November 16). Structured vs Unstructured Data – What's the Difference? Retrieved from https://learn.g2.com/structured-vs-unstructured-data.

38. Garson, J. (2018, November 27). Zalta, E. N. (ed.). The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University – via Stanford Encyclopedia of Philosophy.

39. Newquist, HP (1994), The Brain Makers: Genius, Ego, And Greed In The Quest For Machines That Think, New York: Macmillan/SAMS, ISBN 978-0-9885937-1-8, pp. 231–240

40. Crevier, Daniel (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 195

41. The National Academies of Sciences, Engineering, and Medicine (1999), "Developments in Artificial Intelligence"

42. Roland, Alex; Shiman, Philip (2002). Strategic Computing: DARPA and the Quest for Machine Intelligence, 1983-1993. Cambridge, Mass.: MIT Press. ISBN 0-262-18226-2, p. 2

43. Crevier (1993), AI: The Tumultuous Search for Artificial Intelligence, New York, NY: BasicBooks, ISBN 0-465-02997-3, pp. 203. AI winter was first used as the title of a seminar on the subject for the Association for the Advancement of Artificial Intelligence.

44. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627, pp. 430–431

45. Newquist, HP (1994). The Brain Makers: Genius, Ego, And Greed In The Quest For Machines That Think. New York: Macmillan/SAMS. ISBN 978-0-672-30412-5.

46. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 978-1-56881-205-2, OCLC 52197627, pp. 480–483

47. 'Maloof, M. (2017). Artificial Intelligence: An Introduction. Retrieved on September 14, 2019, from http://people.cs.georgetown.edu/~maloof/cosc270.f17/cosc270-intro-handout.pdf [p37].

48. AIMultiple. (2019, 12 September). 373 experts opinion: AGI / singularity by 2060 [2019 update]. Retrieved on October 15, 2019, from https://blog.aimultiple.com/artificial-general-intelligence-singularity-timing/.

49. Garbade, M. J. (2018, 14 September). Clearing the Confusion: AI vs Machine Learning vs Deep Learning Differences. Retrieved on October 15, 2019, from https://towardsdatascience.com/clearing-the-confusion-ai-vs-machine-learning-vs-deep-learning-differences-fce69b21d5eb.

50. Martin, N. (2019, 19 March). Machine Learning And AI Are Not The Same: Here's The Difference. Retrieved on September 15, 2019, from

https://www.forbes.com/sites/nicolemartin1/2019/03/19/machine-learning-and-ai-are-not-the-same-heres-the-difference/#216b08e51b56.

51. Samson, O. (2017, 11 June). Deep learning weekly piece: the differences between AI, ML, and DL. Retrieved on September 19, 2019, from https://towardsdatascience.com/deep-learning-weekly-piece-the-differences-between-ai-ml-and-dl-b6a203b70698.

52. Russell, S. J., Norvig, P. N., & Davis, E. D. (2010). Artificial Intelligence: A Modern Approach. Upper Saddle River, New Jersey: Prentice Hall.

53. Russell, S. J., Russell, S. J., Norvig, P., & Davis, E. (2010). Artificial Intelligence: A Modern Approach. Upper Saddle River, New Jersey: Prentice Hall.

54. Horan, T. J. (2018, 3 July). 5 Keys to Using AI and Machine Learning in Fraud Detection. Retrieved on October 15, 2019, from https://www.fico.com/blogs/5-keys-using-ai-and-machine-learning-fraud-detection.

55. Russell, S. J., Russell, S. J., Norvig, P., & Davis, E. (2010). Artificial Intelligence: A Modern Approach. Upper Saddle River, New Jersey: Prentice Hall.

56. Haugeland, J. (1985). Artificial Intelligence: The Very Idea. Cambridge, Mass.: MIT Press. ISBN 978-0-262-08153-5.

57. Nilsson, N. (1998). Artificial Intelligence: A New Synthesis. Morgan Kaufmann. ISBN 978-1-55860-467-4.

58. McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1.

59. Luger, George; Stubblefield, William (2004). Artificial Intelligence: Structures and Strategies for Complex Problem Solving (5th ed.). Benjamin/Cummings. ISBN 978-0-8053-4780-7.

60. Stanley, K. O. (2004). Efficient Evolution of Neural Networks through Complexification. Retrieved on January 5, 2020, from http://nn.cs.utexas.edu/downloads/papers/stanley.phd04.pdf.

61. Walia, A. S. (2017, May 29). Activation functions and its types. Retrieved January 25, 2020, from https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f.

62. Core Layers - Keras Documentation. (2019). Retrieved December 15, 2019, from https://keras.io/layers/core/.

63. 1) Tutorial: Overfitting and Underfitting. (2019). Retrieved from https://keras.rstudio.com/articles/tutorial_overfit_underfit.html.

64. Yiu, T. (2019, December 6). Understanding RNNs (Recurrent Neural Networks). Retrieved December 16, 2019, from https://towardsdatascience.com/understanding-rnns-recurrent-neural-networks-479cd0da9760.

65. 1) Olah, C. (2015, August 27). Understanding LSTM Networks -- colah's blog. Retrieved on December 15, 2019, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

66. Russell, S. J., Russell, S. J., Norvig, P., & Davis, E. (2010). Artificial Intelligence: A Modern Approach. p 1053 to p 1059 to Retrieved from https://faculty.psau.edu.sa/filedownload/doc-7-pdf-a154ffbcec538a4161a406abf62f5b76-original.pdf.

67. Dimitri P. Bertsekas, Nonlinear Programming, Athena Scientific 1999, 2nd edition, pp. 187.

68. PyTorch Homepage. Retrieved December 15, 2019, from https://pytorch.org/.

69. Homepage — NumPy. (n.d.). Retrieved December 15, 2019, from https://numpy.org/.

70. Matplotlib: Python plotting — Matplotlib 3.1.2 documentation. (n.d.). Retrieved December 17, 2019, from https://matplotlib.org/.

71. Time — Time access and conversions — Python 3.8.1 documentation. (n.d.). Retrieved December 17, 2019, from https://docs.python.org/3/library/time.html.

72. Git. (n.d.). Retrieved October 17, 2019, from https://git-scm.com/.

73. StackOverflow. (2017, May 5). [Forum post]. Retrieved December 20, 2019, from https://stackoverflow.com/questions/43985250/what-are-the-minimum-system-requirements-for-executing-a-simple-project-in-tenso.

74. McCarthy, J. (n.d.). The Philosophy of AI and the AI of Philosophy. Retrieved November 30, 2019, from http://jmc.stanford.edu/articles/aiphil2.html.

75. Russell, S. J.; Norvig, P. (2003), Artificial Intelligence: A Modern Approach (2nd ed.), Upper Saddle River, New Jersey: Prentice Hall, ISBN 0-13-790395-2.

76. McCarthy et al. 1955. This assertion was printed in the program for the Dartmouth Conference of 1956, widely considered the "birth of AI." also Crevier 1993, p. 28.

77. 1) Newell, A.; Simon, H. A. (1976). Computer Science as Empirical Inquiry: Symbols and Search, Communications of the ACM, 19, archived from the original on 2008-10-07.

78. Searle, John (1999), Mind, language and society, New York, NY: Basic Books, ISBN 978-0-465-04521-1, OCLC 231867665

79. Hobbes, T (1651). Leviathan, chapter 5.

80. Hobbes, T. (1982). Leviathan (Penguin Classics) (59364th ed.). New York, NY: Penguin Random House.

81. Bringsjord, Selmer and Govindarajulu, Naveen Sundar, "Artificial Intelligence", The Stanford Encyclopedia of Philosophy (Winter 2019 Edition), Edward N. Zalta (ed.), https://plato.stanford.edu/archives/win2019/entries/artificial-intelligence/

82. Mozur, P. (2018, September 22). Inside China's Dystopian Dreams: A.I., Shame and Lots of Cameras. Retrieved 30 January 2020, from https://www.nytimes.com/2018/07/08/business/china-surveillance-technology.html

83. OpenAI. (2018, April 9). OpenAI Charter. Retrieved 30 January 2020, from https://openai.com/charter/.

84. Russell, S., & Norvig, P. (2016). Artificial Intelligence: A Modern Approach, Global Edition. Retrieved from http://aima.cs.berkeley.edu/

85. Miliard, M. (2019, February 27). Google, Verily using AI to screen for diabetic retinopathy in India. Retrieved from https://www.healthcareitnews.com/news/google-verily-using-ai-screen-diabetic-retinopathy-india.

86. A BETTER WAY FORWARD | tusimple. (n.d.). Retrieved 3 Februari, 2020 from https://www.tusimple.com/.

87. https://www.descarteslabs.com/industries/agriculture/

88. Butler, S. (1999). Erewhon; Or, Over the Range. Retrieved from http://www.gutenberg.org/ebooks/1906.

89. Butler, S. (2004). The Note-Books of Samuel Butler. Retrieved from http://www.gutenberg.org/ebooks/6173.

90. Dyson, G. B. (1998). Darwin Among The Machines: The Evolution Of Global Intelligence (Helix Books) (First Trade Paper Edition). New York: Basic Books.

91. StackOverflow. (2017, May 5). [Forum post]. Retrieved December 20, 2019, from https://stackoverflow.com/questions/43985250/what-are-the-minimum-system-requirements-for-executing-a-simple-project-in-tenso.