Learning Algorithms
What is Artificial Intelligence Really Capable of?

Rainer Berkemer, Markus Grottke

I. Introduction

Intelligent machines and algorithms, mostly subsumed under the name of artificial intelligence, are pervading ever more areas of modern life. In doing so, they are increasingly presenting results that exceed the capabilities of humans in the respective area of application. Examples are Alpha Go in the Asian game of Go, AI-based chess computers, ChatGPT or the Watson system in the television show Jeopardy. But what can really be concluded from such results with respect to the capabilities of artificial intelligence?

One of the great challenges of analyzing artificial intelligence with regard to such a question is that – outside of certain formal rules – nobody knows the why, that is, for what reason the algorithms in question learn, i.e. how they finally acquire their capabilities and why they act as they act. This is especially true for learning algorithms based on neural networks as explainable AI still remains the exception in this area, even though research in explainable AI is currently starting to mushroom (Gunning et al. 2019: 2; Samek/Müller 2019: 13).

On the other hand, it is the responsibility of university teachers to communicate the implications of such a lack of understanding and to enable a constructive but at the same time critical approach to such learning algorithms. In this respect it might be of interest that many of the questions that currently arise were already discussed by the early pioneers of artificial intelligence, such as Alan Turing, creator of the Turing test, and Norbert Wiener, founder of cybernetics.

Referring back to and expanding on such discussions, the following article poses, based on the famous example of AlphaZero, one central question and derives from it some important points for discussion, namely: Based

on the concrete example of AlphaZero: What does artificial intelligence do and what are the implications of how artificial intelligence is perceived? This points to questions like: What does the lack of understanding – e.g. the blackbox character – that is typical for artificial intelligence really involve? And – based on this – what is the social responsibility of decision makers such as artificial intelligence developers? What social responsibility do economic decision makers of funding for the development of artificial intelligence have? What effects on society and the working world can be expected from the use of artificial intelligence?

The rest of the paper is organized as follows. First, neural networks in general are discussed. Also, important definitions are formulated which prepare the groundwork for the following analyses but also limit the field of analysis. In the next chapter, learning algorithms are discussed, especially those that have been employed in cases where the performance of artificial intelligence compared to human intelligence is discussed. This part of the paper deals with these general questions by making use of the example of AlphaZero. This learning algorithm has been chosen, as chess has long since been regarded as the "drosophila" of AI. Unlike other chess programs, AlphaZero is based on a "general reinforcement algorithm". This suggests that it could be "strong AI" because the principle can be applied easily to other games, as well as strategic situations – in fact, however, the historical course of events was the other way around. An application to the board game Go took place first and only after it was surprisingly successful in that domain was the principle later transferred to chess and shogi. Finally, some considerations on this basis are shown with respect to the discussions around the emerging question of social responsibility for artificial intelligence, before the paper concludes with an overview of the results, limitations of the present analyses and an outlook.

II. Principles of Neural Networks and Possible Definitions of (Artificial) Intelligence

II.1 Neural Networks: Basic Principles and Typical Areas of Application

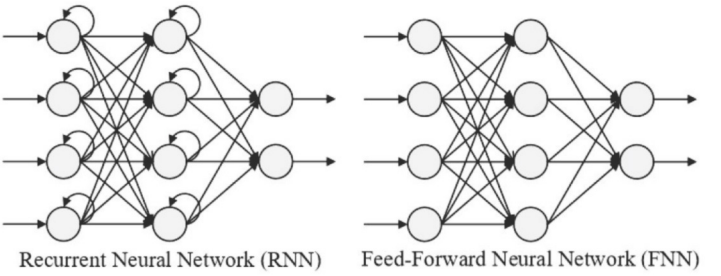
The starting point of an artificial neural network is a schematic reproduction of a biological model, a neuron, which is artificially reproduced in the case

of a neural network. A single neuron has any number of inputs but always only one output.

Within a single neuron, the input information is processed – mostly by *weighted summation* – and the output signal is determined depending on the threshold values that are exceeded or not. This procedure is also closely related to the biological model, in which a nerve cell fires when a threshold voltage is exceeded, i.e. transmits an electrical signal.

The described mechanism becomes relevant with respect to artificial neural networks as soon as a large number of neurons are assembled into more complex structures, typically grouped into layers, with the flow of information usually proceeding forward from the *input layer* to the *output layer*. Figure 1 shows two common network structures.

Figure 1: Two common structures of neural networks (Nguyen et al. 2018: 6)



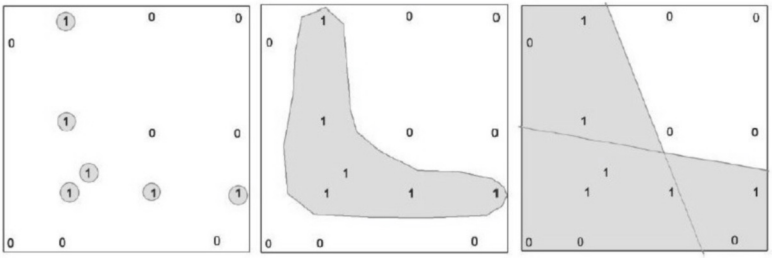
The layers in between are called *hidden layers* and modern neural networks, as they are often used in the field of pattern recognition, typically have a huge number of such hidden layers. These are commonly also called Deep Neural Networks (DNN) insofar as they exhibit a sufficiently high number of layers.

Typically, the task of a neural network is to train artificial neurons to correctly distinguish an unknown data set that follows the same underlying rules as the classification patterns used to train the neural network from a data set that does not follow such classification patterns.

For this purpose, a neural network consisting of artificial neurons is trained using an initial data set with labelled data that has already correctly assigned outputs so that the result of this training could be used for classifying patterns that are still unknown.

One of the challenges in training is that the respective neural network can be over-trained (so-called over-fitting) or under-trained (so-called under-fitting). This is illustrated in figure 2.

Figure 2: Over-fitting, neural network with adequate generalization, under-fitting (Kriesel 2007: 86 adjusted by Jerems 2018: 49)



The left of the three figures shows an overtrained network, while on the right, we can see an undertrained network and the middle of the three figures represents an adequately generalizing network. The overtrained network only concretizes single cases, but has no generalization anymore, as the small circles around the single cases indicate. Thus, it is questionable whether it would still correctly classify an unknown pattern. An undertrained neural network has false positives (as seen in in the classified zeros in the figure on the right). A good classification, on the other hand, separates adequately but also goes beyond pure learning by rote.

II.2 Intelligence

When dealing with learning algorithms in the area of artificial intelligence, we should first think about what we mean when we employ the notion of intelligence. For example, some possible characteristics might immediately come to mind, such as curiosity, long-term-planning, or learning capabilities. One approach to address the notion of intelligence, for example, can be based on the so-called Turing test from 1950.

The Turing test defines artificial intelligence by comparison with a human being as follows:

Before dealing with artificial intelligence one should first develop an understanding of natural (human) intelligence. When we then have to decide whether one should attribute "intelligence" also to a machine, the most obvi-

ous approach is given when artificial intelligence is assigned to a computer or to a machine if it can successfully imitate human answers under certain conditions.

Specifically, a human observer must not be able to determine several times without error whether an answer to a posed question was given by a computer or by another human being. If the person cannot distinguish this, the machine has "passed" the Turing test (cf. Turing 1950).

The first chatbot Eliza by Joseph Weizenbaum (MIT 1967), for example, managed to give its users the impression of talking to a counterpart full of empathy and understanding by skillfully manipulating symbols. The following dialogue may illustrate the idea.

Person: Men are all alike.
ELIZA: In what way?
Person: They're always bugging us about something or other.
ELIZA: Can you think of a specific example?
Person: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here?
Person: He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
Person: It's true. I am unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
Person: I need some help, that much seems certain.
ELIZA: What would it mean to you if you got some help?
Person: Perhaps I could learn to get along with my mother.
ELIZA: Tell me more about your family. (Kendall 2001)

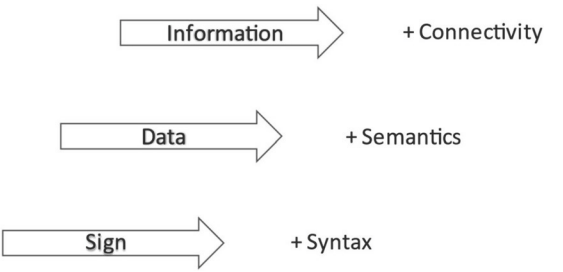
It seems to be significant here that Eliza gets along without semantics, i.e. did not need to understand the meaning of the terms uttered. A knowledge base was sufficient that, for example, connects father, mother and son etc. with family, i.e. that knows the correct use of syntax. Nevertheless, it might pass a Turing test – at least a sufficiently superficial one. Obviously, the currently discussed ChatGPT would also pass the Turing test.

More suitable for our purposes as a method to classify artificial intelligence as intelligent or not is, however, the Chinese room framework that was developed by the linguist John Searle. It assumes that a computer would pass the Turing test and is, additionally, based on English instructions (!) able to convince a human Chinese speaker that the AI program is a living Chinese speaker. What Searle is pointing at here is that it is necessary

to avoid the very deception that Eliza was undergoing. The key question that separates the two is whether a program really understands Chinese or whether it is just simulating it. To test this, the following situation is established: an English-speaking person without any knowledge in Chinese sits in a closed room with a book full of English instructions. Chinese letters are passed into the room through a slit, the person reads the English instructions and on this basis, that is by taking recourse to the instructions and the incoming Chinese letters, also passes Chinese letters to the outside. Searle argues that a person acting in this way is as intelligent as a program based on input and output. But Searle also makes clear that this does not mean that the computer can think: neither computer nor human will think in such a framework, but rather both of them mindlessly follow certain instructions (for more elaborations on the Chinese room cf. Walch 2020).

In this respect, a precise differentiation is required in each case as to when which form of knowledge is received. These can be distinguished from each other according to the following figure 3, which shows the beginning of a knowledge pyramid.

Figure 3: Sign – data – information (North 2021: 37)



If you follow the examples mentioned so far, Eliza takes you to the level of syntax situated in the knowledge pyramid, i.e. the linking of characters according to given rules. Similarly, the program as the human in the Chinese room experiment use and apply only knowledge at this syntax level and do not really understand what they are doing. However, in the Eliza example the syntax level limitations become somewhat apparent after some time. In the Chinese room experiment it is, on the one hand, apparent that it was the creator of the instructions and the selector of the Chinese Letters who put the intelligence in the action of the program while the program still

remains on a syntax level. However, the impression of intelligence of the program (which seems to be able to deal with Chinese) is impressive from the outside. As we will see, this is quite comparable to AlphaZero (which seems to be able to understand chess).

However, both, that is the Turing test and Chinese room, can be considered to be limited to a certain application area for, among others, the following reason: we know that people react differently to identical stimuli and identically to different stimuli because of the meaning that they attach to the words of other people as well as to their own words (Hayek 1979: 43). Neither the Turing test nor Chinese room experiment can address such a behavior because they would need (at least statistically) identical reactions to identical stimuli (according to certain given rules or instructions). It is exactly for that reason that both cannot grasp human behavior generally but only capture an imitation of such behavior by artificial intelligence.

II.3 Types of Artificial Intelligence

Artificial intelligence may be subdivided into numerous facets, a brief overview of which is provided below to the extent that it is relevant to the following discussion. According to a review article by Seifert et al. (2018: 56), AI technology can be divided into three different categories, namely behavioral ("human") AI, *systems that think and act rationally,* and biologically inspired systems. We do not pursue the topic of the latter as they are not commonly counted among the "classical" forms of AI (Seifert et al. 2018: 58).

Examples of behavioral technologies include semantic systems, natural language processing (NLP) and cognitive modelling. Semantic systems attempt to understand meaning in data by formulating relationships and dependencies. Compared to the syntax-based system Eliza, this brings us to the second stage, the linking of data using semantics.

In the case of NLP, human language is addressed in diverse forms and used via text recognition, language generation/comprehension and automatic translation (Seifert et al. 2018: 58). A potential application area for NLP are chatbots.

Cognitive modelling refers to models that mimic human intelligence and simulate human behavior (Seifert et al. 2018: 57). Systems that think and act rationally can be divided into the subfields of computer vision, robotic

process automation and machine learning. Computer Vision refers to the detection and classification of objects or actions in images or videos.

Robotic Process Automation refers to the automatic recognition and processing of routine activities in business processes (Seifert et al. 2018: 60). By means of information technology it provides a promising approach to process automation in which repetitive and time-consuming or error-prone activities are learned and automated by software robots.

II.4 Types of Learning: Supervised, Unsupervised and Reinforcement Learning

Machine learning can be differentiated into supervised learning, unsupervised learning and reinforcement learning (Bruns/Kowald 2019: 9).

Supervised learning means that an algorithm receives a training set with a classification that has already been performed and that resulted in labelled data on which the algorithm is subsequently trained. Thus, the classification to be learned or a form of regression are already given. As soon as the algorithm has acquired these, validation takes place on a further data set that was not used during training, thus allowing a performance estimate of the learning machine on data that has not yet been seen.

Unsupervised learning in the form of data mining involves trying to find patterns in data sets without explicitly specifying a learning goal in order to reveal relationships between the underlying variables. In the now much more common case, however, unsupervised learning is used to predict variable states on as yet unseen data sets.

Reinforcement learning, by contrast, seeks to find a solution to the problem by providing the algorithm with feedback in the form of incentive/punishment (Bruns/Kowald 2019: 9–10). The main difference between reinforcement learning and other machine learning paradigms, therefore, is that reinforcement learning is not only concerned with predictive performance, but with finding a course of action ("policy") under given (or presumed) environmental conditions that leads to the maximization of a previously specified quality criterion (reward).

This objective includes a high predictive power as an intermediate step, but goes beyond this with respect to its purpose, as it desires situation-appropriate actions of the AI as system output. Consequently, with the help of reinforcement learning, an AI "independently learns a strategy for solving a problem or task" (Seifert et al. 2018: 60). While computer vision and robotic

process automation, like supervised learning, do not go beyond the level of data processing described so far, unsupervised learning and reinforcement learning seem to make possible a different quality, which could go beyond the stages that described models of AI (e.g. ELIZA) reached so far in the knowledge pyramid shown in Figure 3.

In a nutshell, what the defining work up to this point reveals is that at least almost all current AI can be classified as being able to connect signs. However, to what degree artificial intelligence goes beyond is not so clear. It is only clear that it is not yet using its capabilities to act on its own. In the following, we will look at one amazing example of AI in detail. Based on our example, chess, we will elaborate very thoroughly, not only because of the very long tradition of chess as a role model for computer science and artificial intelligence applications. AlphaZero, the learning algorithm in question, which is a classical reinforcement learning algorithm, is very suitable for illustration precisely because of this, namely, because the use of artificial intelligence in chess is comparably well-understood. Whether and especially to what degree, however, the algorithm reaches a different level in the knowledge pyramid, is a question that remains to be answered when we analyze AlphaZero.

III. The Analysis of a Self-learning Algorithm: The Example of AlphaZero

III.1 Automatons Playing Chess – Historical Overview of Chess Computing Systems

For a long time chess was regarded as the drosophila of AI research, as a number of important scientists who have also made important general contributions to mathematics and logic have also tried to automate chess. Even before the necessary hardware existed, Charles Babbage, for example, conceived of a corresponding computing system in theory – in the middle of the 19th century (Bauermeister n.d.: 1).

According to Bauermeister, it was not until about 100 years later that the world's first non-mechanical functional program-controlled calculating machine was built, the famous Z3, by Konrad Zuse in 1941. A mechanical functional program-controlled machine had already been achieved with the

Z1. To program the computer, Zuse developed a language called "Plankalkül", the archetype of all of today's algorithmic programming languages. (ibid: 1)

Zuse chose the game of chess as a test object to demonstrate that such a machine can not only compute numbers (the word computer is derived from this) but that this machine, in principle, is also suitable for addressing and solving non-numerical problems.

The first algorithm ready to be implemented was developed by none other than Turing himself. In 1947, Turochamp was launched, a one-move generator with something that many later programs would pick up – a rating function that tried to incorporate chess-specific knowledge (ibid: 2). The pieces were rated according to their potential strength – a pawn was assigned the value of one unit, the minor pieces (knight and bishop) the value of 3 to 3.5 pawn units. A rook was rated as strong as 5 pawns and the queen as strong as 2 rooks.

This form of so-called material evaluation seems at first sight to be extremely static. However, in almost all current chess programs it is still the essential basis and when human chess players analyze games – mostly with the support of computers – then they, normally, think in these "pawn units" and translate other factors, like the mobility of the pieces as well into this scheme.

Now Turing had his algorithm but no corresponding machine – he had to go through the program on paper and had to evaluate the resulting positions by hand (ibid: 3)!

In this way games against human opponents were actually realized – Turing sat on the other side of the board and stubbornly followed (as in the Chinese room) the instructions of his algorithm without being allowed to incorporate his own understanding of chess. This must have been a frustrating experience, because the program revealed the typical shortcomings of the early days of computer chess, such as material overvaluation and planning inability.

During the game, Turing always tried to predict the next move made by his program, but this must have led to considerable disappointment as he regularly failed! After these sobering experiences with Turochamp, Turing put forward the thesis that it is impossible to develop a program that plays stronger than its developer. A claim that, as we know today, is definitely not correct.

The current chess programs have been playing for decades at a level that not only dwarfs the abilities of their developers but also those of the

strongest grandmasters. As early as 1996, a supercomputer from IBM called Deep Blue won against then world champion Garri Kasparov. Even though Kasparov was able to save the "honor of mankind" at the time, holding the upper hand in the overall competition over 6 games, the time had finally come one year later: in the rematch of 1997, the human world champion no longer lost just individual games, but also the overall competition.

In the meantime, it is considered pointless to let humans compete against computers in chess, which is why specially designed competitions for computer chess engines emerged.

During the 2010s, the free chess program Stockfish has dominated the Top Chess Engine Championship (TCEC), which is considered the unofficial championship in computer chess.

III.2 Some Theoretical Considerations on Chess Computing

III.2.1 Zermelo's Theorem (1912)

Besides practical programming trials, Zermelo's Theorem (1912) is one of the hallmarks that also theoretically characterize the challenges involved in chess computing. Zermelo's Theorem can be summarized as follows: in any board game, the first-player can force a win, or the second-player can force a win, or both players can force a draw. In other words, with perfect play the game will always end in the same result. However, it must satisfy the following conditions (Zermelo 1912: 501–504):

- The game board is finite,
- There are only two players,
- The game is of perfect information (nothing is hidden, unlike poker),
- The players alternate turns,
- There is no element of chance.

Zermelo's work anticipated the essentials of game theory, at least for games like chess, several decades before game theory came into being. Later, when von Neumann and Morgenstern presented their seminal work on game theory in 1944 (von Neumann/Morgenstern 1944), a corresponding generalization resulted, which is today known as the minmax algorithm, an algorithm which is suitable for determining the optimal strategy in all two-person zero-sum games with perfect knowledge.

In the general case, three possible results need not be assumed (as for chess). There can also be fewer – there are also games without a draw – or more so-called outcomes can be conceivable. But the decisive point is: if both sides play optimally, the result should always be the same.

From the point of view of game theory, chess is as simple as tic-tac-toe, and the fact that chess tournaments with or without computer participation still produce different results only proves that the optimal strategy has not been mastered yet.

From a practical point of view, Zermelo's theorem does not really help players. It is not even clear which of the three possible outcomes must result from optimal playing. Most practitioners assume the draw, some can also imagine an advantage for the white pieces (the player with this color plays first). But it is theoretically not impossible that having to start may even amount to a disadvantage. There are a number of well-known so-called forced move positions in chess where the side that has to move necessarily loses. While it is rather unlikely that the starting position should also be such a position, this is at least theoretically not impossible.

III.2.2 Further Theoretical Developments by Claude Shannon and Norbert Wiener

Another seminal paper is that of Claude Shannon (1950). It is interesting for a variety of reasons. The most notable of them in this context is that DeepMind's developers picked it up and cited it later in their own work (Silver 2017: 5), which will then bring us to the implementation details of AlphaZero.

Shannon, the founder of information theory, took up the minmax algorithm. However, it was also clear to him that the decision trees in chess would become much too extensive and too complex to manage (even) with computer help. Therefore, among other things, considerations were made to prune these decision trees, and not to consider all possible responses of an opponent, but to limit oneself to the most plausible responses. Norbert Wiener, the founder of cybernetics, again took up Shannon's work in his book The Human Use of Human Beings (Wiener 1954: 178), and obviously discussed it with him and others; the idea of a self-learning chess machine, with its possible potential but also its dangers, also came up already in the early 1950s. We will take up these aspects again in more detail in the conclud-

ing discussion. Against this background, let us now turn to implementation details of AlphaZero.

III.3 Implementation Details with Respect to AlphaZero

III.3.1 Overview

So what makes AlphaZero so innovative? First of all, AlphaZero breaks with the approach of traditional chess engines and acts completely differently. For example, AlphaZero has no domain-specific knowledge at all – with the exception of chess rules. As already described, early on from the time of Turochamp developers counted on evaluation functions which combined material aspects by summing up the value of pieces, positional aspects, for example the pawn structure, and security of the king, among other things. These evaluation functions had to be handcrafted and carefully weighted by human chess experts in traditional approaches. In addition, the aspects of material and position, which are static by nature, had to be supplemented by dynamic factors such as the mobility of the pieces and so-called tempi (a time advantage of three moves is valued approximately at one pawn unit).

With the traditional approaches, it took decades of hard work until it was finally possible to beat a human world champion (Gary Kasparov). Moreover, it should be taken into account that modern chess computers (before the neural network approaches) are also equipped with powerful opening libraries, in which centuries of human experience have been accumulated.

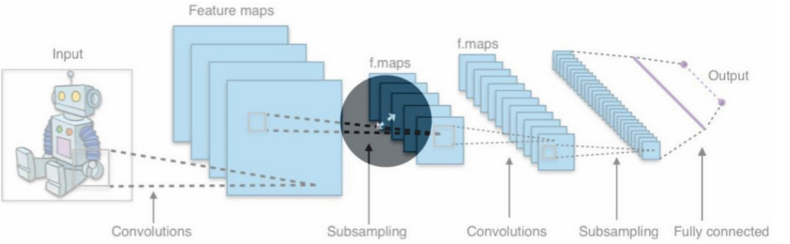
AlphaZero was able do without all that completely. Not even the most basic opening principles were implemented – no endgame table base was needed either. Instead, the algorithm relies on "learning from the scratch". The developers themselves coined the term tabula rasa reinforcement learning algorithm (Silver 2017: 2).

Briefly summarized: with the exception of the chess rules themselves, all knowledge of chess experts is dispensed with. Instead, methods are used that have generally proven themselves in modern AI systems (deep neural nets), such as convolutional neural networks, the gradient descent method and the MCTS approach – we will discuss these below.

III.3.2 Convolutional Neural Networks (CNN)

A CNN consists of filters (convolutional layers) and subsequent aggregations (by pooling layers), which are repeated alternately to be completed by one or more layers of fully connected normal neurons (Dense/Fully Connected

Figure 4: Convolutional Neural Networks ("Convolutional Neural Networks" 2019)



Layer). This leads to another strength of Convolutional Neural Networks, which is to condense information from a matrix input into a meaningful vector. ("Convolutional Neural Networks" 2019)

CNNs are used very successfully in the currently oft-discussed field of autonomous driving, for example, but also for the classification of objects in general. Some of the early layers of the network recognize simple features, such as differences in edges and brightness levels. Later on, these are combined to represent contours. Finally, in the later layers of the deep neural network this information is then combined to recognize objects as a whole.

Using CNNs for the game Go, too, was an obvious choice, in order to generate a one-dimensional fully connected layer from a two-dimensional input. The developers of deep mind first designed their general reinforcement algorithm for Go, and there the contours that separate white and black regions certainly play a special role. When transferring the concept to chess and shogi, it turned out to be useful to keep the principle in order to capture the two-dimensional board position as input.

The algorithm in question thus enables self-learning in the sense of reinforcement learning mentioned above. In the beginning, it was completely amateurish – but after 44 million games against itself, the system had obviously gained enough "experience" to "recognize" non-obvious resources in positions.

III.3.3 Adjustment by Gradient Descent

Learning in neural networks is based on the fact that the parameters of the network are continuously adjusted. A standard is the gradient descent method for training the weights connecting the neurons. Here, we are not dealing with supervised learning, but with reinforcement learning. In this sense, a training phase is never finished, but instead the system keeps improving itself, reinforcing everything that worked well and weakening everything that went wrong. In order to implement this reinforcement, an *externally defined objective function* is needed. In the case of AlphaZero, the developers have implemented a loss function for this purpose:

$$l = (z - v)^2 - \pi^T \log p + c||\theta||^2 \quad (1)$$

Silver et al. (2017: 3) adjust the neural network parameters in such a way that they minimize this loss function which contains on the one hand a mean-squared error $(z\text{-}v)^2$ but also so-called cross-entropy-losses, which is the second part in the equation above containing the logarithm. Most relevant is this second part because it contains with p a vector for the probabilities of the moves. A detailed description of the corresponding formula can be omitted here[1], but there is a close connection between quantities like information – entropy – probabilities and the logarithm function (Berkemer 2020: 242–244).

III.3.4 MCTS (Monte Carlo Tree Search)

The standard technique in traditional chess engines is to use so-called alpha-beta search to prune these decision trees. This was a significant enhancement to the minimax search algorithm that eliminates the need to search large portions of the game tree applying a branch-and-bound technique. Remarkably, it does this without any potential of overlooking a better move. If one already has found quite a good move and searches for alternatives, one refutation is enough to to exclude it from further calculations. (Chessprogramming, n.d.)

But even with alpha-beta pruning, there still remain very large decision trees, which ultimately cause many superfluous calculations in the chess engines. At this point, it proved worthwhile for the deep mind developers to take up Shannon's idea and, similar to human players, preferentially search

1 c is a parameter controlling the level of L2 weight regularisation (c.f. Silver 2017).

the part of the branches that is plausible counterplay. This is based on an update of probabilities. The main idea is that probable moves are analyzed more deeply.

Instead of an alpha-beta search with domain-specific enhancements, AlphaZero used a general-purpose Monte-Carlo tree search (MCTS) algorithm. (Silver 2017: 3). The developers of AlphaZero recur in this respect to an idea already proposed by Shannon to focus on such branches of the decision tree where the more plausible moves are to be found.

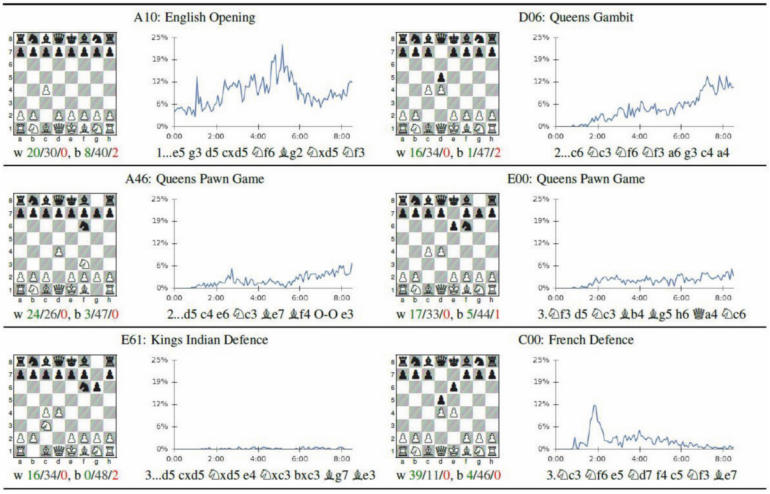
To give an idea of how effective this different approach is: AlphaZero had to search just 80 thousand positions per second, compared to 70 million for Stockfish (Silver 2017: 5). Why is this restriction to evaluate fewer positions an advantage for AlphaZero? Similar to human chess players, it is often not the quantity of calculations that matters, but that the "essential" calculations are carried out. The professional grandmaster is usually able to intuitively grasp in complicated positions what the plausible plans of both sides will be, which is an enormous help in a systematic evaluation. The amateur player, on the other hand, often uses a lot of computing capacity in the same situation to play through many possibilities rather unsystematically. Most individual calculations will be completely superfluous.

III.4 Illustration of AlphaZero's Strengths

III.4.1 Details on How an Opening is Self-learned by AlphaZero

As mentioned, AlphaZero had to learn everything itself and had no opening library, yet the algorithm seemed to succeed in gaining such a deep understanding of the specific problems of some openings that human chess analysts would later comment on this enthusiastically. The paper by Silver et al. (2017) records how often AlphaZero resorted to the 12 most common human chess openings. From their original figure we selected the first six.

Figure 5: Some selected chess openings and the frequency of their use (Silver 2017: 6)



We would like to draw attention to the French defense as an example (the entry at the bottom right). In an initial phase of the self-learning process, this opening is given a considerable frequency, but in the later course of time AlphaZero refrains from it.

It is also interesting to note that some of the other openings are increasingly "sorted out" by the algorithm over time. Let's take a closer look at an example of this, namely the French Defense just mentioned.

To illustrate how AlphaZero works, we refer to the publicly available video on Youtube, in which AlphaZero teaches another computer program, "Stockfish", a lesson in the French Defense. As already mentioned, Stockfish was considered at that time to be one of the strongest traditional chess programs. In one game against AlphaZero, where Stockfish (with the black pieces) used the French defense, the following position arose:

Figure 6: Potential answers to Stockfish capturing a bishop on d2 (own representation based on ChessNetwork, n.d., at minute 5:17)

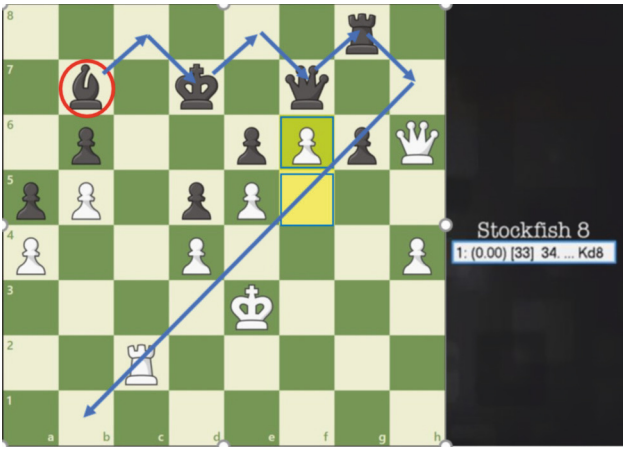
In this position Stockfish (Black) has just captured with his knight on d2. Which piece will AlphaZero use to recapture? In the same game some moves later the following position is reached:

Figure 7: Safe haven for the white king (found by AlphaZero close to center) (own representation based on ChessNetwork, n.d., at minute 5:31)

AlphaZero has recaptured with the king. Normally, chess players are taught to keep the king safe by castling. That is no longer possible here – but it doesn't matter. A few moves later the king is safe – well protected by his pawns (highlighted by circles) but closer to the center of the board than usual. An interesting aspect here is that in such positions it is humans who would react schematically and not the machine. Precisely because the human player is used to dividing the game explicitly into game phases (opening, middlegame and endgame), almost every player would almost automatically remember: "Caution! We are still in the opening and have not yet castled" – therefore, almost every player would recapture with the knight. AlphaZero – unencumbered by any opening knowledge – has the advantage of being more flexible here.

III.4.2 The Foresight of AlphaZero – Does AlphaZero Really Have a Planning Horizon?

Figure 8: About 15 moves later (The foresight of AlphaZero – did it "see" that there is no extra piece?) (own representation based on ChessNetwork, n.d., at minute 13:07)



AlphaZero with White lacks a complete piece – at least at a superficial examination. On the other hand, there is a protected passed pawn on f6, which is nice for White but normally not enough for compensation. After all, version 8 of Stockfish can even be commended for at least recognizing the basic danger. It "seems to feel" that there is an exceptional situation here. Stockfish's assessment gives an evaluation of exactly 0, which should be interpreted as White having "perfect compensation" for the missing material. An evaluation of zero means that Stockfish considers the position as a draw.

Unfortunately, such an assessment is still far too optimistic. The position is completely lost for Stockfish. Black's "additional" piece, the bishop (circled in red on b7), does not play any role. Blue arrows indicate an eventual path for activating this piece. But this would require far too much time to achieve.

Incidentally, the necessary time would be de facto even longer than indicated by the arrows. "Coincidentally" (or was this planned?), both king, queen and rook are themselves also still in the way of this path. In fact, White does not really have a piece less – only the additional passed pawn

counts. White will force an exchange of queens and/or rooks in a few more moves and Black can no longer prevent the opponent's passed pawn from becoming a decisive power. The "extra" bishop is not even good enough to sacrifice itself for this pawn.

AlphaZero seems to be able to "look very far ahead". Other chess programs do sacrifice pieces from time to time, but then they have a concrete compensation in mind.

There are also other examples with respect to the French defense opening reported from matches between AlphaZero and Stockfish where Black's white-squared bishop doesn't come into play – which are discussed on ChessNetwork. But is this really foresight on AlphaZero's part – does the program have a planning horizon?

In fact, this is difficult to answer, because the neural network is a black box. What the adjusted weights and other network parameters mean is not comprehensible. There are good reasons to doubt that any particular foresight was at work here. Highly efficient pattern recognition could alone sufficiently explain these cases.

So does the program really understand anything about chess? It may not even understand that it is playing chess at all!

For this purpose, let us consider the interpretation of the output layer. A common application of CNN are classification problems. In the output layer, as many neurons are provided as there are classes to be recognized. In general, the last layer often receives a so-called softmax activation. In the specific case of AlphaZero, this will always be done, because the output should be interpreted as probability of moves. The sum of all output neurons must then be exactly 1. In AlphaZero, all plausible moves are assigned probabilities. This, together with the iterative application, explains the farsightedness. Thus, even iteratively, several plausible countermoves are considered and not just the best countermove "at first sight". However, human tutoring is still required here – for impossible moves the probability must be "artificially" set to $p = 0$.

As self-contained as it seems to act at first sight, the program is therefore in fact not self-contained. One might even doubt that the program understood that the outputs should be probabilities. On the other hand, it may be unfair to say that it is merely pattern recognition. Good pattern recognition is quite closely related to chess understanding. This is demonstrated by studies comparing the performance of grandmasters and amateur players.

III.4.3 Some General Afterthoughts

Beyond AlphaZero, it might be of interest that it can also be stated quite concretely that, similar to chess, human performance, which was previously considered to be non-reproducible by artificial intelligence, is increasingly being addressed by AI engineers. One thinks here, for example, of composing music, painting pictures, speaking dialogues, answering questions, dreaming etc. For each of these examples, there is a branch of artificial intelligence research that can already be reproduced with considerable success. Creativity, for example, is reproducible in music through a music-producing neural network (cf. the album "I am AI," (see Nelson 2018; Amper AI 2017). Similarly, there are already extensive efforts to measure human emotions and to replicate them or to have an AI react adequately to these emotions (Deml 2019: 86; Bartl 2019: 90).

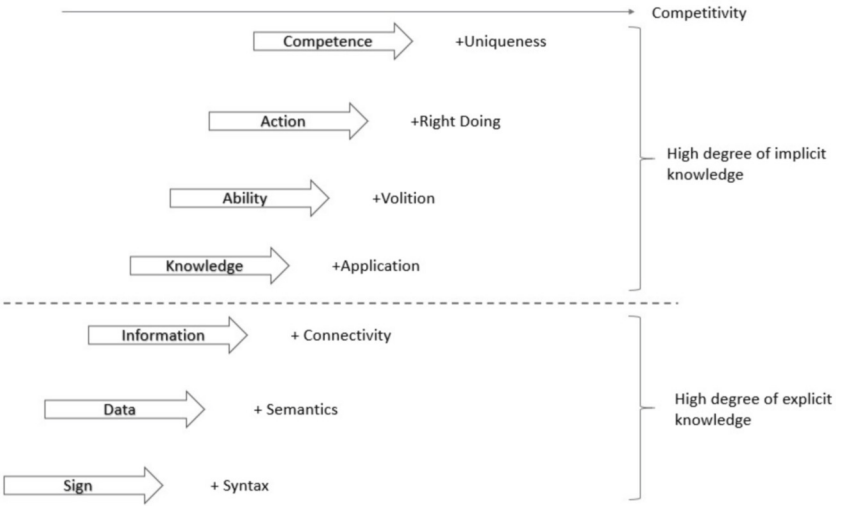
Collaboration is, for example, already partly reproducible through computer collaboration, edge computing, e.g. in the form of the control of decentralized data processing, which controls the processing, utilization or priority in a network of collaborative computers. Google is working on Google Duplex, an assistant that is indistinguishable from a human assistant in the sense of the Turing Test or the Chinese Room (Walch 2020). IBM Watson beat its human opponents in Jeopardy back in 2011 with its Question Answering System. Currently, the AI system GPT-3 is being discussed (not to forget the even more elaborated followers like ChatGPT), which uses an autocomplete function as question answering based on, among other things, the entire Wikipedia entries and has an answer to almost every knowledge question. Do these systems really only go as far as level 3 of the knowledge pyramid mentioned above in Figure 3?

III.5 Back to the Knowledge Pyramid – What Our Insights into AlphaZero Involve

Against the background of the analyses and considerations carried out thus far, let us return to the knowledge pyramid which is now illustrated in figure 9.

Viewed in the context of the knowledge pyramid, it becomes clear how limited artificial intelligence is. AlphaZero can only accomplish its amazing performance because it has precisely specified that this should take place within the framework of the application reference chess and chess rules.

Figure 9: Complete knowledge pyramid (North 2021: 37)



But these rules come from humans. Does AlphaZero have a will? Apparently not, because as Beck (2017: 11) notes:

> Computers may beat us at chess or Go, that is not surprising, neither creative nor worrisome. I would only be seriously concerned if a computer started making mistakes and subsequently proclaimed: "Chess? Oh no, don't feel like it anymore, it's boring ….".

Also, it is hardly to be expected that AlphaZero realizes when playing against itself that it always has to end in a draw – provided that nobody makes a mistake. By the way, it was implicitly assumed just now that the third possibility (draw) of Zermelo's theorem might be the most plausible. We will return to the clarification of this question shortly.

Does AlphaZero have an understanding of chess? If AlphaZero could "recognize" that chess can be treated in principle exactly like tic-tac-toe, it could define concepts like two-person/zero-sum games, make generalizations for Tic-Tac-Toe, Chess, Go, etc. and independently develop something like a Theorem of Zermelo on its own. Something like this would indeed be impressive, but AlphaZero doesn't even have to understand that it is playing chess.

In a nutshell, these limitations mean that some of Norbert Wiener's worries about the advent of the chess machine are (still) unfounded. He was thinking about a machine that could also take over all governmental affairs and certainly sees a threat in such a "machine à gouverner". In this regard, he also quoted an informative book review by Pere Dominique Dubarle in *Le Monde* on Wiener's book "*Cybernetics*". Dubarle comes to the following assessment with reference to Hobbes' Leviathan:

> "[I]n comparison, Hobbes' Leviathan was nothing but a pleasant joke." (Dubarle 1948 cited by Wiener 1950: 180)

As long as the objective function is externally specified, as in AlphaZero, the system will never do anything other than play chess as well as possible and would never arrive at the idea to independently create references to other games or strategically similar situations. It might be important to note that this does not involve transfer learning by using parts of the neural network elsewhere. It is only the general reinforcement principle that is applied once more. Therefore, the following cinematic scenario would also be impossible, even if it is still so appealing. The theme of the movie War Games[2] is the danger that a self-learning software becomes first independent and then virtually unstoppable by humans. Specifically, a boy accidentally triggers a "thermonuclear war" simulation on the supercomputer and things take their fateful course. Unfortunately, the software is so intelligent that it knows how to resist when humans try to interfere. The machine's rationale must necessarily insist on "Mutual Assured Destruction", because anything else would undermine a credible deterrence – there is no place for humans with their emotions.

What is particularly interesting is how the boy, together with the AI researcher responsible for its development, manages to prevent the catastrophe in the end. The machine won't simply switch off. But it does agree to play Tic Tac Toe against itself – like chess and shogi, a 2-person constant-sum game. In the process, the machine learns that it is impossible to win this game. An endless series of draws shows the machine that it is pointless to try any further.

What is the real punchline, then? It is that the software recognizes the analogy to nuclear war itself. It plays through all variants of the "game" ther-

2 For details on the movie War Games from 1983, consider: https://www.themoviedb.org/movie/860-wargames.

monuclear war against itself. As with tic tac toe, the frustrating result every time: WINNER NONE. This leads the AI to the independent conclusion: "it is a strange game" and "the only winning move is not to play."

IV. Discussion

IV.1 What Does the Lack of Understanding Involve?

As we have identified beforehand, it is precisely this lack of understanding that characterizes artificial intelligence learning algorithms such as AlphaZero, despite the impressive results they deliver. Precisely this also repeatedly opens up the question of disruptive attacks on an AI by humans or other AI systems (so-called "adversarial attacks" or "adversarial examples"). One example is the rather famous story in AI research circles about a neural network model that was trained to distinguish between wolves and huskies. The model learned to identify the distinction successfully, achieving high accuracy when given images that weren't used for its training. However, it soon became apparent that something was going wrong – some very clear images were being misclassified. When the researchers looked into why the neural network was making such gross mistakes, researchers figured out that the model had learned to classify an image based on whether there was snow in it – all images of wolves used in the training had snow in the background, while the ones of huskies did not. Unsurprisingly, the model was failing. In other words, what humans understand immediately – that it is wolves that count and not the context in which wolves occur – was precisely that which was not understood by the model, and this made clear that the model only searched for the best way to discriminate between two different labels provided by humans to images. There was no understanding. As Goodfellow/Shlens/Szegedy (2015: 9) pointed out with respect to adversarial examples:

> The existence of adversarial examples suggests that being able to explain the training data or even being able to correctly label the test data does not imply that our models truly understand the tasks we have asked them to perform.

This also entails that the AI itself is not prepared to detect ex ante which of its own weak points will be targeted and the AI can thus easily be misused

or misled into making incorrect decisions. Here again, technical solutions must be found and anticipated by the sense-making human to make an AI robust against such attacks. One way to mitigate such challenges is to carefully select the training data set.

In addition to such weaknesses of AI, there is the realization that the behavior of an AI depends massively on the data set that was used for training the net, or, in the case of reinforcement learning, every interaction leaves its (unpredictable) repercussions on the AI, which in turn depends on the given rules of the AI. As such, it can be observed that an AI incorporates e.g. discrimination into its own behavior without reflection (cf. e.g. Beck 2019: 92–94). This was already evident with Alexa, which, trained on unethical data, famously recommended to its users that they kill their mother-in-law. In other words, a learning algorithm will optimize what it is trained to optimize but leaves out what – perhaps varying – circumstances might come along with the elements focused on in its solution. Furthermore, it will ignore the potential consequences of that. It does not change its optimization rules that determine the direction of learning within the rules of the game set out by humans, it does not contradict or point out that there might be a different perspective involved and it does not highlight the challenge of reconciling diverse perspectives. All this is still the reserve of humans and not very surprisingly an essential part of personality and education: basically, this is a topic that already occupied the pioneers of AI. Early on, Wiener was concerned with the question of what could happen if the chess machine should "grow up". In the case of AlphaZero this does not happen: it will not occur to AlphaZero to introduce new rules, but theoretically there could be manifold variations. Why not play atomic chess, which humans invented as soon as normal chess with always the same openings became too boring for them. However, do machines know boredom at all? Or do machines know such a thing as curiosity? It has obviously been possible to develop admirable algorithms that are very powerful at linking information from different areas and that perform excellently when it comes to answering questions. But formulating questions of one's own accord is probably still the domain of humans. Even young children are inquisitive. You don't have to train them to develop curiosity. Here we have again the component of the will. The child pesters adults with one question after another; it is not satisfied with an insufficient, half-baked answer and keeps on asking: "why this?", "why that?"

It is clearly evident here: there is a lack of independent will for AlphaZero to do something just for the fun of it. Here there is a lack of real understanding that what AlphaZero is trained for in the first place is a "game" of chess, which is supposed to be "fun". AlphaZero doesn't even need to understand that it provides probabilities. Humans had to help out here with a softmax activation. There can be no question of acting and competence on higher levels of the knowledge pyramid. All in all, according to the analyses made here, even the newest AI applications therefore still remain far behind (at least educated) humans even though they might seem to be at the same level (which is especially visible in the example of ChatGPT).

As we elaborated on above, artificial intelligence and its capabilities are the result of the context labelled and established by a human mind. The result that comes to the fore when the algorithm works might then seem to be at the upper end of the knowledge pyramid, but, in reality, it is only performing without its own understanding what the human mind has included in the context that the artificial intelligence has been trained with. What happens, then, when reinforcement algorithms like AlphaZero are trained on subjects that involve volition, right or wrong action and in the end uniqueness without the humans setting adequate labels?

This seems to be difficult in that even human experts do not fully understand in all its consequences what the AI incorporates when learning. We humans resort very consciously to different frames in acting and the more competencies we have, the more unique is the result. This is the exact difference in the case of AI. An AI always acts according to the context that was set beforehand and based on the same frame set defined by the rules. It does not decide whether it wants to act under such conditions. To recur to the wolf example: humans must first anticipate that the context difference could be used at all to classify before they are able to mitigate the misclassification in the training data by providing images with snow but without wolves and images with wolves but without snow. They decide that it makes no sense to discriminate based on whether there is sun or snow. Or, for an even more sensitive question: when we distinguish between good or bad actions, can we really employ algorithms of artificial intelligence in assisting us in such a task? The analysis of Davin (2020) demonstrates how difficult such a task is. The author demonstrates that the nearest neighbor method allowed for identifying separate clusters of "good" or "bad" people in fairy tales. However, it was not able to differentiate between them as good or bad people. One reason put forward by the author is that categories

like good or bad rely on very high degrees of implicit knowledge, which is precisely the competence of humans. However, when we take into consideration that values nearly always influence human actions when seen in a certain context – even driving a car takes on a moral dimension when it is analyzed in the context of its effects on the climate – then it seems to be very naïve to train artificial intelligences within a certain explicit (or even unknown) context but then extend it to other contexts. It seems that humans much too easily assume that they really know what they are training the AI for and then are "surprised" by adversarial attacks that relate to the higher levels of the knowledge pyramid whose context and involvement in the situation have not been adequately reflected. As such, it seems to us that it is not necessarily progress that is currently taking place, but rather disorientation.

IV.2 Societal Consequences and Responsibility

In fact, the discussion above directly leads to a further point of utmost importance. Let us for a moment think to ourselves what this involves when in the most different areas such – probably in most cases only half-reflected – experiments with artificial intelligence take place.

Consider what happens when AI begins to interact with humans in terms of the meaning and implications of higher levels of the knowledge pyramid, although the understanding of unanticipated side-effects is lacking among the human experts responsible for designing the algorithms. Especially in the case of unsupervised learning in datasets that can no longer be fully understood by any human, the implications are likely to be enormous.

It is easy to find current cases of application. What happens with an artificial intelligence that is designed to maximize the time children spend on social networks – without taking into account that this might not be good for the children themselves? What happens when an artificial intelligence is trained to maximize silo communication instead of cooperation, to divide societies and to corrupt elections with respect to certain candidates? What are the side effects of that? What happens when an artificial intelligence is trained – in an industrial context – to maximize profit but not to maximize the adequacy of the products with respect to the customer's safety, needs or health? Anyone familiar with today's business world is also aware that such things could happen. And even more alarming is that it might be nearly impossible to determine this after the fact – whether an artificial

intelligence causes critical adversarial attacks could be – given the existing data masses – just as easily be a result of chance as a deliberately calculated effect of the artificial intelligence. Therefore, it is impossible to judge why an artificial intelligence – which cannot judge itself and which acts according to certain criteria that have been acquired in a mixture of human design and unknown data masses – acts in a certain way. And how should we judge people who become influenced by such information and then act accordingly? Given that we are already subject to such experiments, this reminds us to stay very cautious in our judgements. In particular, judging people or positions as evil or as enemies no longer becomes a viable option. We can only point to possible consequences of one's own actions.

Moreover, an artificial intelligence acts based on data as given or optimized – humans, however, evolve freely. Without this dimension of evolving freely, artificial intelligences could judge and therefore discriminate unthinkingly based on the status quo, whereas e.g. social work would start to reach a different stage.

V. Summary and Conclusion

If we summarize the analyses carried out above, we might state that so far especially the areas of explicit knowledge (supervised learning) and explicit specifications (reinforcement learning, unsupervised learning) have been convincingly addressed by artificial intelligence. In fact, many routine activities that are currently still performed by humans can be performed faster and more efficiently by artificial intelligence in the future. Here, too, the economic potential of artificial intelligence is expected to be realized in the near future. However, for the time being, educated people cover numerous competencies that artificial intelligence cannot (yet?) address and for which, based on our current state of knowledge, one can only speculate as to whether they will ever be open to artificial intelligence.

Even in those areas in which artificial intelligence has high potential, this is tied in particular to the availability of data. A challenge that remains is therefore the existence and selection of training data (Bartschat et al. 2019: 1401). Whether the training data represent a comprehensive problem domain for the posed problem depends on whether later results are also correctly classified or processed. Precisely this comprehensiveness is usually hard to ensure for rare or complex events (Bartschat et al. 2019: 1401), whereas what

is rare and complex for AI can also be a simple manual task from a human perspective.

At the same time, an AI usually processes single data sets in a certain way and does not draw any novel conclusions from the surrounding context in such a way that the perspective is changed, e.g. a phenomenon has to be looked at more closely or differently because an observation otherwise turns out inadequate (Buder 2020: 17–22). Why it might be possible to train the AI with every new classification by taking recourse to human feedback, this will still not solve the problem of situation-specific decision-making as far as the meaning of situations are indeed dependent on a specific context. This might be addressed by including even more data and variables. At the same time, however, an explosion of hidden layers makes it even more difficult to understand how an artificial intelligence arrives at its results (Schmitz et al. 2019: 777).

From the perspective of university teachers (scholars), this results in a threefold task. First, future generations must be trained more strongly in those competencies that make a difference, i.e. knowledge, skills, action and competence. Second, it is necessary to sharpen for the focus on current application areas of AI and not to overinterpret them, in order to maintain a clear view of the prerequisites for the successful use of AI, which could turn out to be much more limited than is often suggested today. Finally, the dependency on the training data set and the rules determined beforehand by humans (in reinforcement learning) is central and must be analyzed in particular detail.

For example, in chess and Go the underlying objective function may be uncontroversial. Providing a general reinforcement algorithm here was extremely successful, as could be seen, and in this respect such an application is unproblematic – but with thermonuclear war and other "war games" as well as strategic management decisions, the situation looks completely different. The tendency to assume that people who are good at chess will be more successful than others in such strategic situations has always been questionable, because from a game-theoretical point of view, two-person-zero-sum games are neither the common case nor the most important ones. But to think that an AI based on the general reinforcement approach should be granted decision-making competence in these critical areas just because it was more successful in chess and Go could have literally catastrophic consequences.

A word of warning is therefore in order against the overinterpretation of the term "general" in the concept of the general reinforcement algorithm. Natasha Regan, co-author of the book Game Changer, claims in an interview that the technique could be used for any complicated system where one needs to navigate a path through lots of possibilities.

Silver et. al. (2017) have successfully transferred the basic concept of the general reinforcement algorithm from one zero-sum game to another. Essentially, this only costs time for new training and this does not seem to be too essential, if then, as in the example of chess, four hours are sufficient to dwarf centuries of human knowledge.[3]

But these games do not represent just "*any complicated system* where you need to find navigate a path". Making strategic decisions in management or, as Shannon had in mind, in simplified military operations is a completely different issue. First of all, these are not zero-sum games. The rules are not so clearly specified. Most importantly, it will not be so easy to gain sufficient experience by "playing against oneself". In chess, go and shogi, all possible countermoves of the opponents are fixed, the consequences of one's own decisions can be clearly calculated. If one wanted to use a general reinforcement approach in management or war scenarios, a simulation model would have to be implemented in addition to predict these consequences, and since simulations can only ever represent a simplified image of reality, an inherent source of error is built in.

Let us conclude with the reflections of Norbert Wiener. On the one hand, Wiener would certainly be thrilled that it was possible to ultimately implement his and Shannon's ideas on self-learning machines. As suggested by Shannon, a focus on plausible (probable) moves proved to be superior to other approaches. It is possible, however, that Wiener would have been disappointed at how shockingly easy it was to implement machine learning. A tabula rasa approach certainly would not have appealed to him. He wanted the machine to keep improving, but he had firmly counted on the premise that in order to learn, it would have to compete against passable or, even better, very good human players. Here, Wiener was completely wrong in his assessment, similar to Turing, who was sure that the machines would never surpass the level of their developers.

3 However, if one also considers, in addition to the time, the energy consumption of such training, the situation looks different (cf. Marischka 2020).

References

Amper AI (2017): *I am AI*. February 2, 2020. URL: https://futurism.com/the-worlds-first-album-composed-and-produced-by-an-ai-has-been-unveiled.

Bartschat, Andreas, Stephan Allgeier, Sebastian Bohn et al. (2019): "Digitale Bildverarbeitung und Tiefe Neuronale Netze in der Augenheilkunde – aktuelle Trends." In: *Klinische Monatsblätter für Augenheilkunde* 236 (2019), pp. 1399–1406.

Bartl, Michael (2019): "The rise of Emotion AI." In: *Handbuch künstliche Intelligenz*. away Medias GmbH, Bonn 2019, pp. 87–90.

Bauermeister, Karsten (n.d.): *Schachcomputergeschichte*. July 6, 2021. URL: http://www.schachcomputer.at/gesch.htm.

Beck, Henning (2017): *Irren ist nützlich. Warum die Schwächen des Gehirns unsere Stärken sind*. Hanser: München.

Beck, Susanne (2019): "Wie diskriminierend ist künstliche Intelligenz?" In: *Handbuch künstliche Intelligenz*. away Medias GmbH, Bonn, pp. 91–95.

Berkemer, R. (2020): "Effiziente Nutzung von Information als Rohstoff im Spannungsfeld von Kommerzialisierung und Kollaboration." In: *Digitale Bildung und Künstliche Intelligenz in Deutschland*. Springer: Wiesbaden, pp. 241–255.

Buder, Jürgen (2020): "Wieviel Mensch steckt in der Maschine?" In: *Information – Wissenschaft & Praxis* 71.1, pp. 17–22.

Bruns, Beate, and Cäcilie Kowald (2019): "Wieviel KI steckt in Bots?" In: *Handbuch künstliche Intelligenz*. away Medias GmbH, Bonn, pp. 77–82.

Buxmann, Peter, and Holger Schmidt (2019): *Künstliche Intelligenz – Mit Algorithmen zum wirtschaftlichen Erfolg*. Berlin: Springer Gabler.

Chessgames Services LLC (n.d.): *AlphaZero – Stockfish (2017)*, London ENG, Dec-04. July 6, 2021. URL: https://www.chessgames.com/perl/chessgame?gid=1899422.

ChessNetwork (n.d.): *AlphaZero teaches Stockfish a lesson in the French Defense*. July 6, 2021. URL: https://www.youtube.com/watch?v=4ebzevCLGbQ.

Chessprogramming (n.d.): *AlphaBeta*. July 8, 2021. URL: https://www.chessprogramming.org/Alpha-Beta.

Convolutional neural networks – Aufbau Funktion und Anwendungsgebiete (2019). August 16, 2020. URL: https://jaai.de/convolutional-neural-networks-cnn-aufbau-funktion-und-anwendungsgebiete-1691/.

Deml, Barbara (2019): "KI für die Arbeitswelten der Zukunft." In: *Handbuch künstliche Intelligenz*. away Medias GmbH, Bonn, pp. 83–86.

Dubarle, Dominique (1948): Book review on "Cybernetics" In: *Le Monde*, December 28, 1948.

Goodfellow, Ian J., Jonathan Shlens, and Christian Szegedy (2015): *Explaining and harnessing adversarial Examples*. International Conference on Learning Representations.

Gunning, David, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Ghuong-Zhang Yang (2019): "XAI-Explainable artificial intelligence." In: *Science Robotics* 4.37, eaay7120.

Hayek, Friedrich August von (1979). *The Counter-Revolution of Science. Studies on the Abuse of Reason*, 2. Aufl. Indianapolis: Liberty Fund.

Jerems, Stefanie (2018): "SYD81 Neuronale Netze 1." AKAD University: Stuttgart.

Kendall, Graham (2001): "G5AIAI – Introduction to Artificial Intelligence". July 6, 2021. URL: http://www.cs.nott.ac.uk/ pszgxk/courses/g5aiai/002 history/eliza.htm.

Kriesel, David (2007): "A Brief Introduction to Neural Networks" available at http://www.dkriesel.com. September 22, 2021. URL: https://www.dkriesel.com/_media/science/neuronalenetze-en-zeta2-1col-dkrieselcom.pdf.

Marischka, Christoph (2020): *Cyber Valley – Unfall des Wissens. Künstliche Intelligenz und ihre Produktionsbedingungen – Am Beispiel Tübingen*, Köln: Papy-Rossa.

Nelson Jr., Keith (2018): "Taryn Southern's new album is produced entirely by AI." August 16, 2020. URL: https://www.digitaltrends.com/music/artificial-intelligence-taryn-southern-album-interview/.

Nguyen Cong Luong, Thai Hoang Dinh, Gong Shimin, Niyato Dusit, Wang Ping, Ying-Chang Liang, and In-Kim Dong (8. 10. 2018): "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". https://arxiv.org/pdf/1810.07862v1.pdf.

North, Klaus (2021): *Wissensorientierte Unternehmensführung: Wissensmanagement im digitalen Wandel*. Wiesbaden: Springer Gabler.

Sadler, Matthew, and Natasha Regan (2019): *Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI*. Alkmaar: New in Chess.

Samek, Wojciech, and Klaus-Robert Müller (2019): "Towards explainable artificial intelligence." In: *Explainable AI: interpreting, explaining and visualizing deep learning*. Springer, Cham, pp. 5–22.

Schmitz, Rüdiger, René Werner, and Thomas Rösch (2019): "Der Einfluss der Algorithmen. Künstliche Intelligenz in der Endoskopie: Neuronale Netze und maschinelles Sehen – Techniken und Perspektiven." In: *Zeitschrift für Gastroenterologie* 57, pp. 767–780.

Seifert, Inessa, Matthias Bürger, Leo Wangler, Stephanie Christmann-Budian, Marieke Rohde, Peter Gabriel, and Guido Zinke (2018): "Potentiale künstlicher Intelligenz im produzierenden Gewerbe in Deutschland." February 29, 2020. (https://www.bmwi.de/Redaktion/DE/Publikationen/Studien/potenziale-kuenstlichen-intelligenz-im-produzierenden-gewerbe-in-deutschland.pdf?__blob=publicationFile&v=8.).

Shannon, Claude, E. (1950): "Programming a Computer for Playing Chess." In: *Philosophical Magazine*, Ser, 7.41, pp. 256–275.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, …, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis (2017): "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." arXiv preprint arXiv:1712.01815.

Turing, Alan (1950): "Computing Machinery and Intelligence." In: *Mind*, pp. 433–460.

Von Neumann, John, and Oscar Morgenstern (1944): *Theory of games and economic behavior (commemorative edition)*. New Jersey: Princeton university press.

Walch, Kathleen (2020): "Rethinking weak vs. Strong AI." August 10, 2020. URL: https://www.forbes.com/sites/cognitiveworld/2019/10/04/rethinking-weak-vs-strong-ai/#499b55006da3.

Wiener, Norbert (1954): *The human use of human beings*. London: Eyre and Spottiswoode.

Zermelo, Ernst (1912): "An application of set theory to the theory of chess-playing." In: *Proc. 5th Int. Congress of Mathematics*, Cambridge.