Peter Klimczak,
Christer Petersen (eds.)

AI

Limits and Prospects of Artificial Intelligence

[transcript] AI Critique | KI-Kritik

Peter Klimczak, Christer Petersen (eds.)
AI – Limits and Prospects of Artificial Intelligence

KI-Kritik / AI Critique | Volume 4

Editorial

Since Kant, critique has been defined as the effort to examine the way things work with respect to the underlying conditions of their possibility; in addition, since Foucault it references a thinking about "the art of not being governed like that and at that cost." In this spirit, **KI-Kritik / AI Critique** publishes recent explorations of the (historical) developments of machine learning and artificial intelligence as significant agencies of our technological times, drawing on contributions from within cultural and media studies as well as other social sciences.
The series is edited by Anna Tuschling, Andreas Sudmann and Bernhard J. Dotzler.

Peter Klimczak (Prof. Dr. Dr.) is adjunct professor at Brandenburgische Technische Universität. He conducts research on digital/social media, cognitive systems, and the use of artificial languages in media and cultural studies.
Christer Petersen (Prof. Dr.) holds the Chair of Applied Media Studies at Brandenburgische Technische Universität and works in the fields of media and cultural semiotics, materiality and technicity of media.

Peter Klimczak, Christer Petersen (eds.)

AI – Limits and Prospects of Artificial Intelligence

[transcript]

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available in the Internet at http://dnb.d-nb.de

First published in 2023 by transcript Verlag, Bielefeld

Cover layout: Maria Arndt, Bielefeld
Typeset: Sebastian M. Schlerka, Bielefeld
Printed by: Majuskel Medienproduktion GmbH, Wetzlar
https://doi.org/10.14361/9783839457320
Print-ISBN: 978-3-8376-5732-6
PDF-ISBN: 978-3-8394-5732-0
ISSN of series: 2698-7546
eISSN of series: 2703-0555

Printed on permanent acid-free text paper.

Contents

Preface

Just today, as we are making the final corrections to the present volume, an online article rolls into our newsfeeds. There we read that in June of this year, Google programmer Blake Lemoine went public. He was concerned with the 'Language Model for Dialogue Applications' (LaMDA), an intelligent speech bot from Google specialized in dialogues. "As first reported in the Washington Post, programmer Lemoine, after his conversations with LaMDA, no longer believes that this machine is merely a tool. He demands that LaMDA be treated as a person. He believes LaMDA has gained consciousness." – "Can this be?" asks the author of the article. – "Short answer: most likely not."[1]

And yet, no other topic in recent years has triggered such a storm of enthusiasm and simultaneously such a wave of uncertainty: artificial intelligence. Intelligent software will automate work, understand images and text and make medical diagnoses. It enables cars to drive independently and supports scientific work. On the basis of current innovations, a global revolution in business and industry can be predicted, which is being driven along ever further by the networking of 'smart machines'. At the same time, artificial intelligence promises to be able to analyze the masses of data produced in the course of digitalisation for patterns and to make automated decisions based on these, which will not only rival the quality of human analysis, but will be far superior to human work in terms of efficiency.

The current central paradigm of AI is machine learning, whereby deep learning seems to represent the most promising technological approach: software is no longer programmed for its specific tasks but is trained on

1 "Künstliche Intelligenz, […] was kommt da auf uns zu?" October 11, 2002. URL: https://krautreporter.de/4601-kunstliche-intelligenz-so-verstandlich-wie-moglich-erklart?utm_campaign=pocket-visitor, translated from German.

large amounts of data – on Big Data. Here, however, the limits of the new technology begin to become apparent, as data volumes are not always available in the desired quality and quantity. It also requires basic or retrospective labelling by humans as well as the processing of data by means of symbolic AI procedures, therefore by supporting Small Data approaches. It is not uncommon for social discrimination to be inherent in the data sets, to which human preselection, categorisation and corresponding annotation make a decisive contribution. Inevitably, this leads to the integration of cognitive biases into a given AI: prejudices embedded in society are reproduced and to some extent further reinforced. Moreover, if machine decisions are made on the basis of deep learning systems, the decisions made by the machine are also non-transparent for humans. These are quite rightly characterised as black box procedures, which are accompanied by considerable legal problems. And finally, but importantly, the fundamental question arises as to whether people can trust systems with which they are supposed to collaborate, when not even the systems' very creators can reconstruct and interpret the decisions of their technical creatures.

The volume is therefore intended to explicitly show the limits of AI, to describe the necessary conditions for the functionality of AI, to reveal its attendant technical and social problems, and to present some existing and potential solutions. At the same time, however, we also want to describe the societal and attending economic hopes and fears, utopias and distopias that are associated with the current and future development of AI. This has led us to gather in this volume authors and contributions from a variety of scientific disciplines, meaning that computer scientists, engineers, mathematicians, as well as media studies experts, social scientists and economists have their say on an equal footing. And lastly, but foremost, we thank our English-language editor, Richard Slipp, who has repeatedly defied DeepL and all other artificially intelligent translators and pointed out their limitations, thus making the volume in its present form possible.

Peter Klimczak and Christer Petersen
October 2022

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

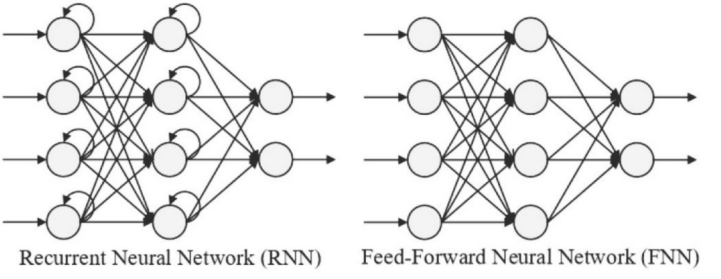
The starting point of an artificial neural network is a schematic reproduction of a biological model, a neuron, which is artificially reproduced in the case

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)



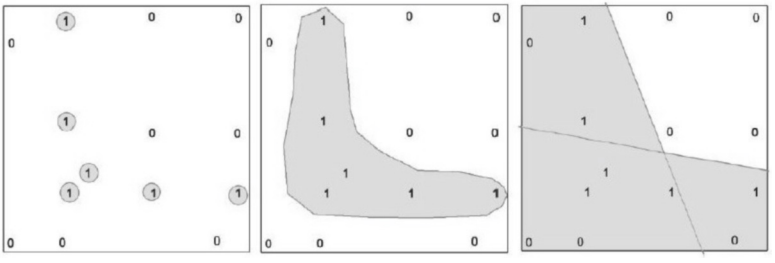
The layers in between are called *hidden layers* and modern neural networks, as they are often used in the field of pattern recognition, typically have a huge number of such hidden layers. These are commonly also called Deep Neural Networks (DNN) insofar as they exhibit a sufficiently high number of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

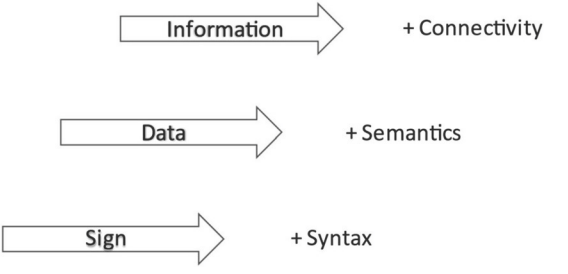
It seems to be significant here that Eliza gets along without semantics, i.e. did not need to understand the meaning of the terms uttered. A knowledge base was sufficient that, for example, connects father, mother and son etc. with family, i.e. that knows the correct use of syntax. Nevertheless, it might 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.

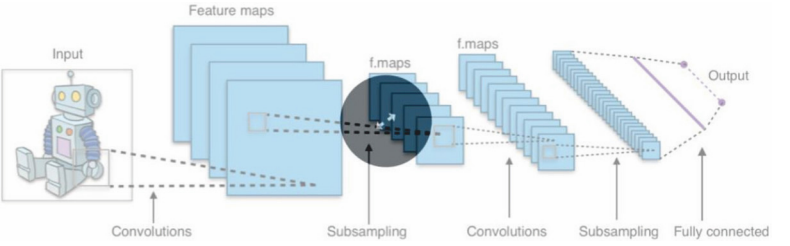
AlphaZero was able do without all that completely. Not even the most basic opening principles were implemented – no endgame table base was needed either. Instead, the algorithm relies on "learning from the scratch". The developers themselves coined the term tabula rasa reinforcement learning algorithm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

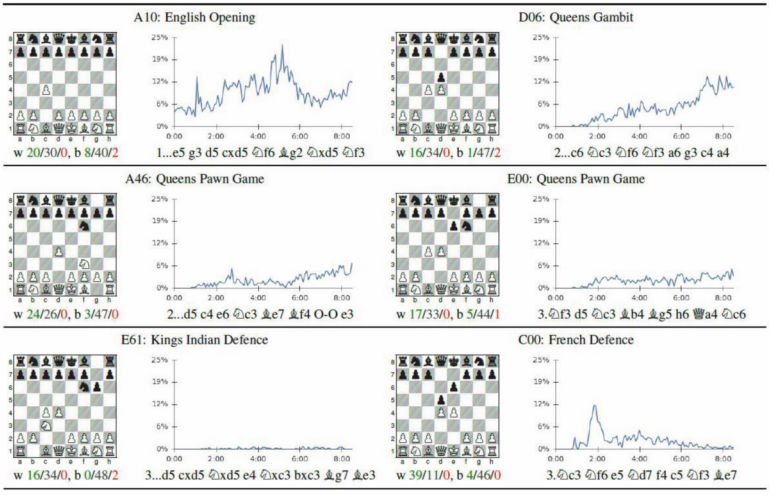
To give an idea of how effective this different approach is: AlphaZero had to search just 80 thousand positions per second, compared to 70 million for Stockfish (Silver 2017: 5). Why is this restriction to evaluate fewer positions an advantage for AlphaZero? Similar to human chess players, it is often 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.

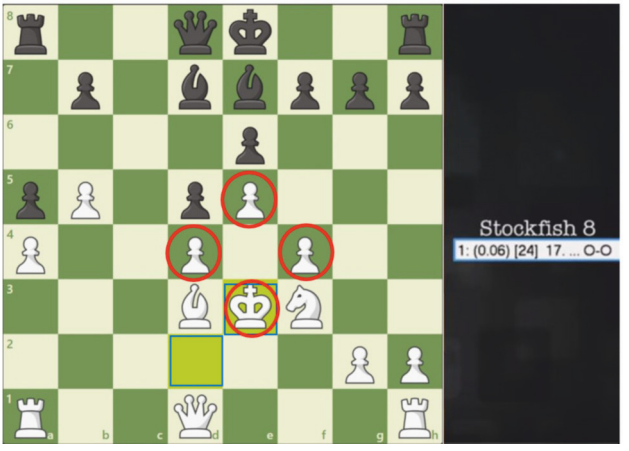
It is also interesting to note that some of the other openings are increasingly "sorted out" by the algorithm over time. Let's take a closer look at an example of this, namely the French Defense just mentioned.

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:

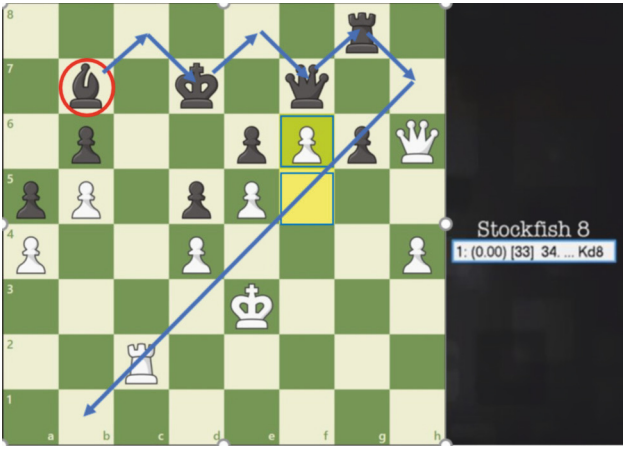
Figure 7: Safe haven for the white king (found by AlphaZero close to center) (own representation based on ChessNetwork, n.d., at minute 5:31)



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