

HOCHSCHULE DARMSTADT

AUSARBEITUNG

**Machine learning im Bereich
Strategiespiele ohne
unberechenbare Zufallsfaktoren**

Lucas Stumm

Matrikelnummer: 764915

Dozentin: Katja Lenz

March 22, 2022

Contents

1 Einleitung	1
2 Randomness	1
2.1 Input randomness	1
2.2 Output randomness	2
2.3 unberechenbare Zufallsfaktoren	2
3 Lernverfahren von machine learning	4
3.1 Decision Tree Learning	4
3.2 Monte Carlo Tree Search (MCTS)	4
4 Probleme und Lösungsansätze	5
4.1 Class Imbalance	6
4.2 Overfitting/ Underfitting	6
5 Einfluss von machine learning Software	7
5.1 Alphastar	8
5.2 AlphaGo Zero	10
5.3 Open AI Five	11
6 Ausblick für die Zukunft	13
7 Anhang	15
8 Eidesstattliche Erklärung	18

1 Einleitung

Die Digitalisierung ist ein großer Bestandteil der Gesellschaft im 21. Jahrhundert. Machine learning spielen eine große Rolle dabei. Insbesondere betrifft dies den Bereich der Spiele.

Daher ist die Forschungsfrage in dieser Arbeit, inwieweit machine learning Strategiespiele obsolet machen und welche weitreichende Konsequenz dies mit sich zieht. Es wird erwartet, dass die Anzahl der Versuche von Betrug sich steigern wird im Bereich der Strategiespiele mit machine learning. Zudem ist es anzunehmen, dass Spieler, die gegen die KI spielen, sich verbessern werden.

Zuerst wird darauf eingegangen, welche Glücksfaktoren es gibt und warum diese störend sind für die Entwicklung von machine learning. Anschließend werden die Verfahren Monte Carlo Tree Search (MCTS) und Decision Tree Learning erklärt. Danach werden die Probleme von Class Imbalance und Overfitting/Underfitting eingegangen. Darauf werden auf die machine learning Software Alphastar, AlphaGo und Open AI und ihre Auswirkungen auf die jeweiligen Spiele eingegangen. Am Ende werden auf möglichen Auswirkungen für die Zukunft erörtert.

2 Randomness

In diesem Kapitel wird darauf eingegangen, was **Input** und **Output randomness** ist und welche Faktoren diese in einem Spiel haben. Zudem wird erörtert, warum diese Faktoren hinderlich für die machine learning Software sind. Zusätzlich zu den Arten von Zufallsfaktoren, die sich auf den Zeitpunkt des Zufalls beziehen, gibt es auch noch Zufallsfaktoren, die mit dem Ergebnis zusammenhängen. Am Schluss wird auf die idealste randomness für die Entwicklung von machine learning eingegangen. Randomness ist ein exponentielles Problem und die Rechenzeit muss so gering wie möglich sein, damit die Maschine schnell genug für Strategiespiele in Echtzeit operieren kann.

2.1 Input randomness

Input randomness bezieht sich auf den Zufallsfaktor, der entsteht, bevor eine Entscheidung getroffen wurde. Als Beispiel kann man ein beliebiges Kartenspiel nehmen. Man zieht seine Karten und trifft dann die Entscheidung.

Dies kann man nochmal in zwei Unterkategorien aufteilen, basierend auf den möglichen Ergebnissen. In einem Fall ziehen mehrere Spieler vom selben Deck. Dies hat zur Folge, dass dem Spieler nicht garantiert ist, dieselben Karten zu ziehen, weshalb das Ergebnis (Karten) variabel ist. Im Gegensatz dazu, wenn die Spieler Karten von einem eigenen Deck ziehen, sind diese garantiert, dasselbe Ergebnis zu erhalten. [4]

2.2 Output randomness

Im Unterschied zu Input randomness wird der Zufallsfaktor nach der Entscheidungsfindung getroffen. Ein solches Beispiel ist Dungeon and Dragons. Ein Spieler entscheidet, dass dieser die Wand klettern möchte und muss dafür einen Würfel werfen, um zu sehen, ob dieser seine Tat umsetzen kann.

Dieses Beispiel würde auch unter der Kategorie Output randomness mit einem variablen Ergebnis fallen, da der Würfel, der in Dungeons and Dragons verwendet wird, ein Ergebnis zwischen 1 bis 20 haben kann.



Figure 1: Ein Würfel mit 20 Seiten. [6]

Man könnte anstelle von einem Würfel zu werfen, ein Deck mit 20 Karten verwenden. Dabei hat jede Karte einen Wert von 1 bis 20. Der Spieler würde zuerst die Entscheidung treffen und dann eine Karte ziehen. Dadurch ist dem Spieler bei dieser Methode garantiert, jede Zahl einmal zu erhalten, bevor das Deck neugemischt wird. Der Erfolg der Entscheidung wäre noch immer basierend auf der gezogenen Karte. [4]

2.3 unberechenbare Zufallsfaktoren

Eine machine learning Software besteht aus Algorithmen. Diese Algorithmen haben eine Logik, die die Software befolgt. Glücksfaktoren sind für das Planen eines Zuges hinderlich, da diese zusätzliche Möglichkeiten sowie deren Wahrscheinlichkeiten erschaffen und dies ist eine Herausforderung, für einen Algorithmus abzuwägen. Man muss diese nämlich berücksichtigen für Rechenzeit, weil die Anzahl der Möglichkeiten ein proportionale komplexere Berechnung darstellt. Daher wird zuerst auf die Auswirkung auf eine machine learning Software, wenn diese ein Spiel mit Input randomness mit variablen Ergebnissen spielen würde, eingegangen. Danach wird Input randomness mit garantierten Ergebnissen erläutert. Dies wird gefolgt von Output randomness mit variablen Ergebnissen und zum Schluss wird Output randomness mit garantierten Ergebnissen erklärt.

Die machine learning Software könnte errechnen, welche der möglichen Karten, diese sie vom Kartenstapel ziehen könnte. Dies würde aber nur unnötig Arbeitszeit und Kapazität kosten, da das Ergebnis irrelevant ist,

weil die Maschine keinen Einfluss auf das Ziehen hat und die Berechnung für alle eventuellen gezogenen Karten und die darauffolgenden Aktionen, wäre viel zu viele zum Berechnen. Zudem müssten alle möglichen gezogenen Karten der Gegner berücksichtigt werden, um eine Strategie zu entwickeln. Allein die Berechnung der möglichen Starthände von den anderen Spielern, würde viel zu lange dauern. Bei einem Spiel mit 64 Karten und 7 Handkarten am Anfang des Spiels beläuft sich die Anzahl der Möglichkeiten nur für die **eigene** Starthand auf 621.216.192. Deswegen würde es sich nicht wirklich Lohnen eine machine learning Software für ein solches Spiel zu entwickeln, da diese nicht in der Lage ist eine Strategie zu entwickeln und selbst wenn diese in der Lage wäre die Anzahl der Möglichkeiten zu berechnen, könnte diese trotzdem durch eine "schwächere" machine learning Software besiegt werden, wenn die schwächere bessere Karten zieht.

Wenn eine machine learning Software ein Spiel mit Input randomness und garantierten Ergebnissen spielt, könnte diese eher eine Strategie entwickeln. Anstatt einen allgemeinen Kartenstappel zu haben, hätte jeder Spieler seinen eigenen. Die Karten im Stappel sind den Spielern bekannt. Aufgrund dieser Umstellung muss die machine learning Software nicht mehr notwendiger weiß die Karten der Gegner berechnen, um die eigenen Karten zu bestimmen. Zudem ist die Software in der Lage anhand der gespielten Karten vorherzusagen, welche Karten im verbleiben Deck der Gegner ist. Wodurch diese eine Strategie entwickeln kann.

Die Berechnung für eine machine learning Software in einem Spiel mit Output randomness und variablen Ergebnissen ist einfacher. Im Beispiel von Dungeon and Dragon gibt es nur hauptsächlich nur zwei Ergebnisse, und zwar Erfolg oder Misserfolg. Dies wird anhand eines 20 Seitigen Würfel bestimmt (siehe Graphik 1) Die Wahrscheinlichkeit zu errechnen ist nur zum Teil möglich, da die machine learning Software nicht genau das Würfelergebnis weiß, welches sie erreichen muss, für einen Erfolg. Trotz der geringen möglichen Ergebnisse ist es für die machine learning Software schwer eine Strategie zu entwickeln, da der Versuch einer Tat immer abhängig von einem variablen Ergebnis ist und dies betrifft auch die Gegner. Wenn zum Beispiel eine machine learning Software nur gegen Gegner kämpft, die nicht in der Lage sind diese zu treffen, führt dies zum Fehlschluss, dass die machine learning Software keine Verteidigung braucht. Durch die Natur des Spieles und der Willkür ist es der machine learning nur sehr schwer möglich eine gute konstante Strategie zu entwickeln.

Um ein konstantes Ergebnis zu bekommen, muss der Würfel zum Beispiel mit einem Kartendeck, welches die möglichen Ergebnis eines Würfel hat, ausgetauscht werden. Dadurch wird die machine learning Software nach 20 Entscheidungen alle möglichen Ergebnissen von 1 bis 20 einmal erhalten haben. Diese Änderung ermöglicht es, die Wahrscheinlichkeit für gewissen Ergebnisse zu bestimmen. Das Problem würde jedoch weiterhin bestehen,

dass die machine learning Software nicht weiß, welches Ergebnis diese für einen Erfolg braucht. Dadurch wird das Entwickeln eines Plans schwierig.

Von diesen vier Möglichkeiten ist Input randomness mit garantierten Ergebnissen am berechenbarsten. Zudem ist es zu berücksichtigen, dass der Output eines anderen Spielers in einem Mehrspielerpiel dem random Input für die machine learning Software entspricht. Daher muss jede machine learning Software mit einem gewissen Grad von Zufall zureckkommen. Deswegen werden auf keine Spiele in dieser Arbeit eingangen, die primär eine andere Variante haben. Solche Spiele sind zum Beispiel Schach, Go oder auch StarCraft 2.

3 Lernverfahren von machine learning

In diesem Kapitel wird auf die Lernverfahren **Decision Tree Learning** und **Monte Carlo Tree Search (MCTS)** eingegangen. Decision Tree Learning wurde ausgewählt, da dieses einer der übersichtlicheren Lernverfahren ist. Zum Vergleich von Decision Tree Learning wird MCTS vorgestellt. Es gibt noch weitere Verfahren wie zum Beispiel Ensemble Learning, Bayes Classifiers, Clustering und viele weitere. Aufgrund des limitierten Platz dieser Arbeit werden die anderen Verfahren nicht weiter erklärt.

3.1 Decision Tree Learning

Decision Tree Learning versucht eine Aufgabe zu lösen, durch das Entlang gehen mehrerer binärer Entscheidungen. Eine solche Aufgabe kann zum Beispiel sein zu bestimmen, ob man eine Gewinnposition in Tic tac toe hat. Um bestimmen zu können, ob ein man eine Gewinnposition hat, müssen verschiedene Kriterien durchgangen werden. [30] Dabei würde man erstmal überprüfen, welcher Spieler am Zug ist. Ein anderes Kriterium ist das Überprüfen des Algorithmus, ob dieses mit einem Zug gewinnen kann. Ein Beispiel für einen solchen Decision Tree wird in Graphik 1 gezeigt.

Man kann diesen Decision Tree weiter verbessern, indem man die Informations-Entropie Formel verwendet. Dabei werden die einzelnen Kriterien auf ihren Informationsgehalt anhand einer Informationstabelle bestimmt. Umso höher der Informationsgewinn, umso höher ist das Kriterium decision Tree. Dabei kann auch festgestellt werden, ob gewisse Kriterien keinen Informationsgehalt haben und können dementsprechend entfernt werden. Dies wird auch als Pruning bezeichnet. [30]

3.2 Monte Carlo Tree Search (MCTS)

Der MCTS Algorithmus basiert auf den vier Phasen **Selection**, **Expansion**, **Simulation** und **Backpropagation**. Jeder Knoten im MCTS Algo-



Figure 2: Ein Beispiel Decision Tree für Tic tac toe. [26]

rithmus hat zwei Informationen. Die Anzahl der Besuche und die Anzahl der Gewinne.

In der Selection Phase wird der Knoten basierend auf dieser Formel bestimmt:

$$\frac{w}{n} + C \sqrt{\frac{\ln N}{n}}$$

w steht für die Anzahl der Gewinne. n steht für die Anzahl der Besuche. C ein Zufallsparameter N steht für die Anzahl der Besuche von Eltern-Knoten.

Anschließend wird in der Expansion Phase ein neues Kind-Knoten erschaffen, das am vorherigen Knoten (Eltern-Knoten) hängt. Danach simuliert der MCTS Algorithmus das Kind-Knoten basierend auf zufälligen Entscheidungen. Die Simulation endet, sobald ein Sieg/Niederlage festgestellt wurde oder die Anzahl der Simulationsschritte überschritten sind. Zum Schluss wird in der Backpropagation Phase die gewonnenen Informationen in das Kind-Knoten, sowie alle Eltern-Knoten, vermerkt. Dieser Prozess wird wiederholt, bis der optimale Weg gefunden ist. Der Parameter C wird dabei verwendet, damit auch anderen Knoten getestet werden, da diese eventuell besser sein könnten.

Der Vorteil dieses Algorithmus ist, dass man diesen zu einem beliebigen Zeitpunkt unterbrechen kann und dieser trotzdem den besten möglichen Zug gibt, den dieser berechnet hat. MCTS wird deswegen auch von der machine learning Software AlphaGo Zero verwendet. [17]

4 Probleme und Lösungsansätze

Dieses Kapitel geht auf die Probleme und Möglichkeiten sowie Lösungsansätze ein. Dabei wird auf das Problem der Class Imbalance und Overfitting/Underfitting eingegangen. Die Probleme der Class Imbalance und Overfitting/Underfitting wurden ausgewählt, da diese besonders

im Zusammenhang mit Zufallsfaktoren häufiger auftreten. Es gibt weitere Probleme, aber die sind eher Spiel spezifisch, wie zum Beispiel den Wert einer Schachstellung oder Go Position zu bestimmen. Da diese Probleme Spiel spezifisch sind, werden diese nicht in diesem Kapitel behandelt.

4.1 Class Imbalance

Class Imbalance beschreibt das Problem, wenn eine machine learning Software anhand von Testdaten lernt, die nicht ausgewogen sind. Damit ist gemeint, dass die Anzahl der positiven und negativen Fällen gleich vertreten sind. [30] Wenn man zum Beispiel einen fiktiven Datenbanksatz mit 6 Siegen und 14 Niederlagen nimmt, leidet diese unter Class Imbalance. Dies ist ein Problem, da die machine learning Software immer bestimmen könnte, dass diese eine Niederlage hat und damit mit einer Wahrscheinlichkeit von 70% richtig liegen würde. Deswegen sollten die Testdaten im Optimalfall 50/50 sein.

Es gibt mehrere Möglichkeiten dieses Problem zu beheben. Eine Möglichkeit besteht darin mehr Informationen für den Datensatz zu sammeln. Im fiktiven Beispiel würde man von den 20 Fällen auf 30 erweitern und hoffen, dass die Fälle mehr ausgewogen werden. Diese Methode ist keine Garantie, dass der Class Imbalance verschwindet. Es könnte auch zum Gegenteil kommen und es verschlimmern. Dennoch ist der Ansatz mehr Informationen zu sammeln immer sinnvoll. [16]

Die nächste Möglichkeit wäre den Datensatz anzupassen. Dabei gibt es die Möglichkeit die Anzahl der Fälle, die dominieren, zu reduzieren oder die Anzahl der Fälle, die in der Unterzahl sind, zu verdoppeln. Beide Methoden haben ihre stärken und schwächen. Wenn man die Fälle mit dem Ergebnis Siegen verdoppelt, würde dies gegen die Class Imbalance vorgehen, aber es könnte zu einem Problem von Overfitting (Kapitel 4.2) geben. Auf der anderen Seite, wenn man Fälle mit dem Ergebnis Niederlage entfernt, kann dies zu einem Underfitting (Kapitel 4.2) Problem führen. Daher sollte man moderate mit diesen Methoden vorgehen. [16]

Eine andere Möglichkeit ist es künstliche Daten in den Datensatz hinzuzufügen. Es gibt verschiedene Algorithmen, um dies zu erreichen. Einer dieser Algorithmen lautet Synthetic Minority Over-sampling Technique (SMOTE). Dieser begutachtet, die Fälle, die in der Unterzahl sind, und generiert ähnliche Daten zu den bestehenden Fällen derselben Kategorie. Weshalb die Wahrscheinlichkeit für ein Overfitting Problem bei dieser Methode geringer ist. [16]

4.2 Overfitting/ Underfitting

Overfitting/ Underfitting beschreibt das Problem, wenn die machine learning Software zu spezifisch gelernt hat oder diese zu allgemein ist. Die

machine learning Software soll zum Beispiel feststellen, ob die Schachposition gut für weiß oder schwarz ist. Im Falle von Overfitting hat diese hauptsächliche nur Daten von Schachpositionen bekommen, die Vorteilhaft für weiß sind. Dies kann zur Folge haben, dass die machine learning Software nicht in der Lage vorteilhafte Positionen für schwarz zu identifizieren. [12]



Figure 3: Ein Pixel wurde geändert. Das Ergebnis ist unter dem ursprünglichen Objekt. [28]

Wie man in der Graphik 3 erkennt, kann das Problem von Overfitting sehr extrem sein, dass bei dem Wechsel von einem Pixel ein Pferd zu einem Frosch mit einer Sicherheit von 99.9% wurde. Selbst wenn der Datensatz für die machine learning Software diverse ist, kann es zu Overfitting kommen. Mit cross-validation kann Overfitting reduziert werden. Dabei wird der Datensatz in einen Trainingssatz und in einen Validierungssatz aufgeteilt. Die Sätze werden zufällig verteilt, damit die machine learning Software keine spezifischen Tendenzen, zu einem gewissen Satz, entwickeln kann. Diese lernt an dem Trainingsatz und wird mit den Validierungssatz überprüft, ob ein Overfitting Problem sich gebildet hat. [12]

Underfitting im Gegensatz zu Overfitting ist das Problem, wenn die machine learning Software zu allgemein ist. Nimmt man das Beispiel mit der Schachposition wieder, wäre das Problem mit Underfitting, dass die machine learning Software ein Go-Brett Vorteilhaft für weiß bestimmt. Dieses Problem kann gelöst werden, indem man den Lernalgorithmus spezifischer macht. Dabei würde man zum Beispiel bei der Graphik 1 noch mehr Verzweigungen einbauen. Alternativ kann auch die Lernzeit für die machine learning Software erhöht werden, damit diese mehr Iterationen durchgehen kann.. [30]

5 Einfluss von machine learning Software

In diesem Kapitel werden auf die machine learning Software Alphastar, AlphaGo Zero und Open AI Five eingegangen. Es wurden diese Beispiele ausgewählt, weil Alphastar versuchen muss, mit limitierten Informationen, einen menschlichen Gegner in Echtzeit zu besiegen. AlphaGo Zero hat die Herausforderung, dass diese Software das komplexeste Brettspiel in der Welt spielt. Open AI Five wurde ausgewählt, weil in diesem Spiel nicht nur eine Iteration von Open AI Five spielt, sondern fünf verschiedene, die alle

miteinander kooperieren müssen, um zu gewinnen. Dabei wird erläutert, wie erfolgreich die machine learning Software in ihrem Spiel ist. Außerdem wird erörtert, inwieweit die machine learning Software eine Gefahr für das eigene Spiel darstellt und dieses obsolet macht oder das Spiel bereichert.

5.1 Alphastar

Alphastar wurde von DeepMind entwickelt für das Spiel StarCraft 2 von Blizzard Entertainment. StarCraft 2 ist ein Echtzeitstrategiespiel, bei welchem die Kontrahenten sich gegenseitig versuchen zu besiegen. In einem Echtzeitstrategiespiel bekommen alle Kontrahenten eine Basis auf einer Karte, die sich mit jedem Spiel ändern kann, und müssen versuchen eine funktionale Wirtschaft und Militär zu bauen. Dabei kann die Anzahl der Kontrahenten von zwei bis zu acht Personen reichen.

Zusätzlich gibt es auch die Möglichkeit das Spiel mit Teams zu spielen. Dabei ist die Hauptaufgabe des Spiels sämtliche gegnerische Gebäude zu vernichten. Außerdem hat jeder Spieler eine Auswahl, die dieser vor dem Spiel treffen muss, zwischen drei Fraktionen. Die drei Fraktionen lauten **Terran**, **Protoss** und **Zerg** (siehe Graphik 4). Terran sind futuristische Menschen, die dementsprechende Einheiten besitzen wie zum Beispiel Marines, Bombenflugzeug mit Tarnkappentechnologie oder auch Belagerungspanzer. Protoss entsprechen futuristische Aliens, die sich auf Robotik spezialisiert haben. Die Zerg stellen eine alien Rasse dar, die alles aus biologischem Material herstellen.



Figure 4: Unten links sind die Zerg (Violett). Oben in der Mitte sind die Terraner (Gelb). Unten links sind die Protoss (Blau). [25]

In Starcraft 2 werden öffentliche als auch private Turniere mit Preisgeldern veranstaltet. Dabei war der Preispool von StarCraft 2 bisher **164.661,28\$** bis zu **700.000\$** belaufen. [9] Zudem kann man auch seinen StarCraft 2

Account verkaufen. Der Preis kann zwischen 25\$ bis zu 350\$ varieren. [21]

Alphastar wurde 2019 veröffentlicht und war zu diesem Zeitpunkt auf dem Niveau eines Grandmasters. Dies hat die Bedeutung das Alphastar besser als 99,8% der bestehenden Spieler war. [24] Daher könnte Alphastar verwendet werden, um einem Kontrahenten in einem Turnier einen unfairen Vorteil zu verschaffen. Jedoch ist die Wahrscheinlichkeit, dass dies zu einem Problem wird, unwahrscheinlich. Bei Turnieren werden die genutzten Computer von den Kontrahenten auf Software überprüft, die einem Kontrahenten einen unfairen Vorteil verschaffen würde.

Auf der anderen Seite könnte Alphastar verwendet werden, um den Wert im **Ranking** für einen Account zu erhöhen. Ranking beschreibt ein System, bei welchem jeder Spieler einen Wert zu geteilt wird. Dabei soll dieser Wert die Fähigkeit des Accountnutzers widerspiegeln. Spieler kaufen einen solchen Dienst oder einen Account, der den gewünschten Wert im Ranking hat, für Prestige. Dies führt jedoch zum Problem, dass dann Spieler gegen einen unfairen Gegner im Ranking antreten, müssen. Dafür gibt es Beispiel wie "Fortnite" oder auch "Fall Guys: Ultimate Knockout". [14]



Figure 5: Spiel cover von "Fall Guys: Ultimate Knockout" [11]

Jedoch ist es unwahrscheinlich, dass StarCraft 2 Ranking aufgrund von Alphastar zerstört wird. Da der kommerzielle Gewinn beim Verkauf eines Accounts im Verhältnis zum Aufwand minimal ist und dies sich für ein Unternehmen wie DeepMind nicht investiert.

Man muss auch berücksichtigen, dass Alphastar im Gegensatz zu anderen machine learning Software nicht von einem menschlichen Spieler unterscheidbar ist. Die einzigen Maßstäbe, die man verwenden kann, um zwischen einem Menschen und einer Maschine zu unterscheiden, sind die Anzahl der Aktionen, die ein Spieler in einem kurzen Zeitraum durchführt und die Art dieser Befehle. Es ist zu berücksichtigen, dass verschiedene Fraktionen es dem Spieler ermöglichen eine höhere Anzahl von **Aktionen pro Minute** (APM) durchführen zu lassen. Ein menschlicher Spieler ist dabei limitiert, aufgrund von Informationsaufnahme und Verarbeitung. Dies kann eine Mas-

chine deutlich schneller, weshalb diese mehr Befehle im selben Zeitraum geben kann.

Die andere Möglichkeit, um eine Maschine festzustellen, ist es zu beobachten, welche Befehle diese in einem kurzen Zeitraum gibt. Ein normaler menschlicher Spieler ist darauf limitiert, dass dieser nur eine Maus hat, um Befehle zu geben. Daher kann dieser nicht innerhalb einer Sekunde zehn verschiedene Befehle geben (siehe Graphik 6).



Figure 6: Bewegungsbefehle zweier machine learning Algorithmen. [15]

Eine Maschine hingegen, kann zum Beispiel zehn individuellen Einheiten zehn verschiedene Bewegungsbefehle geben. Das Problem mit Alphastar ist, dass diese Maschine diese menschlichen Limitierungen auch besitzt, weshalb man den Unterschied nicht erkennen kann.

Abgesehen davon hat Alphastar auch positive Aspekte. Spieler hatten die Möglichkeit gegen Alphastar zu spielen. [10] Dies hilft den Spielern ihre eigene Fähigkeiten zu verbessern.

5.2 AlphaGo Zero

DeepMind hat AlphaGo Zero für das Spiel Go entwickelt. Go ist ein altes chinesisches Brettspiel für zwei Spieler. Das Ziel des Spieles ist am Ende die meisten Punkte zu haben. Punkte kann man durch das Erobern gegnerischer Steine oder das kontrollieren von einem Bereich erhalten.

Es gibt verschiedene Go Turnier mit variierenden Preispoolen. Wenn man in LG CUP gewinnt, erhält man ein Preisgeld von **250.000\$**. [7] Wohin gegen

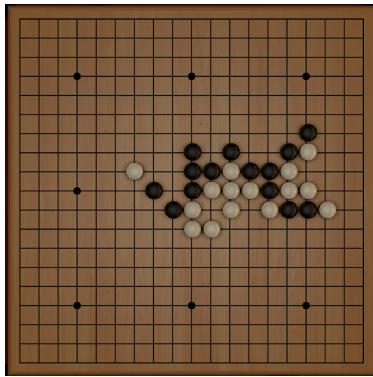


Figure 7: Ein Go-Spiel nach 15 Runden. [27]

ING Cup 2009 ein Preisgeld von **400.000\$** an den Gewinner gab. [13]

AlphaGo Zero war in der Lage nach 3 Tagen Lernens den Weltmeister Lee Sedol zu besiegen. Nach 40 Tagen konnte AlphaGo Zero alle bisherigen machine learning Software in Go besiegen und ist damit **der stärkste Go Spieler** in der Welt. [23] Deswegen kann AlphaGo Zero verwendet werden, um in einem Go Turnier einen unfairen Vorteil zu bekommen. Dies würde wie folgt funktionieren. Der Spieler, der im Turnier gegen einen anderen Spieler antritt, würde die Züge des Gegners einer zweiten Person mitteilen. Diese zweite Person macht dieselben Züge gegen AlphaGo Zero und gibt dem Spieler, im Turnier, die Züge von AlphaGo Zero. Der kommerzielle Gewinn dabei ist zu gering, im Vergleich des Risikos des Aufstiegs und dem darauffolgenden Ruf schaden. Zudem gibt es Software zum Beispiel in Schach, welches von der Logik sehr ähnlich zu Go ist, um herauszufinden, ob ein Spieler sich einen solchen unfairen Vorteil verschafft hat. [5]

DeepMind hat 50 Spiele veröffentlicht, in welchen AlphaGo gegen AlphaGo gespielt hat. [8] Dies kann der Go Gemeinde helfen, neue und bessere Strategien zu entwickeln. Außerdem ist AlphaGo Zero ein Meilenstein hinsichtlich der Entwicklung machine learning, da dieses in Lage ist Go, welches über $10^{10^{100}}$ mögliche Spiele hat, zu spielen. [29]

5.3 Open AI Five

Open AI Five ist eine Software von einer nonprofit Organisation. Dabei wurde diese entwickelt für das Spiel Dota 2. Dota 2 ist eine **multplayer online battle arena** (moba). Es gibt zwei Teams aus jeweils 5 Spielern, die die Aufgabe haben, die feindliche Basis zu vernichten. Am Anfang eines Spiels wählt jeder Spieler einen Charakter aus mit einzigartigen Fähigkeiten.

Dota 2 veranstaltet jährlich Turniere mit Preisgeldern. Dabei beläuft sich der Preispool vom letzten Turnier auf **40.018.195\$** [9]. Außerdem ist es



Figure 8: Ein Kampf, bei welchem 4 Spieler von einem Team 3 Spieler vom anderen bekämpfen. [2]

möglich seinen Account für Geld zu verkaufen. Dabei kann der Preis zwischen 5\$ bis zu 1.600\$ variiieren. [20]

2018 hat Open AI Five die Weltmeister in Dota 2 besiegt. Zudem hat es 7.000 Spiele gegen Menschen gespielt, von denen es 99,4% gewonnen hat. [19] Ähnlich wie bei den anderen Beispielen könnte Open AI Five verwendet werden, um einem Spieler einen unfairen Vorteil zu verschaffen. Wobei zu berücksichtigen ist, dass die Computer, auf denen die Spieler spielen, kontrolliert werden. Daher ist es unwahrscheinlich, dass ein Spieler in einem offiziellen Turnier, einen unfairen Vorteil bekommt.

Da der Preis von einem Dota 2 Account bei über 1.500\$ liegen kann, könnte man die Annahme treffen, dass man Open AI Five verwenden könnte, einen Account mit einem hohen Ranking zu erstellen und zu verkaufen. Dies wäre ein Problem, da es das Ranking von Dota 2 zerstören würde und das Interesse an dem Spiel mit den Investoren sinken würde. [14] Jedoch ist es unwahrscheinlich, dass das jemals ein Problem wird, da einerseits die Ersteller von Open AI Five eine nonprofit Organisation ist und dementsprechend ihr Ziel ist nicht, damit Geld zu verdienen. Zudem kann man auch feststellen, ob die machine learning Software der Spieler ist.

Es werden alle Mausbewegungen und Aktionen des Spielers in einem Spiel analysiert. Dabei werden verschiedene Parameter beobachtet wie zum Beispiel der Abstand der Maus auf dem Bildschirm und dem Ort, an dem die Software den Mausinput registriert hat. Dann wird machine learning Technik verwendet Namens **anomaly detection**. Der Algorithmus lernt basierend auf Spielen, die definitiv keine Betrüger beinhalten. Dabei hat der Algorithmus eine Fehlerrate von weniger als 3%. [3]

Außerdem können andere Spieler feststellen, ob ein Spieler maschinelle Unterstützung hat. Eine Möglichkeit ist es zu beobachten, ob der Spieler

über übernatürliche Reaktionsfähigkeiten verfügt wie zum Beispiel dem regelmäßigen Ausweichen von **Skillshots** oder auch das Treffen dieser. Dies wird ersichtlich, wenn der Spieler versucht, einen Skillshot zu treffen oder auszuweichen. Skillshot beschreibt eine Fähigkeit eines spielbaren Charakters, welche nicht automatisch trifft, sondern sichtbar im Spiel erkennbar ist (Graphik 9).



Figure 9: Das violette Viereck zeigt den Bereich an, in welchem der Charakter getroffen werden würde. Der weiße Bereich zeigt einen Skillshot. [1]

Die Spieler von Dota 2 sind in der Lage die Spiele von Open AI Five zu analysieren und sich dadurch zu verbessern, was ein positiver Aspekt ist. Zudem ist es ein Meilenstein für die machine learning Software, da fünf verschiedene Open AI Five Software, die jeweils einen Charakter gesteuert haben, miteinander gegen ein Team von Menschen gespielt und gewonnen hat. [19]

6 Ausblick für die Zukunft

Anhand der gesammelten Beispiele bin ich definitiv der Meinung, dass Strategiespiele ohne unberechenbare Zufallsfaktoren nicht durch machine learning obsolet werden. Es wird natürlich neue Möglichkeiten geben, im jeweiligen Spiel zu schummeln. Wobei dieses Probleme bekämpft werden können. Auf der anderen Seite finde ich, dass die machine learning Software einen positiven Aspekt für die Spiele hat, sofern man gegen diese spielen kann oder Statistiken oder Information über die Software hat, da die Spieler ihre Strategien anhand dieser verbessern können.

Generell könnten solche machine learning Software sehr nützlich sein, da diese in der Lage sind, komplexe Probleme in kurzer Zeit zu lösen. Daraus können die Entscheidungslogiken dieser machine learning Software zum Beispiel als Grundlage für autonomes Fahren verwendet werden. Eine andere Anwendung könnte die effizientere Gestaltung des Welthandels darstellen, damit Engpässe schneller behoben oder sogar präventiv verhindert werden.

dert werden können.

Machine learning Software besonders in komplexen Spielen wie zum Beispiel StarCraft 2 und Dota 2 werden, aber auch das Interesse des Militärs wecken. Prof. Silver sagt: "*DeepMind has pledged never to develop technologies for lethal autonomous weapons.*" und "*To say that this has any kind of military use is saying no more than to say an AI for chess could be used to lead to military applications*" [18]. Ich würde der zweiten Aussage von Prof. Silver widersprechen. Die künstliche Intelligenz für Schach hat alle Information und bewegt sich immer im selben Umfeld ohne Veränderung. Alphastar hingegen muss mit limitierten Informationen und Anpassungen an das Kernspiel umgehen. Mit Anpassung sind zum Beispiel das hinzufügen neuer Einheiten oder das Anpassen bestehender gemeint. Dies entspricht der Realität, da der Kommander des Militärs nicht alle Informationen über den Gegner verfügt und die Technologie des Militärs sich ständig ändert. Daher zu behauptet, dass diese machine learning Software nicht nützlicher als eine machine learning Software für Schach ist, finde ich falsch. Diese Ansicht vertritt auch das chinesische Militär und Prof. Sharkey: "*the enormous potential of artificial intelligence in combat command*" [18] und "*While DeepMind states that it will never be involved in military work, and Starcraft II is not a realistic war simulation, the results will be of interest to the military*" [22] Wobei DeepMind davor warnt diese Technologie für das Militär zu nutzen, da man nicht die Aktionen der machine learning Software vorhersagen kann in Kombination mit gefährlichen Waffen. Dies könnte zur Folge haben, dass die machine learning Software Gesetzte verletzen würde. [22] Diese Befürchtung teile ich mit DeepMind. Dennoch gehe ich davon aus, dass das Militär versuchen wird, diese Technologie in das Militär zu integrieren und zu verwenden.

7 Anhang

References

- [1] admin. Nidalee spear hitbox - nidalee's aspect of the cougar *., 2020. <https://goimages-power.blogspot.com/2020/05/nidalee-spear-hitbox-nidalee-aspect-of.html>.
- [2] Anuraj. Huge dota 2 teamfight, 2014. <https://www.youtube.com/watch?v=rUGKufMwA=0>.
- [3] GOSU.AI Blog. Gosu.ai against dota 2 cheaters, 2018. <https://gosu.ai/blog/dota2/gosu-ai-against-dota-2-cheaters/>.
- [4] Keith Burgun. Three types of bad randomness, and one good one. *Keith Burgun games*, 2018. <http://keithburgun.net/three-types-of-bad-randomness-and-one-good-one/>.
- [5] ChessBase. Fide: Wie soll man gegen cheater vorgehen?, 2020. <https://de.chessbase.com/post/fide-wie-soll-man-gegen-cheater-vorgehen>.
- [6] Cults. Dual d20, 2017. <https://cults3d.com/en/3d-model/game/dual-d20>.
- [7] LG CUP. Lg cup world baduk championship, 2022. <http://baduk.lg.co.kr/eng/about.asp>.
- [8] DeepMind. Alphago vs alphago, 2017. <https://deepmind.com/alphago-vs-alphago>.
- [9] Esports Earnings. Starcraft 2 prize pools, 2022. <https://www.esportsearnings.com/games/151-starcraft-ii>.
- [10] Blizzard Entertainment. Deepmind research on ladder, 2019. <https://news.blizzard.com/en-us/starcraft2/22933138/deepmind-research-on-ladder>.
- [11] Donovan Erskine. Fall guys: Ultimate knockout hands-on preview: Wipeout. *SHACKNEWS*, 2020. <https://www.shacknews.com/article/119442/fall-guys-ultimate-knockout-hands-on-preview-wipeout>.
- [12] David Forsyth. Applied machine learning, 2019. <https://link.springer.com/book/10.1007%2F978-3-030-18114-7#authorsandaffiliationsbook>.

- [13] GoGameWorld.com. Go tournament: Ing cup, 2022. https://web.archive.org/web/20110929082353/http://gogameworld.com/gophp/pg_titlelist_detail.php?termvalue=Ing%20Cup&selectvalue=title&matchvalue=exact&ordervalue=year.
- [14] Aaron Greenbaum. Games that were ruined by too many cheaters, 2021. <https://www.looper.com/257502/games-that-were-ruined-by-too-many-cheaters/>.
- [15] Harstem. Highest rated starcraft ii ai's fight each other (eris vs micro-machine), 2022. <https://www.youtube.com/watch?v=Qfm2A6aIQOA>.
- [16] Ken Hoffman. Machine learning: How to handle class imbalance. *Analytics Vidhya*, 2021. <https://medium.com/analytics-vidhya/machine-learning-how-to-handle-class-imbalance-920e48c3e970>.
- [17] Simon Cornelius Hülkenberg. Monte carlo tree search for imperfect information games with the example card game 'gwent', 2020. <https://opus4.kobv.de/opus4-haw-landshut/frontdoor/index/index/docId/233>.
- [18] Leo Kelion. Deepmind ai achieves grandmaster status at starcraft 2. *BBC NEWS*, 2019. <https://www.bbc.com/news/technology-50212841>.
- [19] OpenAI, Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Jozefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Ponde de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. 2019.
- [20] PlayerAuctions. Dota 2 accounts for sale - dota ii marketplace, 2022. <https://www.playerauctions.com/dota-2-account/>.
- [21] Playerup. Starcraft 2 accounts - buy sell trade, 2022. <https://www.playerup.com/accounts/starcraftaccount>.
- [22] Ian Sample. Ai becomes grandmaster in 'fiendishly complex' starcraft ii. *The Guardian*, 2019. <https://www.theguardian.com/technology/2019/oct/30/ai-becomes-grandmaster-in-fiendishly-complex-starcraft-ii>.
- [23] David Silver and Demis Hassabis. Alphago zero: Starting from scratch. *DeepMind*, 2017. <https://deepmind.com/blog/article/alphago-zero-starting-scratch>.

- [24] Nick Statt. Deepmind's starcraft 2 ai is now better than 99.8 percent of all human players. *The Verge*, 2019. <https://www.theverge.com/2019/10/30/20939147/deepmind-google-alphastar-starcraft-2-research-grandmaster-level>.
- [25] Lucas Stumm. Die drei fraktionen aus starcraft 2.
- [26] Lucas Stumm. Ein beispiel decision tree für tic tac toe.
- [27] Lucas Stumm. Ein go-spiel.
- [28] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks, 2017. <https://arxiv.org/abs/1710.08864>.
- [29] Matthieu Walraet and John Tromp. A googolplex of go games. In Aske Plaat, Walter Kosters, and Jaap van den Herik, editors, *Computers and Games*, pages 191–201, Cham, 2016. Springer International Publishing.
- [30] Zhi-Hua Zhou. Machine learning, 2021. <https://link.springer.com/book/10.1007%2F978-981-15-1967-3>.

8 Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit eigenständig und ohne fremde Hilfe angefertigt habe. Textpassagen, die wörtlich oder dem Sinn nach auf Publikationen oder Vorträgen anderer Autoren beruhen, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Name (Unterschrift)

Lucas Stumm

March 22, 2022

Ort, Datum