Künstliche Intelligenz am Beispiel des Gesellschaftsspiels Carcassonne

Abkürzungsverzeichnis

KI	Künstliche Intelligenz
NN	Neuronales Netz
ML	Maschinelles Lernen

1. Einleitung

Künstliche Intelligenz (KI) findet in der Gesellschaft immer mehr Anwendung, beispielsweise durch Suchmaschinen, Sprachassistenten oder Gesichtserkennung. Sie wird zunehmend auch in Spielen eingesetzt. Um künstliche Intelligenz an einem praxisnahen Beispiel zu erforschen, soll eine KI für das Gesellschaftsspiel Carcassonne entworfen werden, die als Gegner agiert und das Ziel hat zu gewinnen. Das Spiel wurde gewählt, da die KI-Entwicklung große Erfolge im Bereich der Gesellschaftsspiele erzielt hat und Carcassonne verschiedenste Strategieansätze hat, jedoch in der Vergangenheit keine Beachtung in der KI-Entwicklung gefunden hat, bis auf spieltheoretische Ansätze[1].

2. Definition von künstlicher Intelligenz

Künstliche Intelligenz bezeichnet die automatisierte Lösung von teilweise unbekannten Problemen basierend auf Erfahrung, die durch Lernprozesse erlangt wird.

Algorithmen fallen nicht unter diese Definition, wenn sie nur bekannte Probleme lösen können oder anhand eines vordefinierten Entscheidungsbaum agieren, den sie nicht in einem eigenen Lernprozess erhalten.

3. Künstliche Intelligenz am Beispiel des Gesellschaftsspiels Carcassonne

3.1. Spielregeln

Quelle: eigene Aufnahme

Carcassonne ist ein Kartenlegespiel für zwei bis fünf Spieler*Innen mit dem Ziel die meisten Punkte zu erzielen. Das Spiel startet mit einer Weg-Stadt Karte, in jeder Runde zieht ein/e Spieler*In von einem Ziehstapel eine Karte, auf der Strukturen wie z.B. Wege, Wiesen, Städte oder Klöster abgebildet sind. Diese legt man passend an die bereits auf dem Tisch befindlichen Karten an, sodass gleiche Strukturen aneinandergrenzen. Um Punkte zu erzielen kann man beim Legen der Karten eine von seinen sieben Figuren auf eine Struktur setzen. Um doppelte Punkte zu erhalten und seine Figur zurückzuerlangen werden die Strukturen vollendet, indem Wege beendet, Städte geschlossen oder Klöster eingeschlossen werden. Das Spiel endet, wenn keine Karte mehr gezogen werden kann und der/die Spieler*In mit den meisten Punkten gewinnt. Die Spielregeln sind unter dem Verlag „Hans im Glück“ [10] zu finden.

3.2. Zielsetzung

Die KI hat das Ziel gegen eine/n Mitspieler*In das Spiel Carcassonne zu gewinnen. Für diesen Zweck bekommt sie das Spielfeld, die gezogene Karte und die Spieler*Innenpunkte als Eingabe und soll daraufhin die Koordinate, Rotation und Figurposition (oben, unten, links, rechts, mittig, keine) für die gezogene Karte bestimmen. Dabei sollen nicht nur die Kartentypen und gültige Spielzüge erlernt werden, sondern wenn möglich auch Strategien entwickelt werden.

3.3. Aufbau der künstlichen Intelligenz

Die künstliche Intelligenz, die Carcassonne gewinnen soll, besteht aus einem Neuronalen Netz, dass mithilfe von maschinellem Lernen so angepasst wird, dass es basierend auf der Spielsituation den möglichst besten Zug vorhersagt. Ein Neuronales Netz (NN) ist ein schichtenbasierter Zusammenschluss von Neuronen, dass basierend auf Eingabedaten eine oder mehrere Wahrscheinlichkeiten berechnet.

Ein Neuron ist ein Knotenpunkt in diesem Netz, welches die Eingaben (= inputs) der vorherigen Schicht mithilfe von justierbaren Gewichten (= weights & biases) zu Ausgabewerten

transformiert und basierend auf Aktivierungsschwellen (= thresholds) an die Neuronen der nachfolgenden Schicht übergibt.

Um Eingabedaten in das Netzwerk geben zu können, muss das Spielfeld in ein maschinenlesbares Format umgewandelt werden. Dazu wird jedem Kartentyp eine eindeutige Nummer zugewiesen und zusammen mit der Kartenrotation an das Netzwerk übergeben. Dabei werden die Karten sequenziell von oben nach unten und von links nach rechts ihrer entsprechenden Spielfeldposition übergeben, um deren X und Y Koordinate anzugeben. Wenn an einer Spielfeldposition keine Karte vorhanden ist, wird stattdessen eine leere Karte übergeben.

Das Spielfeld von Carcassonne ist theoretisch nicht begrenzt, doch praktisch durch die 71 vorhandenen Karten limitiert. Deshalb wird die Größe des neuronalen Netzes auf eine Obergrenze festgelegt. Hierfür wurde eine passende Feldgröße ermittelt, indem 50 zufällige Spiele simuliert und am Ende jeweils die Feldgröße gemessen wurde. Die durchschnittliche benötigte Größe eines Spielfeldes beträgt 12.7 Karten, wobei die größte Spielfelddimension 17 und die kleinste 9 groß war. Deshalb wird die maximale Dimension eines Spielfeldes auf 15 festgelegt, um die Anzahl der Eingabedaten und deren Berechnung zu minimieren und damit zu beschleunigen. Die KI berechnet für jede Spielfeldposition eine Wahrscheinlichkeit zwischen 0 und 1. Anschließend wird auf die Spielfeldposition mit der höchsten Wahrscheinlichkeit die nächste Karte gesetzt. Für diese Karte wird anschließend die Rotation und Figurposition bestimmt.

3.4. Maschinelles Lernen

Maschinelles Lernen (ML) bezeichnet das automatisierte Anpassen der Gewichte eines NN durch Mustererkennung mittels einer Vielzahl von Beispieldaten und wird häufig im Lernprozess einer künstlichen Intelligenz eingesetzt.

3.4.1. Lernansätze

Für ML gibt es abhängig vom Anwendungszweck der KI unterschiedlich optimale Lernansätze. Deshalb werden im Folgenden verschiedene Lernansätze bezüglich ihrer Anwendbarkeit auf eine KI für das Spiel Carcassonne evaluiert.

3.4.1.1. Überwachtes Lernen

Überwachtes Lernen (= supervised learning) bezeichnet das eigenständige Lernen von Korrelation zwischen Eingabe und erwarteten Ergebnis durch eine Großzahl von Beispieldaten. Bezogen auf Carcassonne würde die KI mit einer Vielzahl von Spielverläufen trainiert werden. Daraus würde die KI mögliche Strategien erlernen und basierend auf einer gegebenen Spielsituation den besten Zug berechnen. Jedoch ist dieser Ansatz auf Carcassonne nicht anwendbar, da es keine Vielzahl von öffentlich einsehbaren Spielpartien gibt.

3.4.1.2. Unüberwachtes Lernen

Unüberwachtes Lernen (= unsupervised learning) bezeichnet das eigenständige Lernen von Mustern in einer großen Menge von Daten. Jedoch ist diese Lernmethodik nicht sinnvoll, da das Ziel der KI darin besteht den besten Zug vorherzusagen, denn unüberwachtes Lernen wird verwendet, um Muster oder Strategien in vergangenen Spielverläufen zu erkennen. Zusätzlich benötigt diese Form des Lernens auch eine Vielzahl von Spielpartien, die für Carcassonne nicht vorhanden sind.

3.4.1.3. Bestärkendes Lernen

Bestärkendes Lernen (= reinforcement learning) bezeichnet das Aneignen von Aktionen basierend auf negativer und positiver Rückmeldung. Hierfür agiert die KI als Agent (= Spieler) in einer Umgebung und kann Aktionen ausführen. Dafür erhält die KI das Spielfeld und die gezogene Karte als Eingabe und kann daraufhin eine Karte legen und optional eine Figur darauf setzen. Wenn die Karte nicht an die Stelle gelegt werden kann, so erhält sie eine negative Rückmeldung und wenn sie Punkte erzielt, eine positive. Dies wird so lange wiederholt, bis sie korrekte und sinnvolle Züge durchführt. Diese Lernmethodik wurde für die Carcassonne KI gewählt, da sie keine Beispieldaten benötigt und sie bereits große Erfolge in der KI-Entwicklung von Gesellschaftsspielen erzielt hat. Bestärkendes Lernen ermöglicht der KI neue Strategien zu entwickeln, die in fiktiven Beispieldaten nicht vorhanden sind.

3.4.2. Backpropagation

Bei der Erstellung von NN sind die weights und biases zufällig initialisiert und führen noch nicht zu einem gewünschten positiven Ergebnis. Deshalb wird Backpropagation für das Lernen von NN verwendet, um die Gewichte anzupassen, sodass die Fehlerquote sinkt und die Belohnung

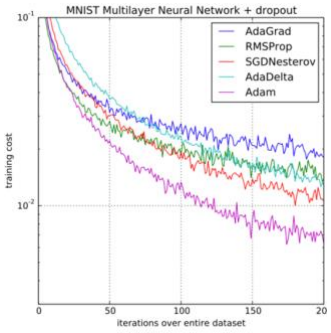
steigt. Hierfür wird die Fehlerquote C (= cost function) für jedes Ausgabeneuron berechnet. Wobei n der Index des Neurons, y die vorhergesagte Wahrscheinlichkeit und g die gewünschte Wahrscheinlichkeit ist:

$C_n = \frac{1}{2}(y_n - g_n)^2$.

Diese Fehlerquote wird mithilfe des Gradientenverfahren benutzt, um die neuen Werte der Gewichte zu berechnen. Wobei η eine konstant festgelegte Lernrate (= learning rate), ∇w der Gradient der vorherigen Gewichte und W das Gewicht des Neurons ist: $W_{neu} = W_{alt} - \eta \cdot (\nabla \mathrm{w} \cdot \mathrm{C}) \cdot W_{alt}$

3.4.3. Optimizer

Jedoch ist das Gradientenverfahren sehr rechenintensiv, da es für jeden Trainingsschritt rekursiv die Gradienten aller Gewichte berechnen muss. Daher wurden verschiedene Optimierungen dieses Gradientenverfahren nach ihrer Effizienz untersucht und der erfolgreichste ausgewählt. Der verlustbehaftete Adam[8] Optimizer wurde gewählt, weil dieser andere Optimizer deutlich übertrifft und mit weniger Rechenaufwand in weniger Durchläufen eine der geringsten Fehlerquoten (= training cost) erzielt.



Quelle: Adam[8] (2017)

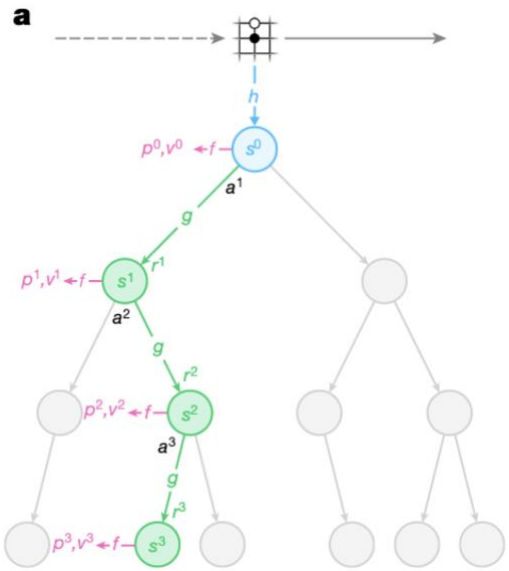
3.5. Umsetzung

Für die Umsetzung der KI wurde eine frei verfügbare Implementierung von MuZero[2] gewählt, dem Nachfolder von Googles AlphaZero Schach KI. Denn MuZero ist nicht auf ein spezifisches Spiel angepasst und kann lernen, ohne die Spielregeln zu kennen, indem es drei Prinzipien anwendet:

- Value: Wie gut ist die momentane Position?
- Policy: Was ist der beste Zug?
- Reward: Wie gut war der letzte Zug?

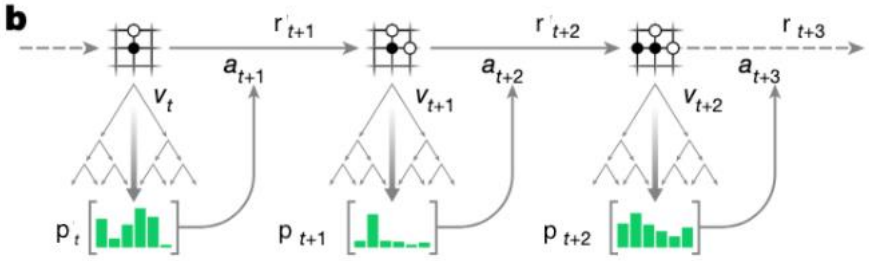
3.5.1. Der MuZero Algorithmus

MuZero basiert auf dem Monte Carlo Suchbaumalgorithmus (= MCTS), der durch neuronale Netze optimiert wird. Weil es im gesamten Spiel zu viele Züge gibt, um alle berechnen zu können, wird mithilfe des value NN die momentane Spielposition bewertet und damit die vielversprechendsten Züge ermittelt. Danach wird mit dem policy NN der beste Zug ausgewählt.



Quelle: Google, MuZero (2020)

Konkret wird ein Spielzug geplant, indem die aktuelle Spielfeldsituation an die Funktion h übergeben wird, die es in ein maschinenlesbares Format umwandelt und in s (= state) speichert. Daraufhin werden basierend auf s mögliche Züge a (= action) berechnet, um die beste Aktion mithilfe der beiden Neuronalen Netzwerken p (= policy NN) und v (= value NN) zu bestimmen. Daraufhin wird diese Aktion mithilfe der Spielfunktion g (= dynamics function) angewendet, die den neuen Spielstand s und die Belohnung r (= reward) zurückgibt. Dies wird so lange wiederholt, bis der beste Zug gefunden wurde oder ein Zeit- beziehungsweise vorausschauendes Zug-Limit überschritten wurde.



Quelle: Google, MuZero (2020)

Diese Spielzugplanung wird für jede Spielrunde durchgeführt und die Rückmeldung r der Funktionen v und p werden verwendet, um mithilfe von Backpropagation die richtigen Züge a zu lernen und damit r zu maximieren.

3.5.2. Implementierung

Um den MuZero Algorithmus anwenden zu können, muss eine Umgebung (= Environment) erstellt werden, in der Aktionen ausgeführt werden können. Dazu wurde das Spiel Carcassonne im Vorfeld programmiert und eine grafische Oberfläche erstellt. Um die Komplexität auf das Nötigste zu reduzieren, wurde auf den Fluss und auf weitere Erweiterungen verzichtet und nur das Basisspiel implementiert.

Zudem wurde ein Agent (= Spieler) entwickelt, der die Züge der KI ausführt und für ungültige Züge einen reward von -1, und für einen validen Zug +1 zurückgibt. Die KI erhält zusätzlich eine positive Rückmeldung, wenn sie Punkte erzielt und +100, wenn sie das Spiel gewinnt.

Da die MuZero KI jedoch für jede Situationseingabe nur eine Aktion ausführen kann, muss ein Spielzug in folgende drei Schritte unterteilt werden: Karte rotieren (-1), Karte legen (0) und Figur setzen (1). Der Bereich -1 bis 1 wurde gewählt, um die Eingabedaten zu normalisieren, weil das NN im Bereich von |0 bis 1| rechnet. Nun wird der aktuelle Spielschrittstatus (-1, 0, 1), zusätzlich zur Spielsituation, übergeben wird.

Die KI berechnet dann die Wahrscheinlichkeiten für alle Spielfeldpositionen, Kartenrotationen und Figurpositionen. Danach wird für jeden Schritt die Aktion mit der größten Wahrscheinlichkeit ausgeführt. Dafür erhält die KI eine Rückmeldung (= reward) und wird entsprechend dem MuZero Algorithmus trainiert.

Hierbei spielt die KI gegen sich selbst und benötigt daher keine weiteren menschlichen Eingaben. Außerdem kann die KI dadurch besser werden als menschliche Spieler*Innen, da sie zusätzlich einen reward erhält, wenn sie bessere Züge spielt als sie selbst. Als Kontrollgruppe wird ein computergesteuerter Spieler benutzt, der valide Züge berechnet und einen zufälligen Zug auswählt.

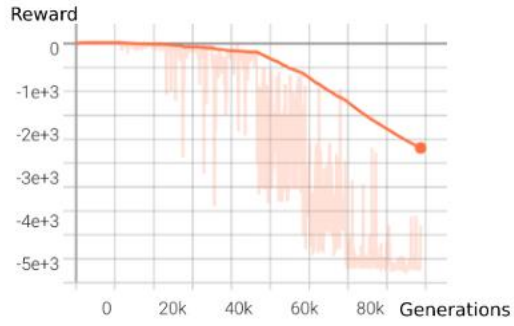
Zum Training der KI werden viele Spiele durchgeführt und um den Lernfortschritt zu beobachten, wird zusätzlich zur graphischen Oberfläche des Spiels ein TensorBoard[6] verwendet, welches den reward der KI über viele Iterationen (= generations) darstellt. Dies

wird so lange wiederholt, bis die KI das Spiel meistert oder der Lernfortschritt sich nur noch marginal verbessert.

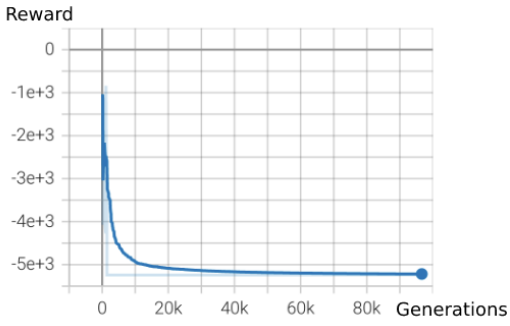
4. Fazit

Zusammenfassend sind die Funktionsweisen einer künstlichen Intelligenz ersichtlich geworden und wurden anschaulich an einer KI für das Spiel Carcassonne dargestellt.
Anhand der gewonnenen Erkenntnisse wurde daraufhin die KI implementiert und nach mehreren Tagen des KI-Trainings gegen unterschiedliche Gegner, wurden folgende Ergebnisse festgestellt:

KI spielt gegen zufällig agierenden Spieler



KI spielt gegen sich selbst



Quelle: eigene Grafik

Der reward fällt bei dem zufällig agierenden Spieler als Gegner deutlich langsamer ab im Vergleich zur KI, die gegen sich selbst spielt. Ursache dafür ist, dass die KI zu Beginn die Regeln und Züge noch nicht erlernt hat und daher nicht aus den validen Zügen eines zufällig agierenden Spielers lernen kann. Insgesamt fällt der reward bei beiden Gegnern negativ aus. Dies bedeutet, dass die KI ungültige Züge durchführt und kann darauf zurückzuführen, dass die KI die Züge in drei Schritten vollziehen muss und die Reihenfolge der Schritte nicht erlernt hat.
Eine andere Erklärung für den nicht vorhanden Lernerfolg könnte die erhebliche Menge der 6750 Eingabedaten sein, da für jede der 225 Karten zusätzlich die Kartentypen durch 20, die Rotationen durch 4 und die Figurpositionen durch 6 boolesche Werte angegeben werden müssen.
Mögliche Verbesserungen für weiterführende Forschungen in diesem Gebiet wäre es den MuZero Algorithmus anzupassen, sodass dieser die einzelnen Schritte in einem einzigen Zug

berechnet und damit der komplizierte Ablauf der sequenziellen Zugführung wegfällt. Außerdem könnten die Karten in ihre einzelnen Strukturen aufgeteilt und dann an die KI übergeben werden, sodass sie nicht die unterschiedlichen Kartentypen erlernen muss.

5. Literaturverzeichnis

[1] Cathleen Heyden (2009): „Implementing a computer player for Carcassonne“ https://project.dke.maastrichtuniversity.nl/games/files/msc/MasterThesisCarcassonne.pdf [Aufgerufen am 16.11.2020]

[2] Werner Duvaud, Aurèle Hainaut (2019): „MuZero General: Open Reimplementation of MuZero“ https://github.com/werner-duvaud/muzero-general [Aufgerufen am 27.09.2021]

[3] Source Code (2021): https://github.com/Flam3rboy/carcassonne-ai [Aufgerufen am 17.03.2021]

[4] Jan Philipp Schwenck (2008): „Reinforcement Learning zum maschinellen Erlernen von Brettspielen am Beispiel des Strategiespiels 4-Gewinnt“ http://www.gm.fh-koeln.de/~konen/Diplom+Projekte/PaperPDF/DiplomSchwenck08.pdf [Aufgerufen am 11.01.2021]

[5] Volker Wittpahl (2019): „Künstliche Intelligenz Technolog | Anwendung | Gesellschaft“ https://link.springer.com/content/pdf/10.1007%2F978-3-662-58042-4.pdf [Aufgerufen am 11.01.2021]

[6] Google, TensorFlow (2021): https://www.tensorflow.org/ [Aufgerufen am 22.02.2021]

[7] Google, MuZero (2020) https://www.nature.com/articles/s41586-020-03051-4.epdf [Aufgerufen am 20.09.2021]

[8] Diederik P. Kingma, Jimmy Lei Ba (2017) „Adam: a method for stochastic optimization“ https://arxiv.org/pdf/1412.6980.pdf [Aufgerufen am 04.10.2021]

[9] Fenglei Fan, Wenxiang Cong, Ge Wang (2017): “General Backpropagation Algorithm for Training Second-order Neural Networks” https://arxiv.org/ftp/arxiv/papers/1708/1708.06243.pdf [Aufgerufen am 04.10.2021]

[10] Hans im Glück Verlag, „Carcassonne Spielregeln“: https://www.hans-im-glueck.de/_Resources/Persistent/3581b911bf5113e36646fe91e1f4b47ee03f67b0/CarcBB6_Rule_D_Final_2_Web.pdf [Aufgerufen am 11.10.2021]