Fehlerhafte Rewardfunktionen: Wie kann sichergestellt werden, dass die Rewardmaximierung mit den Entwicklerinteressen übereinstimmt - bezogen auf das Tischkicker Projekt der Hochschule Darmstadt

Dennis Netzer

Matrikelnummer: 764885
Hochschule Darmstadt

Abstract

Im aktuellen Projekt der Hochschule Darmstadt soll eine künstliche Intelligenz entwickelt werden, welche die Stangen einer Mannschaft eines Tischkickers steuert und in der Lage ist gegen menschliche Spieler zu bestehen. Im aktuellen Stand des Projektes wird ein Agent auf Basis von Reinforcement Learning mit Hilfe des Double-Deep-Q-Network Algorithmus für das Abwehren eines Schusses mit dem Torwart trainiert. Die aktuelle implementierte Reward Funktion für den Agenten des Torwarts birgt Gefahren der Strategieentwicklung im Bereich des Reward Hacking. Als weitere Lösungsansätze für die Entwicklung eines intelligenten Agenten werden das Behavioral Cloning und das Inverse Reinforcement Learning des Bereichs Imitation Learning analysiert. Dieses Paper zeigt, dass die aktuell verwendete Reward Funktion des Torwartagenten im späteren Verlauf des Projektes geändert werden muss, da sonst unerwünschte Verhaltensweisen entstehen können. Für das aktuell betrachtete Teilproblem der Torwartsteuerung erweist sich die bisherige Lernweise aber als sinnvoll und der trainierte Agent kann weiterverwendet werden. Bei der Aufarbeitung der untersuchten Ansätze wurde auch das Tischkickerprojekt als Ganzes betrachtet. Es wird gezeigt, dass das untersuchte Problem des Reward Hacking für das Gesamtprojekt eine marginale Rolle spielt. Des Weiteren wird erläutert, wie mit Hilfe des Behavioral Cloning der Trainingsprozess für eine künstliche Intelligenz, die in der Lage ist kooperative Strategien für die Stangen zu entwickeln, realisiert und im Vergleich zum bisher verwendeten Q-Learning beschleunigt werden kann. Das verwendete Q-Learning kann im Anschluss an das Behavioral Cloning verwendet werden, um die künstliche Intelligenz weiter zu verbessern. Es werden außerdem die Vorteile die Inverse Reinforcement Learning bei der Entwicklung einer geeigneten Rewardfunktion mit sich bringt aufgezeigt und erklärt warum dieses Verfahren für den Tischkicker als wenig sinnvoll erachtet wird.

1 Einleitung

In einem aktuell laufenden Projekt der Hochschule Darmstadt am Fachbereich Informatik wird eine Künstliche Intelligenz (KI) entwickelt, welche die Stangen eines Teams eines Tischkickers steuert. Die KI wertet Kamera- und Sensordaten auf einem realen Tischkicker aus und trainiert so mit Hilfe von Reinforcement Learning einen KI-Agenten. Des Weiteren steht eine Simulierte Umgebung über Unity zur Verfügung mit der der Agent ebenfalls trainiert werden kann. Dem Agenten steht ein Aktionsraum zur Verfügung mit gewissen Handlungen die er vollziehen kann. Je nach Umweltzustand entscheidet sich der Agent eine der Aktionen des Aktionsraumes auszuführen. Um nun bewerten zu können, ob die Aktion bei der Zielerreichung hilfreich oder störend war, wird dem Agent eine positive oder negative Rückmeldung in Form eines Rewards gegeben. Für die Bestimmung des Rewards werden Rewardfunktionen verwendet. Im aktuellen Status des Hochschulprojektes wird die KI zum Steuern des Torwarts verwendet. Dazu kann die KI des Torwartes 21 disjunkte Positionen anfahren. Diese 21 Positionen stellen den Aktionsraum dar. Die Umweltfaktoren, welche analysiert werden, sind der Geschwindigkeits- und der Richtungsvektor des Balles. Für das Trainieren des KI-Agenten wird aktuell der Double-Deep-Q-Network Algorithmus verwendet. Dabei wird einem künstlichen neuronalen Netz der aktuelle Zustand der Umwelt als Input übergeben. Das neuronale Netz lernt nun, welche Aktionen in

welchem Zustand am besten sind, um das Ziel der Rewardmaximierung zu erreichen. Die KI bekommt einen positiven Reward von 100 bei Berühren des Balles, einen neutralen Reward von 0 für das Bewegen des Torwarts und einen negativen Reward von -50 bei einem Tor gegen das Team. Die aktuelle Herangehensweise ist es also eigenständig eine Rewardfunktion zu definieren, anhand deren die KI ihre Aktionen bewerten kann. Für diese spezifische Rewardfunktion können im späteren Verlauf Probleme auftreten. Beispielsweise könnte die KI versuchen den Ball seitlich gegen die Wand zu schießen oder zwischen sich und der Wand einzuklemmen, um den erhaltenen positiven Reward bei Berühren des Balles zu maximieren. Dieses Verhalten ist nicht wünschenswert und verfolgt nicht das Ziel der Entwickler. Dieses Problem hat in der Literatur keinen eindeutigen Namen. Es wird von Value-Alignment-Problem, Reward-Hacking, Reward Corruption Problem oder Faulty-Reward-Function gesprochen. Dieses Paper soll nun weitere Gebiete des Reinforcement Learning untersuchen mit Hilfe derer diese Probleme verhindert werden können. Die gefundenen Erkenntnisse der Analyse der Verfahren werden anschließend dazu genutzt eine Einschätzung abzugeben, ob eine Anwendung für das Tischkickerprojekt sinnvoll ist. Dabei wird sowohl das Teilproblem der Schussabwehr des Torwarts als auch das Tischkickerprojekt als Ganzes betrachtet.

2 Theoretische Grundlagen zur aktuellen Implementierung

2.1 Q-Learning

Um Reinforcement Learning mit Q-Learning Algorithmen besser verstehen zu können, müssen erst ein paar Definitionen erfolgen. [1] Der Agent kann in dem Zustand der Umgebung zu Zeitpunkt t eine Aktion $a_t \in A$ ausführen. Diese Aktion a_t ändert den Zustand $s_t \in S$ der Umwelt in den Zustand $s_{t+1} \in S$. Nach jeder Aktion oder Folge von Aktionen erhält der Agent außerdem einen Reward $r_{t+1}(s_t, a_t)$. Dieser Reward ist numerisch und kann positiv, negativ oder neutral sein. Ziel des Agenten ist es nun den kumulierten Reward in Gleichung (1) zu maximieren. Um diesen kumulierten Reward bei endlosen Interaktionen zwischen dem Agenten und der Umwelt ($T = \infty$), wie es bei kontinuierlichen Sachverhalten der Fall ist, nicht unendlich groß werden zu lassen, wird ein Diskontierungsfaktor eingeführt. Dieser sorgt dafür, dass später erwartete Rewards weniger stark gewichtet werden. Der Diskontierungsfaktor ist also definiert als $\gamma \in [0, 1]$ und für $T = \infty, \gamma \in [0, 1)$.

$$R = \sum_{t=0}^{T} \gamma^t * r_{t+1} \tag{1}$$

Des Weiteren verfolgt der Agent eine Strategie π, auf Basis derer der Agent entscheidet, welche Aktion a_t in einem Zustand s_t durchgeführt wird. Die Strategie kann daher auch als eine Wahrscheinlichkeitsverteilung $\pi : p(A = a|S)$ gesehen werden. Ziel ist es nun eine Strategie zu finden, welche den kumulierten Reward R maximiert. Diese Strategie wird als optimale Strategie $\pi*$ bezeichnet.

$$\pi* = \underset{\pi}{\operatorname{argmax}}\, \mathbf{E}(R|\pi) \tag{2}$$

An dieser Stelle kann nun eine Unterteilung der Reinforcement Algorithmen stattfinden. [2] Die Q-Learning Algorithmen gehören zu den wertebasierten Algorithmen. Es wird versucht, wie bisher erläutert, den kumulierten Reward für jeden gegebenen Zustand zu maximierten. Mathematisch lässt sich diese sogenannte Zustands-Werte-Funktion durch Gleichung (3) darstellen. Eine weitere Gleichung, welche den erwarteten kumulierten Reward in Abhängigkeit sowohl des Zustandes als auch der Aktion zum Zeitpunkt t darstellt ist die Aktion-Werte Funktion (4). Es wird also in Abhängigkeit eines Aktion-Zustand-Paares der kumulierte Reward R errechnet der unter Befolgung der Strategie π erreicht werden kann. Andere Algorithmen, die versuchen die Strategie direkt zu lernen werden strategiebasierte Algorithmen genannt.

$$V^{\pi}(s_t) = \mathbf{E}[R|s_t, \pi] \tag{3}$$

$$Q^{\pi}(s_t, a_t) = \mathbf{E}[R|s_t, a_t, \pi] \tag{4}$$

Da es nicht möglich ist die Gleichungen (3,4) zu maximieren, ohne die genauen Dynamiken der Umgebung zu kennen, werden die Gleichungen als Bellman-Gleichungen dargestellt (5). Mit Hilfe dieser ist es möglich, die aktuellen Q-Werte iterativ den optimalen Q-Werten, für welche der Agent den erwarteten Reward maximierten kann, anzunähern. [3]

$$Q^{\pi}(s_t, a_t) = \mathbf{E}[r_{t+1} + \gamma * Q^{\pi}(s_{t+1}, \pi(s_{t+1})] \tag{5}$$

Es ist in den meisten Anwendungsfällen unmöglich alle Q-Werte für alle Aktion-Zustand-Paare in einer Tabelle zu speichern und diese laufend anzupassen. [3] Daher werden bei dem Deep Q-Network (DQN) neuronale Netze verwendet, um diese Q-Werte zu ermitteln. [4]

Dafür wird dem neuronalen Netz der Zustand der Umwelt übergeben. Das neuronale Netz ermittelt nun auf Basis dieses Umweltzustandes die Q-Werte aller durchführbaren Aktionen. Um das neuronale Netz nun trainieren zu können wird ein Target-Q-Wert benötigt. Der Target-Q-Wert ist durch Gleichung 6 definiert.

$$target = r + \gamma \max_{a'} Q(s', a', \theta_i^-) \tag{6}$$

Der vorhergesagte Q-Wert ist nun einfach der Output des Netzwerkes. Mit dem vorhergesagten und dem Target Wert kann nun die Loss-Funktion erstellt werden, die für das Training minimiert wird. Dabei sind θ_i, die Gewichte des Netzwerkes bei Iteration i und θ^- die Gewichte des Netzwerkes mit welchem die Target Q-Werte berechnet wurden.

$$L = \mathbf{E}[||(r + \gamma \max_{a'} * Q(s', a', \theta_i^-) - Q(s, a, \theta_i)||^2] \tag{7}$$

Wird das Netzwerk nun laufend mit dieser Verlustfunktion angepasst, kann es zu instabilem Training oder sogar zu Divergenz kommen. [4] Dies liegt zum einen an der bestehenden Korrelation der sequentiellen Observationen der Zustände. Da sich durch das Updaten der geschätzten Q-Werte die Policy ändert, ändern sich auch die Target Q-Werte, dadurch gibt es zum anderen auch dort eine Korrelation. Um die Korrelation der Observationen zu beheben wird das sogenannte Experience Replay eingesetzt. Dabei werden beim Training immer wieder zufällig bereits gespeicherte Daten aus einer Experience Replay Datenbank geladen und für das Training verwendet. Die Korrelation der Target Q-Werte und der geschätzten Q-Werten wird durch ein periodisiertes Updaten der Target Q-Werte verhindert.

Dieser Algorithmus wurde etwas später zum Double-Deep-Q-Network (DDQN) erweitert. [5] Beim DQN Algorithmus wird dasselbe Netzwerk sowohl zur Entscheidung, welche Aktion ausgeführt werden soll, als auch zur Evaluation, wie gut diese Aktion war, verwendet. Dies führt zur häufigeren Selektion von überbewerteten Aktionen. Daher wird beim DDQN Algorithmus ein anderes Netzwerk für die Evaluation, als für die Selektion der Aktion verwendet. Die Selektion der Aktion wird weiterhin vom Online-Netzwerk ausgeführt. Die Evaluation wird nun von dem Netzwerk durchgeführt, mit welchem die Target Q-Werte berechnet werden, welches nur periodisch geupdated wird. Aktuell im Tischkicker Projekt wird dieser DDQN Algorithmus für die Steuerung des Torwartes verwendet.

2.2 Probleme des Reward Hackings

Wie wir auf Basis des letzten Kapitels nun gesehen haben, ergibt sich bei dieser Art des Trainings die Strategie anhand derer der Agent Entscheidungen trifft, durch das Maximieren der Aktions-Werte Funktion. Das bedeutet der Agent ist lediglich daran interessiert den kumulierten Reward so groß wie möglich werden zu lassen. Da die Rewards in dieser Art des Trainings vom Programmierer fest vorgegebenen werden, kann der Agent eine komplett eigene Strategie entwickeln. Das bedeutet die Verhaltensweisen des Agenten können in keiner Weise vom Entwickler kontrolliert oder vorgegeben werden.

In dem Paper [6] von 2019 hat OpenAI ein Spiel gebaut, in dem zwei Agenten gegeneinander Hide And Seek spielen. Es gibt jeweils zwischen einem und drei Hiders und Seekers. Auf der Spielfläche sind außerdem Boxen und Rampen, die sowohl von beiden Seiten verschoben als auch für die andere Seite gesperrt werden können. Die Seeker bekommen einen Reward von 1, wenn sie die Hiders finden und einen Reward von -1, wenn sie das nicht tun. Die Hiders bekommen genau andersherum, einen Reward von 1 bei erfolgreichem Verstecken und einen Reward von -1, wenn sie gefunden werden. Ein Spiel dauert 240 Zeiteinheiten, welche angepasst werden können. Die Hiders bekommen außerdem einen Zeitvorsprung, in welchem sie sich verstecken können. Die Forscher haben vorhergesagt, dass der letzte erreichbare Zustand der ist, in dem die Hiders die Rampen für die Seekers sperren und sich mit den Boxen ein versteck bauen. Allerdings hat der Agent Strategien entwickelt, die von den Entwicklern nicht gewollt waren. Zum Beispiel konnten die Seeker bewegbare Boxen zu den Rampen bewegen, auf die Box springen und anschließend mit der Box zu dem Versteck der Hiders ”surfen”. Das heißt die Seeker konnten sich bewegen ohne den Boden zu berühren, was so von den Entwicklern nicht gewollt war. Auch hier gab es keine Möglichkeit die Strategie des Agenten zu bewerten oder zu verändern. Die einzige Interaktionsmöglichkeit ist es den Reward der Agenten anzupassen. Dieser Sachverhalt wird in dem Paper [7] als Reward Hacking beschrieben. Der Agent erreicht durch Verhalten, welches vom Entwickler zwar nicht gewünscht, aber durch die Umgebung zulässig ist, eine Maximierung des Rewards. Ein weiteres Beispiel, dass in diesem Paper genannt wird ist ein Putzroboter, der absichtlich Dinge kaputt macht oder seinen bereits gesammelten Schmutz wieder ausschüttet, um einen höheren Reward für das Sammeln

von Schmutz zu bekommen. Andere Paper sprechen bei diesen Sachverhalten vom Reward Corruption Problem [8] oder vom Value-Alignment-Problem [9].

Wenn also der Agent des Torwarts im späteren Verlauf des Projektes für ein komplettes Spiel verwendet wird, kann es zu unerwünschtem Verhalten kommen. Der Torwart könnte, um seinen Reward zu maximieren versuchen den Ball zwischen sich und der Wand einzuklemmen, um den Reward den er bei Berühren des Balles erhält immer wieder zu triggern.

3 Imitation Learning

3.1 Einführung

Wie im vorherigen Kapitel erläutert wurde, kann die manuelle Definition von Rewards zu Problemen bei der Strategieentwicklung des Agenten führen. Unerwünschte Verhaltensweisen können auftreten. Ein weiteres Problem, welches im Zusammenhang mit den bisher vorgestellten Reinforcement Learning Algorithmen auftritt ist der enorme Trainingsbedarf der schon bei für den Menschen unkomplizierten Aufgaben notwendig ist. [10] Die Agenten beim Q-Learning müssen zu Beginn des Trainings zufällig Aktionen ausführen und hoffen, dass diese Aktionen einen positiven Reward einbringen. Dieses Verhalten wird auch als Exploration bezeichnet und ist mit hohem Zeitaufwand verbunden.

Der Begriff Imitation Learning beschreibt allgemein das Gebiet in welchem mit Hilfe eines Experten oder Lehrers Wissen und Fähigkeiten erlernt werden. Für den Machine Learning Kontext bedeutet dies, dass der Aspekt des Supervised Learning mit in das Training mit eingebracht werden kann, da die Aktionen des Agenten anhand von Expertenmetriken bewertet werden können. Der gesamte Bereich des Imitation Learning für Machine Learning Anwendungen lässt sich in mehrere Teilbereiche unterteilen, von welchen in diesem Paper zwei betrachtet werden. Dabei wird deren Eignung für das Tischkicker Projekt analysiert.

Im ersten Bereich des Behavioral Cloning versucht der Agent exakt das Verhalten des Experten nachzuahmen und so seine Strategie zu lernen. Dabei werden dem Agenten Beispiele gezeigt, wie sich der Experte in den verschiedenen Situationen verhält. Das heißt es werden ohne Rewards Aktionen ausgewählt, welche der Experte in der Vergangenheit in gegebenen Umweltzustand auch ausgeführt hat.

Im zweiten Bereich des Inverse Reinforcement Learning wird dem Agenten ebenfalls optimales Verhalten gezeigt. Die Idee in diesem Fall ist es allerdings, die Rewardfunktion des Experten zu erlernen. Statt zu jedem Umweltzustand einfach die zugehörige Aktion des Experten auswendig zu lernen, versucht der Agent nun zu erlernen, welche Rewards hinter den Aktionen stecken. Es wird zuerst versucht die passenden Rewards basierend auf dem optimalen Expertenverhalten zu rekonstruieren. Anschließend werden die ermittelten Rewards dazu verwendet eine eigene Strategie des Agenten zu entwickeln. Diese zwei Schritte erfolgen iterativ. [11]

Mathematisch kann das Ziel des Imitation Learning folgend definiert werden. Der Agent soll eine Strategie π_θ finden, welche ähnlich gute Ergebnisse liefert, wie die Strategie π_E eines Experten mit einer unbekannte Rewardfunktion $R(s, a)$. [12]

3.2 Behavioral Cloning

Beim Behavioral Cloning werden dem Agenten Demonstrationen zur Verfügung gestellt, welche aus einem Zustand-Aktions Paar bestehen (s_i, a_i). Die Menge der Zustand-Aktions Paare werden in einer geeigneten Datenstruktur D gespeichert. Wird nun die Strategie des Agenten als π_θ bezeichnet kann das Ziel des Agenten als Minimierungsproblem dargestellt werden (8). [13]

$$\min_{\theta} \sum_{(s_i,a_i)} ||a_i - \pi_\theta(s_i)||^2 \tag{8}$$

Das bedeutet es wird versucht den quadratischen Abstand zwischen der Aktion des Experten und der Aktion des Agenten, welche auf Basis der Strategie π_θ getroffen wurde, zu minimieren. Dadurch wird der Agent so trainiert, dass er versucht vorherzusagen welche Aktion der Experte in einem Zustand s_i treffen würde. Dieser Sachverhalt erinnert stark an die Probleme die aus dem Supervised Learning bekannt sind. Ein Modell liefert einen Wert, welcher mit einem vorhandenen Target Wert verglichen wird. Darauf hin werden die Parameter des Modells angepasst. Daher kann diese Art des Lernens auch als Supervised Klassifikationsproblem betrachtet und mit entsprechenden Algorithmen gelöst werden. [10]

Da der Agent nun das Verhalten des Experten reproduziert, kann es zu keinen vom Entwickler unerwünschten Verhaltensweisen des Agenten kommen. Dem Agenten werden keine festen Rewards übergeben, welche er für bestimmte Aktionen bekommt, um eine eigene Strategie zu entwickeln. Somit kann die Problematik des Reward-Hackings umgangen werden. Wie allerdings durch die Gleichung (8) auch zu erkennen ist werden Aktionen des Experten zu jedem Zeitpunkt t_i

benötigt. Liegt zu einem Zustand s_i keine entsprechende Aktion a_i des Experten vor, ist der Agent nicht in der Lage die Verhaltensweise des Experten nachzuahmen und entsprechend ähnliche Aktionen durchzuführen. Auf weitere Problematiken und Vorteile wird genauer in Kapitel 4 eingegangen.

Aktuell besteht der Aktionsraum des Torwart Agenten aus 21 Aktionen, welche dem Anfahren von 21 disjunkten Positionen entspricht. Für das Training des Torwarts mit Behavioral Cloning bedeutet dies, dass menschliche Spiele aufgenommen werden müssen. Bei diesen Spielen wird die Position des Torwartes zusammen mit den Inputdaten aus der Umgebung als Demonstrationen abgespeichert. Das bedeutet dem Geschwindigkeits- und Richtungsvektor des Balles wird jeweils die Position des Torwarts zugeordnet an die der menschliche Spieler den Torwart bewegt. So entstehen die benötigten Zustand-Aktions Paare, welche als Trainingsdaten verwendet werden können. Wird nun der Torwart von dem Agenten gesteuert, vergleicht er seine getroffene Aktion mit den Aktionen die der Experte bei den bestehenden Inputdaten getroffen hat. Der Agent gleicht so über die Zeit seine Strategie der des Experten an. Um zu ermitteln welche Aktion ausgeführt werden soll, kann ein beliebiger Klassifikationsalgorithmus aus dem Bereich des Supervised Learning verwendet werden. Da bereits ein Neuronales Netz besteht, welches für die Inputdaten Richtungs- und Geschwindigkeitsvektor des Balles Werte für die Aktionen des Agenten, bestehenden aus den 21 möglichen Positionen des Torwarts, ermittelt, könnte dieses Netzwerk auch für das Behavioral Cloning verwendet werden. Der Unterschied liegt nur an den Target Werten die für das Training des Netzwerkes verwendet werden. Bei der aktuellen Implementierung werden von dem Netzwerk die Q-Werte der Aktionen berechnet welche mit den Target Werten aus Gleichung (6) verglichen werden. Statt Q-Werten errechnet das Netzwerk nun eine Wahrscheinlichkeit, welche Aktion am ehesten zu der Aktion passt, die der Experte ausgeführt hätte. Das bedeutet die Aktivierungsfunktion der Outputschicht muss ebenfalls angepasst werden. Die Target Werte mit welchen das Netzwerk nun lernt bestehen aus den Zustand-Aktions Paaren, die bei der Erhebung der Trainingsdaten gespeichert wurden. Die Aktion die der Experte ausgeführt hat erhält den Target Wert 1 und alle anderen Aktionen den Target Wert 0. So kann mit Gleichung 8 und herkömmlichen Algorithmen wie Gradient Descent das Netzwerk trainiert werden.

Da langfristig ein komplettes Spiel stattfinden soll, wird dieser Sachverhalt auch analysiert. In dem Paper [14] versuchen die Autoren ein Zusammenspiel mehrerer Agenten bei einem virtuellen 3D Fußballspiel von Robotern für die RoboCup Soccer Simulation 3D League durch Behavioral Cloning zu erreichen. Konkreter wird versucht eine Situation zu erlernen bei der die Roboter angreifende Gegner verteidigen und ein Tor verhindern müssen. Da dieser Sachverhalt dem Tischkickerprojekt der Hochschule ähnelt werden die gewonnen Erkenntnisse hier vorgestellt. Die vier Roboter auf dem Spielfeld sollen jeweils von einem eigenen Agenten gesteuert werden. Das entspricht einem eigenen Agenten für jede Stange des Tischkickers. Die Roboter müssen zusammen agieren, um erfolgreich zu sein. Das gleiche trifft auch auf die Agenten des Tischkickers zu.

Zu Beginn haben die Entwickler die Roboter von menschlichen Experten über Controller steuern lassen. Jeder Roboter auf dem Feld wurde somit von einem eigenen Experten gesteuert. Während die Experten gegeneinander gespielt haben wurden Daten gesammelt mit welchen den Roboter Agenten beigebracht werden konnte, wie sie sich zu verhalten haben, um eine koordinierte Strategie zu entwickeln. Wie bisher erläutert bestehen die gesammelten Daten aus den Zustand-Aktions Paaren der Experten. Genaue technische Umsetzungen können dem Paper entnommen werden. Die Entwickler haben außerdem versucht einen manuell erstellen Agenten zu entwickeln, der als Vergleich dienen soll. Das Erstellen eines solchen manuellen Agenten wird von den Autoren als äußerst schwierig bezeichnet, da extrem viel Domänenwissen über alle möglichen Zustände und Aktionen notwendig ist. Ein Agent, welcher mit herkömmlichen Reinforcement Learning Algorithmen wie dem Q-Learning lernt wird von den Autoren als zu rechenintensiv und zeitaufwendig beschrieben. Dazu käme Schwierigkeit eine sinnvolle Rewardfunktion für jeden einzelnen Agenten zu definieren. Hierzu wird wie für den manuellen Agenten sehr viel Domänenwissen benötigt. Nachdem die Entwickler genug Experten Demonstrationen gesammelt haben wurden vier verschiedene Klassifikationsalgorithmen auf den Daten trainiert. Die Performance der einzelnen Agenten wurde anhand der durchschnittlichen Tore pro Spiel gemessen. Die Ergebnisse sind in Abbildung 1 dargestellt. Dabei wurde der Imitating Agent mit dem Klassifikationsalgorithmus trainiert der von den vier getesteten Algorithmen das beste Ergebnis erzielt hat. Wie an den Ergebnissen zu erkennen ist hat der menschlich programmierte Agent am schlechtesten performt, während der Imitating

Type of agent	No. of goals scored in 10 matches	Avg. number of goals per match
Human-controlled agent	22 (2,3,2,3,2,2,0,3,1,4)	2.2
Imitating agent	22 (3,2,2,2,3,1,5,1,1,2)	2.2
Hand-coded agent	31 (3,2,4,1,2,3,4,3,3,6)	3.1

Figure 1. Tore des gegnerischen Teams [14]

Agent dieselben Ergebnisse wie der Experte erzielen konnte. Des Weiteren wurde verglichen wie ähnlich die Strategie des Imitating Agenten und des Experten war. Dafür wurden mehrere Verhaltensweisen definiert wie beispielsweise "Face Ball". Dann wurde gemessen wie viel Prozent der Zeit diese Verhaltensweisen in den Spielen aufgetreten sind. Zu erkennen war, dass der Imitating Agent die Strategie des Experten übernommen hat, da die Verhaltensweisen in etwa für die gleiche Zeitdauer aufgetreten sind. Demnach hat das Paper gute Grundlagen dazu geleistet, dass das angewandte Behavioral Cloning Experten nahe Ergebnisse auch bei komplexeren kooperativen Aufgaben erzielen kann.

3.3 Inverse Reinforcement Learning

Der erste Schritt des Inverse Reinforcement Learning ist identischen mit denen des Behavioral Cloning. Es werden als optimal angesehene Demonstrationen D, bestehend aus Zustand-Aktions Paaren, welche als Trainingsdaten dienen, benötigt. Auch hier muss also das Verhalten eines Experten unter einer optimalen Strategie π_E in geeignetem Format gespeichert werden, welches für das Verfahren genutzt werden kann. Inverse Reinforcement Learning nimmt nun an, dass der Experte versucht eine unbekannte Reward Funktion zu maximieren. Die Aufgabe des Agenten ist es also die Rewardfunktion des Experten R_E mit Hilfe der Demonstrationen zu lernen. Während beim Q-Learning in Gleichung (1) die Rewardfunktion, welche maximiert werden soll durch die festgelegten Rewards bekannt ist, muss diese beim Inverse Reinforcement Learning erlernt werden. Dadurch wird es ermöglicht Agenten demnach auch dort zu trainieren, wo eine manuelle Definition von Rewards schwierig oder unmöglich ist. Daher hat dieser Forschungsbereich den Anwendungsbereich des Reinforcement Learning in den letzten 20 Jahren stark erweitert.

Zur Ermittlung der Rewardfunktion R^* des Agenten, welche sich der des Experten annähern soll, gibt es in der Literatur eine Reihe von Algorithmen, die sich diesem Problem zugewandt haben. Die ersten die sich diesem Problem gewidmet haben waren Andrew Y. Ng und Stuart Russel. [11] Sie haben erkannt, dass die Rewardfunktion in manchen Sachverhalten schwer manuell zu definieren ist. Beispielsweise beim Autofahren gibt es zu viele Faktoren die eine gute Fahrweise ausmachen, als dass von Entwickeln eine entsprechenden Rewardfunktion definiert werden könnte, auch wenn dieses optimale Fahrverhalten bekannt und von den Entwicklern selber durchführbar ist. Trotzdem ist es wichtig eine Rewardfunktion zu definieren damit der Agent auch in noch nicht beobachteten Zuständen sinnvoll agieren kann und seine möglichen Aktionen eigenständig bewerten kann. Um die unbekannte Rewardfunktion ermitteln zu können, müssen für diese bestimmte strukturelle Annahmen getroffen werden, welche es ermöglichen einen Algorithmus aufzustellen. Die beiden Autoren verwenden dazu eine Linearkombination der Inputfeatures der Umwelt, wie in Gleichung (9) gezeigt.

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + ... + \alpha_d\phi_d(s) \quad (9)$$

Dabei sind $\phi_1 - \phi_d$ Basisfunktionen welche die Inputfeatures nach R abbilden und zu Beginn festgelegt werden. $\alpha_1 - \alpha_d$ sind dabei die Parameter, die versucht werden zu optimieren. Seit dem Paper von Andrew Y. Ng und Stuart Russel sind zahlreiche weitere Algorithmen entstanden, die versuchen eine Rewardfunktion zu rekonstruieren, ohne dass sich die Performance eines einzelnen Algorithmus dabei stark von den anderen abhebt. Ziel des Papers ist es nicht verschiedene Inverse Reinforcement Learning Algorithmus miteinander zu vergleichen und den besten für das Tischkickerprojekt zu identifizieren. Daher wird an dieser Stelle nicht weiter auf die Einzelheiten der Algorithmen eingegangen. Interessierte Leser finden in dem Paper [15] eine gute Gegenüberstellung der verschiedenen Algorithmen sowie deren Vor- und Nachteile. Die Autoren haben allerdings ein Ablaufmodell auf Basis aller untersuchten Algorithmen erstellt, welches die Idee des Inverse Reinforcement Learning perfekt darstellt.

Die allgemeine Vorgehensweise des Inverse Reinforcement Learning kann bei allen Algorithmen daher in fünf Schritte unterteilt werden. Der erste Schritt ist wie schon erläutert wurde das Erstellen der Experten Demonstrationen mit Hilfe derer der Agent lernen kann. Im zweiten Schritt muss eine Rewardfunktion mit Annahmen definiert und deren Parameter initial festgelegt werden. Dies kann beispielsweise die in Gleichung (9) dargestellte Funktion sein. Die Parameter

$\alpha_1 - \alpha_d$ werden dabei entweder zufällig oder mit Hilfe von Demonstrationen initialisiert. Im nächsten Schritt kann der Agent mit Hilfe der definierten Rewardfunktion ein herkömmliches Reinforcement Learning durchführen, um eine Strategie zu entwickeln. Mit herkömmlichem Reinforcement Learning ist hier beispielsweise das vorgestellte Q-Learning gemeint. Im vierten Schritt können mit Hilfe der Experten Demonstrationen die Parameter der Rewardfunktion optimiert werden, sodass sich die Strategie des Agenten der Strategie des Experten annähert. In Schritt fünf werden nun die Optimierungsschritte drei und vier so oft wiederholt bis die Strategie des Agenten sich der Strategie des Experten ausreichend angenähert hat. Die entstandene Rewardfunktion wird schließlich als die Rewardfunktion angesehen, die der Experte versucht hat in den Demonstrationen zu maximieren. Wie dieses Ablaufmodell zeigt hängt die entstehende Reward von den Annahmen ab, die über sie getroffen werden. Dadurch sind die Ergebnisse die von den unterschiedlichen Algorithmen erzielt werden, je nach Anwendungsfall auch sehr unterschiedlich.

4 Ergebnisse

4.1 Vor- und Nachteile der vorgestellten Ansätze

4.1.1 Q-Learning

Für das Q-Learning wurde gezeigt, dass eine vordefinierte Rewardfunktion notwendig ist mit welcher der Agent seine gewählten Aktionen bewerten kann. Diese Definition einer Rewardfunktion ist in komplexen Anwendungsfällen mit großem Aktions- und Zustandsraum schwierig. Des Weiteren ist die Rewardfunktion das einzige Mittel die Strategie und entsprechend das Verhalten des Agenten zu steuern. Es hat sich in mehreren Papern immer wieder gezeigt, dass vom Entwickler nicht alle möglichen Strategien, die von dem Agenten unter der definierten Rewardfunktion entwickelt werden können, im Vorhinein durchdacht werden können. Dies birgt das Risiko, dass es auch zu unerwünschten Strategien kommt. Allerdings ist gerade dieses Entwickeln einer eigenen Strategie, auf die der Mensch nicht selber kommt der Kernpunkt vieler Anwendungsfälle. Reinforcement Learning ermöglicht es somit Maschinen den Menschen in vielen Gebieten überlegen werden zu lassen. Hierbei muss entsprechend das Risiko und der Nutzen analysiert werden.

Ein weiterer Punkt der die Anwendung oft erschwert ist der enorm hohe Zeit- und Rechenaufwand der sich beim Q-Learning ergibt. Der Agent wird zu Beginn ohne Wissen über die Folgenden seiner Aktionen in einen für ihn unbekannte Umgebung geschickt. Der Agent muss nun durch Exploration untersuchen, welche Aktionen in welchen Zuständen ihm einen positiven Reward und welche ihm einen negativen Reward bringen. Diese Exploration kann oft sehr lange dauern bis der Agent Strategien findet, die ihm bei der Problemlösung helfen.

4.1.2 Behavioral Cloning

Für die Entwickler ist es einfacher der Maschine zu zeigen was sie machen soll, als über Rewards ein gewünschtes Verhalten herbeizuführen. Somit kann verhindert werden, dass von dem Agenten eigene Strategien mit unerwünschtem oder schädlichem Verhalten entwickelt werden. Behavioral Cloning kann des Weiteren auch angewandt werden, wenn eine Definition von sinnvolles Rewards aufgrund der hohen Komplexität nicht durchführbar ist. Der große Nachteil hierbei ist es, dass die Maschine nur das menschliche Expertenverhalten nachahmt. Liegen zu einem Zustand keine Expertenaktionen vor, weiß der Agent nicht welche Aktionen vorteilhaft und welche nachteilhaft sind. Darüber hinaus kann auch der Experte Fehler machen, welche dann von der Maschine übernommen werden. Entstehen beim Sammeln der Demonstrationen Sensorfehler oder andere Messfehler, werden diese Fehler mit in die Trainingsdaten übernommen und reduzieren den Trainingserfolg des Agenten. Die Performance des Agenten beschränkt sich somit auf die Qualität und den Umfang der Demonstrationen. Außerdem hat der Agent bei dieser Art des Trainings keine Chance besser zu werden als der menschliche Experte. Wir wollen in vielen Fällen allerdings einen Agenten der auch in ungesehenen Situationen sinnvoll handeln kann.

Dennoch kann Behavioral Cloning auch in dieses Anwendungsfällen hilfreich sein. Es ist möglich einen Agenten zuerst mit Behavioral Cloning zu trainieren und ihn dann mit herkömmlichen Methoden des Reinforcement Learning, wie zum Beispiel dem Q-Learning, weiter zu verbessern, in dem die erlernte Strategie weiterverwendet wird. [10] Dies bringt die Gefahr des Reward Hacking wieder mit ein, kann den angesprochenen hohen Trainingsaufwand allerdings reduzieren, der entsteht, wenn von Anfang an ein Q-Leraning verwendet wird. Dies wurde auch bei dem bekannten Agenten von Google für das Spiel Go angewandt. [16] Dem Agenten wurden zuerst vergangene Spiele von Experten bereitgestellt. Mit Hilfe dieser Spiele

konnte der Algorithmen eine zu den Experten ähnliche Strategie entwickeln. Anschließend wurde mit Hilfe von wertebasierten Reinforcement Algorithmen dieser Agent weiter verbessert, um übermenschliche Ergebnisse zu erzielen.

Wird Behavioral Cloning für das Tischkicker Projekt eingesetzt gibt es weitere Punkte die beachtet werden müssen. Das in der Literatur bezeichnete Correspondence Problem beschreibt die unterschiedlichen Freiheitsgrade die ein Experte und die Maschine haben können. Dies entsteht, wenn der Experte beispielsweise Aktionen durchführen kann, die für die Maschine aufgrund der technischen Umsetzung nicht möglich sind. Der Agent des Torwartes ist aktuell so eingestellt, dass kein Drehen möglich ist. Das bedeutet das Drehen des menschlichen Spielers muss entsprechend rausgefiltert werden und nur die laterale Position übergeben werden. Des Weiteren kann der Agent des Torwarts nur 21 disjunkte Positionen anfahren. Der menschliche Spieler kann den Torwart weit feiner positionieren. Das bedeutet beim Erstellen der Trainingsdaten müsste die Position des menschlich gesteuerten Torwarts auf diese 21 Positionen gerundet werden. Dieser Prozess wird auch als Mapping von dem Experten Aktionsraum zu dem Agenten Aktionsraum bezeichnet. Da bei diesem Verfahren mit ungesehenen Zuständen nicht gut umgegangen werden kann, macht es Sinn die Inputdaten weiter zu Runden, um den Zustandsraum zu verkleinern. Aktuell werden bei den Geschwindigkeits- und Richtungsvektoren drei Nachkommastellen gemessen. Wird dies auf eine Nachkommastelle reduziert, reduziert sich auch der Zustandsraum stark. Für das Erheben der Demonstrationen müsste es möglich sein die Daten die aktuell für den AI Torwart erhoben werden auch für den vom Menschen gesteuerten Torwart zu erheben. Am schnellsten könnten genügend Trainingsdaten gesammelt werden, wenn zwei menschliche Spieler gegeneinander spielen und die Daten von beiden Seiten gespeichert werden. Die simulierte Umgebung die mit Hilfe von Unity erstellt wurde kann bei diesem Verfahren leider keine Anwendung finden.

4.1.3 Inverse Reinforcement Learning

Beim Inverse Reinforcement Learning wird ebenfalls versucht die Probleme des Reward Hacking und der Schwierigkeit bei der sinnvollen Definition von Rewardfunktionen zu beheben. Hierbei entstehen viele der Probleme die schon beim Behavioral Cloning erläutert wurden. Auch beim Inverse Reinforcement Learning hängt der Lernerfolg von der Qualität der Demonstrationen ab. Dieser Effekt ist nicht ganz so groß wie beim Behavioral Learning ist trotzdem ein wichtiger Erfolgsfaktor. Anders als beim Behavioral Cloning beschränkt sich die Performance des Agenten nicht auf die Performance des Experten. Der Agent übernimmt lediglich die Rewardfunktion, die der Experte versucht zu maximieren. Der Agent kann dennoch eigenen Strategien entwickeln, die besser performen als die des Experten. Da die Rewardfunktion des Experten übernommen wird, ist ein unerwünschtes Verhalten des Agenten viel geringer. Wird die Rewardfunktion aufgrund, geringem Umfang und schlechter Qualität der Demonstrationen allerdings nicht gut approximiert, kann es aufgrund der eigenständigen Entwicklung von Strategien des Agenten dennoch zu unerwünschtem Verhalten kommen.

Ein weiterer Vorteil der gegenüber des Behavioral Cloning entsteht ist der Wegfall des Correspondence Problems. Der Agent versucht nicht exakt die Aktionen des Agenten nachzuahmen, sondern versucht nur Ziel, welches der Experte verfolgt mit seinen eigenen Aktionen zu erreichen. So müssen die Bewegungsfreiheiten von dem Experten und dem Agenten nicht identisch sein.

Eine gelernte Rewardfunktion kann des Weiteren leichter auf andere Problemstellungen übertragen werden als eine gelernte Policy. [15] So mit bietet ein einmal trainierter Agent auf Basis des Inverse Reinforecment Learning eine bessere Ausgangsbasis für Agenten anderer ähnlicher Gebiete als Agenten auf Basis des Behavioral Cloning, bei welchen nur die Policy übergeben werden kann.

Ein großes Problem das auftritt ist, dass es oft mehrere Rewardfunktionen gibt, welche die Demonstrationen des Experten erklären können. Dadurch ist es mathematisch schwer eine optimale Rewardfunktion zu finden.

4.2 Empfehlung für die Tischkicker-KI

Wird der Agent des Torwarts weiter wie bisher trainiert und mit derselben Rewardfunktion später für ganze Spiele eingesetzt ist es sehr wahrscheinlich, dass der Agent eine Strategie des Bereichs Reward Hacking entwickelt. Bekommt der Agent positive Rewards für das Berühren des Balles wird der Agent nach langem Training erkennen, dass er seinen Reward maximieren kann indem er den Ball zwischen sich und der Wand einklemmt oder Pässe mit sich selber und der Wand spielt. Für den simplen Anwendungsfall, dass der Torwart nur einen einzelnen Schuss abwehren soll führt

das Q-Learning allerdings zu guten Ergebnissen. Die Definition der Rewardfunktion ist hierbei einfach und nachvollziehbar. Die erlernte Strategie des Agenten den Torwart zu der Stelle des Balles zu bewegen kann im späteren Verlauf des Projektes auch als Grundlage für einen weiteren Agenten, der in das Gesamtspiel mit anderer Rewardfunktion integriert wird, genommen werden. Behavioral Cloning könnte an dieser Stelle die Geschwindigkeit des Trainingsprozesses deutlich erhöhen. Wäre es möglich Daten eines menschlichen Spielers bei der Schussabwehr zu speichern und als Demonstrationen für den Agenten zu verwenden, würde der Agent die Strategie den Torwart zum Ball zu bewegen schneller erlernen. Allerdings müssen hierfür viele Demonstrationen manuell erstellt werden. Bei einzelnen Schüssen auf den Torwart ist diese Arbeit sehr monoton und dauert lange. Da für das Q-Learning auch eine simulierte Umgebung zur Verfügung steht ist es für dieses Teilproblem leichter die Strategie über die Simulationen erlernen zu lassen.

Eine Anwendung des Inverse Reinforcement Learning für dieses Teilproblem des Torwart Agenten ist nicht besonders sinnvoll. Die Rewardfunktion für das Abwehren eines Schusses ist einfach zu definieren. Das Inverse Reinforcement Learning würde hier unnötige Komplexität und Rechenkapazitäten verursachen.

Schauen wir uns das Tischkickerspiel als Ganzes an, können die Erkenntnisse der RoboCup Soccer Simulation 3D League auf den Tischkicker übertragen werden. Wenn jede Reihe des Tischkickers mit einem eigenen Agenten gesteuert wird ist das Ziel der beiden Anwendungsfälle identisch. Mehrere Agenten müssen zusammen ein Tor für die eigenen Mannschaft erzielen und dabei verhindern, dass die gegnerische Mannschaft ein Tor erzielt. Die Entwickler in dem vorgestellten Paper haben von Beginn an einen Agenten auf Basis von Exploration-Methoden ausgeschlossen. Der Grund hierfür war der enorme Zeitbedarf der prognostiziert wurde für das Erlernen einer brauchbaren Strategie bei der die Agenten zusammenarbeiten, um das Ziel zu erreichen. Mit dem Verfahren des Behavioral Cloning konnten die Agenten die gleiche Performance wie die Experten erreichen. Für schnelle gute Ergebnisse beim Tischkickerprojekt wäre es also hilfreich bei einem Tischkicker, bei welchem beide Seiten von einem Menschen gesteuert werden können, Sensoren anzubringen um Demonstrationen zu sammeln. Das aufwendige Sammeln von Demonstrationen macht Tischkicker begeisterten Studierenden auf diese Weise sogar noch Spaß. Hinweise zur Umsetzung und auftretende Schwierigkeiten wurden bereits in Kapitel 4.1.2 gegeben. Die so trainierten Agenten könnten im Anschluss mit Hilfe des DDQN Algorithmus die erlernte Expertenstrategie weiter verbessern. Dafür muss eine passende Rewardfunktion definiert werden. Sinnvoll erscheint hier jedem Agenten einen positiven Reward beim Erzielen eines Tores und einen negativem bei einem gegnerischen Tor zu geben. Für die Verbesserung der Strategie kann nun auch erneut die simulierte Umgebung zum Einsatz kommen.

Die technischen Voraussetzungen für die Anwendung des Inverse Reinforcement Learning sind die gleichen wie beim Behavioral Cloning. Es werden Demonstrationen von menschlichen Spielen benötigt. Die Schwierigkeit die hierbei gesehen wird ist, dass das Verfahren versucht die Rewardfunktion von den Demonstrationen zu erlernen. Da viele Tischkickerspiele von Nicht-Profis in der Realität allerdings aus mehr Zufall als geplantem Vorgehen bestehen, besteht die Gefahr, dass dieser Ansatz sehr ineffizient ist, da eine Rewardfunktion erlernt wird, welche nicht das Tore erzielen und verteidigen im Vordergrund hat. Hier wird die Notwendigkeit von wirklichen Experten Demonstrationen höher eingeschätzt damit das Inverse Reinforcement Learning das vorgestellte Vorgehen der Kombination von Behavioral Cloning und Q-Learning schlägt. Inverse Reinforcement Learning findet dort Anwendung wo es schwer oder unmöglich ist eine Rewardfunktion eigenständig zu definieren. Dies ist beim Tischkicker nicht der Fall.

Für das Projekt als Ganzes kann es also interessant sein Behavioral Cloning genauer zu betrachten und die genauen technischen Umsetzungen zu erforschen. Einige erste Einstiegspunkte wurden hierzu in diesem Paper genannt.

References

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, second ed., 2018.

[2] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv*, no. 61573353, pp. 1–13, 2019.

[3] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves,

M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[5] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2094–2100, 2016.

[6] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," *arXiv*, 2019.

[7] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete Problems in AI Safety," pp. 1–29, 2016.

[8] T. Everitt, V. Krakovna, L. Orseau, and S. Legg, "Reinforcement learning with a corrupted reward channel," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 0, pp. 4705–4713, 2017.

[9] D. Hadfield-Menell, A. Dragan, P. Abbeel, and S. Russell, "Cooperative inverse reinforcement learning," *Advances in Neural Information Processing Systems*, no. Nips, pp. 3916–3924, 2016.

[10] Hussein, Ahmed; Gaber, Mohamed M.; Elyan, Eyad; Jayne, Chrisina, "Imitation learning: A Survey of Learning Methods," *ACM Computing Surveys*, vol. 50, no. 2, 2017.

[11] Andrew Yang Ng, Stuart Russel, "Algorithms for Inverse Reinforcement Learning," *Proceedings of the international conference on machine learning (ICML)*, vol. 99, pp. 278–287, 2000.

[12] A. Y. Ng and P. Abbeel, "Apprenticeship Learning via Inverse Reiforcement Learning," *Proceedings of the twenty-first international conference on machine learning*, 2004.

[13] V. G. Goecks, G. M. Gremillion, V. J. Lawhern, J. Valasek, and N. R. Waytowich, "Integrating behavior cloning and reinforcement learning for improved performance in dense and sparse reward environments," *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2020-May, pp. 465–473, 2020.

[14] S. Raza, S. Haider, and M. A. Williams, "Teaching coordinated strategies to soccer robots via imitation," *2012 IEEE International Conference on Robotics and Biomimetics, ROBIO 2012 - Conference Digest*, pp. 1434–1439, 2012.

[15] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *arXiv*, pp. 1–48, 2018.

[16] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.