Projektbericht - Soccer Twos RL

Eric Echtermeyer[1], *Lasse* Friedrich[2], *Benedikt* Prisett[3], and *David* Schäfer[4]

[1] *6373947*
[2] *9924680*
[3] *5709658*
[4] *7086451*

Abstract. Das Training von Reinforcement Learning (RL) Agents in simulierten Spielumgebungen ist eine verbreitete und geeignete Herangehensweise, um Reinforcement Learning (RL)-Verfahren zu erlernen und zu vergleichen. Oftmals finden diese Verfahren Anwendung in sehr einfachen Umgebungen mit einem Agenten und wenigen Beobachtungen. Eine besondere Herausforderung ist es jedoch, wenn die Umgebung einen gewissen Realitätsbezug hat und eine erweiterte Komplexität aufweist.
Im Rahmen dieser Gruppenarbeit werden verschiedene Reinforcement Learning (RL)-Verfahren für die "Soccer-Twos"-Umgebung, welche ein Fußballspiel zwischen mehreren Agents simuliert, implementiert und ausgewertet. Zum einen werden Single-Agent-Methoden betrachtet, bei denen ein Agent allein ein Tor schießen soll. Zum anderen wird das deutlich komplexere Multi Agent Reinforcement Learning (MARL) Problem angegangen, und eine theoretisches Trainingsverfahren in Form von "competitive self-play" entwickelt.
Die Evaluierung und der Vergleich mit einer einfachen Baseline-Methode ergeben, dass erste Ergebnisse ausbaufähig sind. Jedoch zeigen Algorithmen wie Proximal Policy Optimization (PPO) vielversprechende Erfolge und auch die Betrachtung der Multi-Agent-Umgebung erweist sich als wertvoll.

1 Einführung

Im Rahmen dieser Gruppenarbeit mit dem Ziel der Implementierung von erlernten Reinforcement Learning (RL) Verfahren wurde sich für das "Soccer Twos"-Problem [1, p. 15] entschieden. Hierbei handelt es sich um eine Umgebung die als Teil des "Unity Machine Learning (ML) Agents Toolkit" [1] veröffentlichte wurde. Es wird ein vereinfachtes Fußballspiel zwischen zwei Teams mit je zwei Spielern simuliert wodurch sich eine Multi Agent Reinforcement Learning (MARL) Problemstellung abzeichnet. Die Entscheidung für dieses RL-Problem wurde auf Grundlage der geigneten Anwendungsmöglichtkeit von den in der Veranstaltung behandelten Verfahren, sowie den Anspruch auf einen gewissen Realitätsbezug getroffen. Dieser ist sowohl durch die simulierte Physik der Umgebung, als auch durch die Interaktion von mehreren Akteuren gegeben.

1.1 Soccer Twos Umgebung

Das Unity ML Agents Toolkit verfügt über diverse Komponenten um RL verwandte Aspekte in Unity-Umgebungen umzusetzen und bietet zudem eine Vielzahl an Beispiel-Umgebungen und Implementierungen von RL-Algorithmen, welche es Spieleentwicklern, oder Forschern ermöglicht RL-Verfahren zu entwicklen und zu testen. Die bereitgestellten Beispielumgebungen zeigen die Einsatzmöglichkeiten des Toolkits durch kleinere Spiele auf, welche zum größtenteil in 3D-Umgebungen stattfinden. In den Umgebungen agieren blockförmige, als auch komplexere mit Gelenken ausgestattete Agenten, die alleine oder gemeinsam verschiedene Aufgaben lösen müssen. Die Beobachtungsräume dieser Agent sind primär in Vektorform dargestellt sowie in einzelnen Fällen auch visuell. Die Aktionsräume sind je nach Spiel entweder diskret oder kontinuierlich. Dies bietet somit eine Ausgangslage für die Anwedung von RL-Verfahren in unterschiedlichstem Umfang.

Da die Benutzung des gesamten Toolkits komplex ist und über mehrer Softwarelösungen hinweg erfolgt, wird das Projekt in einer ausschließlich Python-basierten Gymnasium-Umgebung umgesetzt. Hierfür gibt es ein Paket, welches eine der Umgebungen des Toolkits, extrahiert und als stand-alone Version bereitstellt [2]. Dieses Paket ermöglicht es, die komplette Implementierung und Visualisierung in Python durchzuführen und die bereits bekannten Gymnasium-Schnittstellen zu nutzen.

Bei dem Spiel "Soccer Twos" spielen zwei Agenten gemeinsam in einem Team gegen ein weiteres Zweier-Team Fußball. Es handelt sich hierbei um eine 3D-Spielumgebung, die in Abbildung 1 dargestellt ist. Das Spielfeld ist mit einer Bande begrenzt, wodurch es kein Aus gibt und das Spiel nicht unterbrochen wird. Zu Beginn eines Spiels liegt der Ball in der Mitte des Feldes, und die beiden Teams starten auf ihrer jeweiligen Seite des Spielfelds. Die Agenten sind nicht in der Lage, komplexe Schüsse auszuführen, sondern passen und schießen, indem

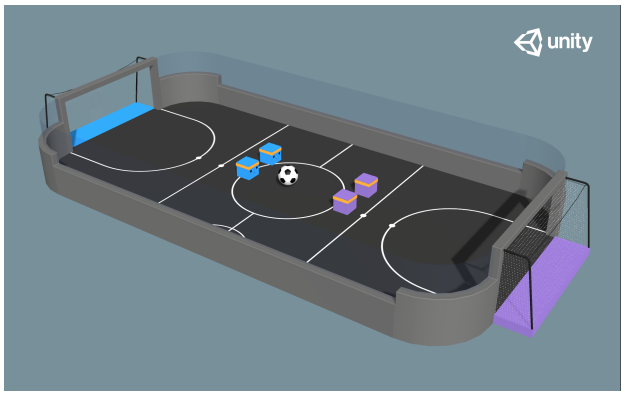

Figure 1. Die visualisierte 3D-Spielumgebung von "Soccer Twos" des Unity ML-Agents Toolkits. Die zwei Teams in Blau und Pink mit jeweils zwei blockförmigen Agenten stehen sich gegenüber, während der Fußball im Mittelkreis liegt. [1]

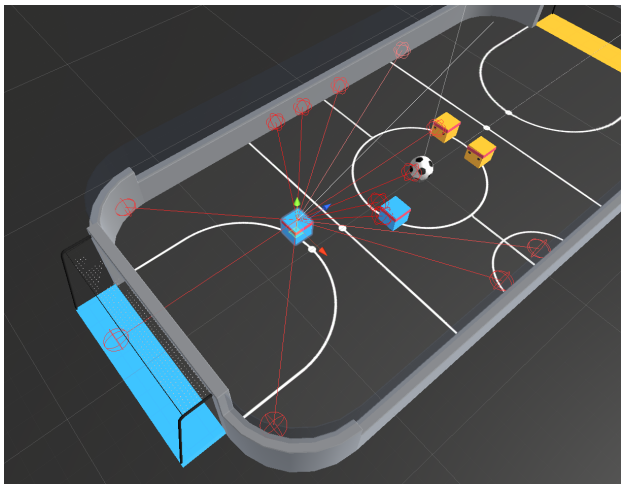

Figure 2. Eine Visualisierung des Beobachtungsraum des Agent mit den genutzten Ray-Casts. [2]

sie gegen den Ball laufen. Es ist das Ziel jedes Teams, den Ball in das gegnerische Tor zu befördern und gleichzeitig das eigene Tor zu schützen.

Der Aktionsraum eines Agents ist diskret und umfasst die drei Dimensionen "Vorgehen oder Zurückgehen", "Rechtsgehen oder Linksgehen" sowie "Linksdrehen oder Rechtsdrehen". Für jede Dimension kann ein Wert aus $\{0, 1, 2\}$ gewählt werden, wobei 0 für keine Ansteuerung auf dieser Dimension steht, 1 für die zuerst genannte Aktion und 2 für die zweite. Zum einfachen Vorwärtsgehen sieht der Aktionsvektor somit wie folgt aus: $[1, 0, 0]$ oder schräg Vorwärts und Links: $[1, 2, 0]$. Es gibt somit 27 mögliche diskrete Aktionen. Die Aktionen werden für einen Schritt des Spiels für alle Agents gleichzeitig definiert und in einem 4-elementigen Dictionary an die Umgebung übergeben.

Der Beobachtungsraum besteht aus einem Vektor mit 336 Dimensionen. Diese Dimensionen ergeben sich aus mehreren Strahlen (sogenannten Ray-Casts), die die Umgebung eines Agents abtasten. Konkret werden elf Strahlen nach vorne ausgesendet, die einen Bereich von 120° abdecken, und drei Strahlen nach hinten, die einen Bereich von 90° abdecken. Jeder dieser Strahlen kann sechs verschiedene Objekttypen erkennen und die Entfernung zu diesen Objekten messen. Um Bewegungsabfolgen zu erkennen, werden die Daten von drei aufeinanderfolgenden Beobachtungen zusammengeführt. Die Identifikation der einzelnen Objekte die getroffen werden, erfolgt über einen one-hot-encoded Vektor. Der Objekttyp des Gegners wird dabei zweimal geführt, wodurch eine Differenzierung der Gegner stattfinden kann. Insgesamt beläuft sich die Dimension des Detektionsvektor eines einzelnen Strahls damit auf die Größe acht. Die Berechnungsvorschrift lautet:

$$336 = (14 \text{ Strahlen} \cdot (7 + 1 \text{ Entfernung}) \cdot 3 \text{ Beobachtungen})$$

Jedes Spiel wird entweder für 1000 Iterationen oder bis ein Tor erzielt wird gespielt. Die Belohnungsfunktion verleiht eine positive Belohnung von $1 -$ akkumulierte Zeitstrafe, wenn ein Tor erzielt wird. Zu Beginn einer Episode wird die Zeitstrafe auf 0 gesetzt und dann bei jedem Update um $\frac{1}{\text{max Schritte}}$ erhöht. Diese Berechnung der Belohnung trägt somit dazu bei, dass vor allem das schnelle Schießen eines Tores unterstützt wird. Erhält ein Team ein Gegentor, wird eine negative Belohnung von -1 verhängt, um so die Verteidigung des eigenen Tores anzureizen.

1.2 Multi Agent Reinforcement Learning

Bei "Soccer Twos" handelt es sich um ein MARL-Problem, welches ein Teilgebiet des normalen RL darstellt. MARL befasst sich mit Umgebungen, in denen mehrere Agenten agieren und somit auch miteinander interagieren können und sich durch ihre Aktionen gegenseitig beeinflussen.

MARL-Probleme können anhand zweier Interaktionstypen zwischen den Agenten klassifiziert werden [3]. In kooperativen Umgebungen agieren mehrere Agenten, um eine gemeinsame Belohnung zu maximieren. Es ist also das Ziel, dass die Agenten zusammen eine Aufgabe lösen und sich gegenseitig unterstützen. In konfrontativen Umgebungen treten mehrere Agenten gegeneinander an und versuchen, ihre eigene Belohnung zu maximieren, während die Belohnung des Gegners minimiert wird. Weiter ist es möglich, dass diese beiden Arten auch in gemischter Form auftreten und es somit in einer Umgebung sowohl kooperative Agenten gibt, die aber konfrontativ gegenüber anderen auftreten. Genau um eine solche gemischte Umgebungsform handelt es sich auch bei dem "Soccer Twos"-Problem. Zwei Agenten eines Teams spielen gemeinsam kooperativ gegen ein anderes Team, welches einen konfrontativen Gegner darstellt.

Neben der Interaktionsart, kann eine Umgebung zudem dezentral oder zentral aufgebaut. In einer dezentralen Umgebung wird jeder Agent alleine und unabhängig von den anderen Agenten trainiert, was bedeutet, dass jeder weiterhin seine eigene Belohnungsstruktur, Beobachtun-

gen und Aktionen hat und im Lernprozess keine Informationen zwischen den Agenten geteilt werden. Dies hat den Vorteil, dass jeder Agent alleine trainiert werden kann und die anderen Agenten einfach als Teil der Umgebung wahrnimmt. Eine derartige Struktur hat jedoch den großen Nachteil, dass sie eine nicht-stationäre Umgebung erzeugt, auf die viele RL-Algorithmen angewiesen sind. In einer zentralen Umgebung hingegen gibt es eine übergeordnete Struktur, welche die Beobachtungen und Belohnungen aller Agenten sammelt und verarbeitet. Dadurch kann eine einzige gemeinsame Policy erlernt werden und das Wissen wird zwischen den Agenten geteilt [4].

2 Methoden

Zum lösen des "Soccer Twos" Problem wurden im Rahmen dieser Gruppenarbeit verschiedene, unterschiedlich komplexe, Methoden angewendet. Zunächst wurde eine einfach Baseline entwickelt mit der die späteren RL-Algorithmen verglichen werden können. Im Folgenden wurde als erstes ein alleiniger RL-Agent trainiert, mit dem Ziel ein Tor zu erziehlen. Anschließend wurde versucht dies nun auf das MARL-Problem mit dem Training mehrerer Agents zu erweitern.

2.1 Baseline

Um eine adäquate Vergleichbarkeit und Bewertung der implementierten RL-Verfahren zu ermöglichen, wurde zunächst eine einfache Baseline-Methode für das "Soccer Twos"-Problem entworfen und implementiert. Diese Baseline beruht auf den zwei Verhaltensweisen, "Verteidigungsmodus" und "Angriffsmodus", die auf Basis der von der Umgebung bereitgestellten Informationen durchgeführt werden.

Der "Verteidigungsmodus" wird ausgelöst, sobald sich der Ball auf der Heimseite des Spielfelds befindet und somit die Gefahr für ein Gegentor besteht. Die Agents versuchen in diesem Fall, das eigene Tor zu verteidigen, indem sie sich zwischen das Tor und den Ball bewegen. Dies geschieht, indem sie sich zunächst zum Mittelpunkt zwischen dem Ball und dem Tor drehen und anschließend vorwärts zu diesem Mittelpunkt laufen. Da dies wiederholt ausgeführt wird, bewegen sich die Agents somit zwischen den Ball und das Tor und wehren Schüsse auf das eigene Tor ab. Der "Angriffsmodus" wird ausgeführt, sobald der Ball auf die gegnerische Spielfeldseite gelangt und somit ein Angriff auf das gegnerische Tor stattfinden kann. Die Agents drehen sich nun so lange, bis der Winkel zwischen Spieler und Ball kleiner als 30 Grad ist, und laufen dann auf den Ball zu. Befindet sich der Ball jedoch hinter dem Spieler wird die Bewegung angepasst, sodass ein Eigentor verhindert werden kann. Die Idee des "Angriffsmodus" ist es somit, dass ein Agent alles darauf setzt, einen Schuss durchzuführen und somit ein Tor zu erzielen.

Diese simple Baseline, beruhend auf festen Regeln, ermöglicht ein einfaches Fußballspiel zwischen den Agents, welches nicht besonders komplex und effizient ist, aber ein vereinfachtes Spielgeschehen mit gelegentlichen erfolgreichen Torschüssen oder Abwehraktionen erlaubt. Das Ziel der im Folgenden entwickelten RL-Verfahren ist es nun, diese einfache Spielweise zu übertreffen.

2.2 Single Agent RL Algorithmen

Zum Einstieg wurden für das "Soccer Twos" Problem zunächst einige grundlegende RL-Algorithmen angewendet, die sich jedoch nur auf einen Agent beschränken. Es wurde somit eine Vereinfachung vorgenommen und nicht direkt das deutlich komplexere MARL Problem, bei welchem mehrere Agents trainiert werden müssen, betrachtet, um zu Beginn die allgemeine Machbarkeit und Komplexität des Trainingsprozesses zu untersuchen. Im Rahmen dieser Vereinfachung werden weiter vier Agents und ein Ball zu Start des Spiels zufällig auf dem Feld platziert, jedoch kann sich nur einer dieser Agents bewegen und wird trainiert.

Insgesamt wurden zwei Off-Policy- und zwei On-Policy-Methoden für das vereinfachte, auf einen Agent beschränkte Problem angewendet. Bei Off-Policy-Verfahren wird die Policy, die der Agent erlernt, unabhängig von jener behandelt, die das Verhalten während des Lernprozesses bestimmt. Das bedeutet, dass die gesammelten Erfahrungen nicht nur von der aktuellen Policy des Agenten abhängen, sondern auch von einer anderen Policy stammen können. Bei On-Policy-Methoden hingegen erlernt ein Agent die gleiche Policy, die auch während des Lernprozesses verwendet wird, um Aktionen zu bestimmen. Der Agent bewertet und optimiert diese Policy direkt anhand der durch diese Policy generierten Erfahrungen.

Ein typischer Ansatz zum lösen von RL-Problemen ist der **Deep Q-Network (DQN)** Algorithms. Es handelt sich hierbei um eine Off-Policy Methode welche das Q-Learning mit tiefen neuronalen Netzwerken kombiniert, um optimale Handlungsstrategien zu erlernen. Der Kern des DQN besteht darin, eine Q-Funktion zu approximieren, die den erwarteten zukünftigen Belohnungswert für eine gegebene Aktion in einem bestimmten Zustand angibt. Durch kontinuierliches Training und Anpassung der Gewichte des neuronalen Netzwerks kann der Agent somit lernen, optimale Aktionen zu wählen, die langfristig zu höheren Belohnungen führen [5].

Ein ebenfalls verbreitete Off-Policy Methode ist der Policy Gradient (PG) Algorithmus, welcher sowohl für kontinuierlichen als auch diskrete Aktionsräume geeignet ist. Bei diesem Verfahren wird direkt eine Policy-Funktion, in Form eines Neuronalen Netzes, erlernt, die angibt, welche Aktion in einem gegebenen Zustand ausgeführt werden soll. Eine Erweiterung dieser Methode stellt der Deterministic Policy Gradient (DPG) und auch **Deep Deterministic Policy Gradient (DDPG)** Algorithmus dar. Diese verwenden zwei Netzwerke: ein Actor-Netzwerk, das die Policy darstellt und Aktionen auswählt, und ein Critic-Netzwerk, das den Q-Wert der ausgewählten Aktionen schätzt. Es wird somit die Stabilität und Effizienz des Lernens verbessert und bietet vor allem bei komplexeren Aufgaben in kontinuierlichen Aktionsräumen Vorteile.

Ein weiterer bedeutender Ansatz in diesem Bereich ist die **Proximal Policy Optimization (PPO)** Methode. PPO

ist ein On-Policy Algorithms, gehört zur Familie der PG Methoden und zielt darauf ab, die Instabilitäten und hohen Varianzen traditioneller PG Algorithmen zu reduzieren. Dies wird durch die Einführung eines speziellen Clipping-Mechanismus erreicht, der die Updates der Policy innerhalb eines bestimmten Bereichs hält. Dieser Mechanismus stellt sicher, dass die Änderungen an der Policy während des Trainings nicht zu drastisch sind, wodurch ein stabilerer und effizienterer Lernprozess gewährleistet wird [6].

Ein weiterere On-Policy Methode ist der **Advantage Actor Critic (A2C)** Algorithmus. A2C kombiniert Elemente der Actor-Critic-Architektur mit Vorteilsschätzungen (Advantage-Funktion). Der Kern der A2C Methode besteht somit wie bei den vorherigen Methoden ebenfalls aus einem Policy- und Critic-Netzwerk. Zusätzlich wird jedoch noch eine Advantage-Funktion erlernt, welche den Vorteil einer Aktion in einem bestimmten Zustand im Vergleich zum durchschnittlichen erwarteten Wert aller möglichen Aktionen in diesem Zustand, bestimmt. Durch die Berechnung des Advantage-Werts kann der Algorithmus genauer die Unterschiede zwischen guten und schlechten Aktionen erfassen, was zu effizienteren Policy-Updates führt [7].

Da die Ansteuerung der Agents, wie zuvor beschrieben, über einen dreielementigen Aktionsvektor mit drei möglichen Werten für jedes Element erfolgt, wurde die Struktur der bei den RL-Algorithmen verwendeten neuronalen Netze entsprechend angepasst. Die Ausgabeschicht des Netzwerkes besteht aus neun Neuronen, die in drei Gruppen, welche jeweils eine Bewegungsdimension darstellen, aufgeteilt sind. Innerhalb jeder dieser Gruppen wird ein Softmax angewendet und je ein Neuron steht für keine Ansteuerung, die erste mögliche Aktion. Die Auswahl der letztendlich durchgeführten Aktion für dieses Element des Aktionsvektors erfolgt durch die Wahl des Maximalwertes innerhalb dieser Gruppe.

Bereits beim Training dieser ersten Algorithmen wurden einige Auffälligkeiten beobachtet. Besonders die Tatschache das in der "Soccer-Twos"-Umgebung nur relativ selten ein Tor und somit eine Belohnung für den Agenten erziehlt wird führte zu einem sehr langsamen Lernprozess und schlechten Ergebnissen. Da es sich hier somit um eine sogennante "sparse reward" Umgebung handelt, in welcher die Belohnungen selten und unregelmäßig auftreten, wurde die Behlohnungsfunktion der Agenten angepasst. Neben einer Belohung für ein erzieltes Tor erhalten die Agenten nun fortlaufend einen Belohnung welche auf ihrem Abstand zum Ball baisert. Diese neue Behlohung nimmt 1 an, wenn der Abstand zum Ball minimal ist und 0 wenn der Abstand maximal, also 32 Einheiten, groß ist. Die neue Belohnungsstruktur der Umgebung kann somit mit folgender Formel zusammengefasst werden:

$$\text{Belohnung} = \begin{cases} 1 - \text{akkumulierte Zeitstrafe} & \text{Tor} \\ -1 & \text{Gegentor} \\ \max\left(0, 1 - \frac{\text{Abstand zum Ball}}{\text{maximaler Abstand}}\right) & \text{sonst} \end{cases}$$

Durch diese Anpassung wird der Agent nun angeregt sicher schneller zum Ball zu bewegen und die Wahrscheinlichkeit das er eine sinnvolle Aktion und letztendlich ein Tor erziehlt steigt.

Eine weitere wichtige Entscheidung, die im Laufe des Entwicklungsprozesses getroffen wurde, war die Verwendung der Stable Baselines3 (SB3) Bibliothek [8] für die Implementierung der PPO und A2C Methoden. Es handelt sich hierbei um eine Open-Scource-Bibliothek welche robuste und effiziente Implementierungen einer Vielzahl an moderneren RL-Algorithmen bietet. Zu Beginn wurden die verschiedenen Verfahren zwar eigenständig implementiert, jedoch stießen diese aufgrund der hohen Komplexität schnell an ihre Grenzen. Durch die Nutzung von SB3 konnte eine korrekte Implementierung sichergestellt werden. Zudem ermöglicht die Bibliothek den Einsatz fortschrittlicher Optimierungsverfahren und unterstützt Multiprocessing während des Trainings. Dies führte zu einer erheblichen Verbesserung der Trainingszeit und der erzielten Ergebnisse.

2.3 Multi Agent Methode

Aufbauend auf den bei der Implementierung der vorherigen Methoden gewonnenen Erkenntnissen wurde im Folgenden das MARL-Problem betrachtet. Da es sich hierbei um ein großes Themenfeld mit vielen, teils sehr komplexen Methoden handelt und bereits beim Training in einfacheren Single-Agent-Umgebungen einige Herausforderungen auftraten, wurden die Untersuchungen im MARL-Bereich auf die Anwendung eines "competitive self-plays" mit dem PPO-Algorithmus beschränkt.

Bei "competitive self-play" spielt und lernt ein Agent, indem er gegen sich selbst oder eine alternative Version seiner selbst antritt. Im Kontext des "Soccer-Twos"-Problems wurde eine Population an Agenten erzeugt, die in anfangs zufällig zugeordneten Spielen gegeneinander antreten. Basierend auf diesen Spielen erhalten die Agenten ein Update zu ihrem ELO-Rating, einer Bewertung, die ihre relative Spielstärke widerspiegelt, und sammeln währenddessen individuell Erfahrungen. Nach ihren Spielen werden diese Erfahrungen genutzt, um ihre jeweilige Policy zu aktualisieren. Dieses Verfahren wird wiederholt fortgesetzt, wobei jedoch nun die Zuordnung der Spielgegner aufgrund der ELO-Ratings getroffen wird. Hierdurch kann sichergestellt werden, dass ein Agent gegen einen Gegner auf ähnlichem Niveau antritt und so durch die angemessene Herausforderung des Spiels in der Lage ist, seine eigenen Fähigkeiten iterativ zu verbessern.

3 Anwendung

Die in Kapitel 2 beschriebenen Methoden erwiesen sich bei ihrer Anwendung auf das "Soccer Tows" Problem als unterschiedliche erfolgreich.

Abbildung 3 zeigt die erzielten Belohnungen als gleitender Mittelwert von 10 Episoden der verschiedenen Algorithmen. Es ist zu erkennen, dass die beiden On-Policy-Methoden höhere Belohnungen erzielen als die gewählten

Off-Policy-Ansätze. Im Vergleich untereinander ist es eindeutig, dass der PPO-Algorithmus die höchsten Belohnungen erzielt. Jedoch ist hier, wie auch bei den anderen, leider kein wirklicher Lerneffekt zu erkennen und die Werte schwanken nur um ein gleichbleibendes Mittel. Besonders im Vergleich zu der Baseline-Methode wird klar, dass die trainierten Agents bisher noch keine guten Ergebnisse erzielen können. Zwar gibt es Phasen, in denen die Baseline schlechtere Ergebnisse hervorbringt als einige der RL-Algorithmen, jedoch ist es eindeutig, dass die Baseline im Gesamtverlauf deutlich höhere Belohnungen erzielen kann.

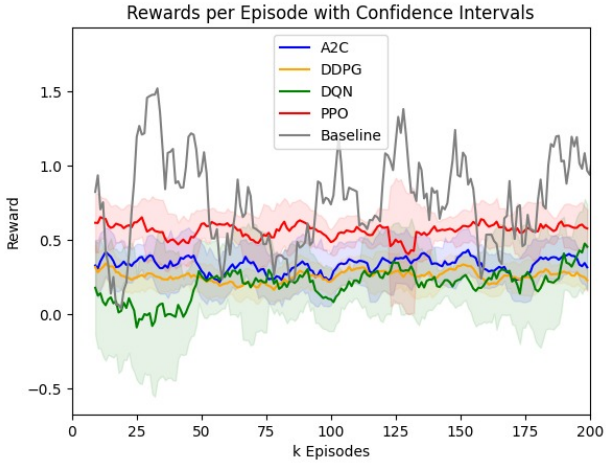

Figure 3. Belohnungen der unterschiedlichen Single-Agent-Verfahren pro Episode mit Konfidenzintervallen

Abbildung 4 stellt einen Violin-Plot der Belohnungen der letzten 10.000 Episoden für die verschiedenen Agents dar und erlaubt so einen allgemeinen Vergleich der Stärke der unterschiedlichen Agents. Auch diese Grafik bestätigt die Erkenntnis, dass der PPO-Algorithmus die besten Ergebnisse mit einem durchschnittlichen Score von 0,5 erzielen konnte. Es wird jedoch auch hier deutlich, dass, obwohl die Baseline-Methode deutlich höhere Schwankungen aufweist, diese weiterhin bessere Ergebnisse erreicht und eine durchschnittliche Belohnung von 0,7 erreicht.

Eine weitere einblicksreiche Betrachtung ist die Heatmap in Abbildung 5, welche die Bewegungen der Agents darstellen kann.

Auch die Anwendung des “competitive-self play” Verfahrens für das Multi-Agent-Szenario wurde ausgewertet.

4 Abschluss

Die in dieser Gruppenarbeit angewendeten und untersuchten Verfahren haben gezeigt, dass RL-Methoden erfolgreich eingesetzt werden können, um in einer simulierten Fußball-Umgebung, in der ein Spieler das Ziel hat, ein Tor zu schießen und auch kompetitiv gegen andere Spieler antreten kann, einen RL-Agenten zu trainieren. Besonders der PPO-Algorithmus erwies sich als erfolgreich, und auch die Anwendung von “competitive self-play” im MARL-Kontext zeigt erste Potenziale auf.

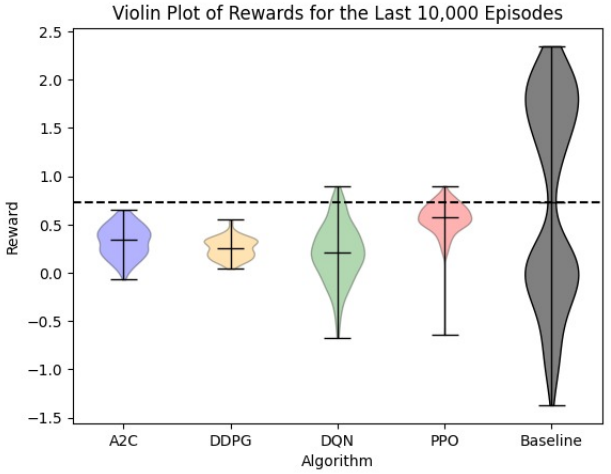

Figure 4. Violin-Plot der Belohnungen für die letzten 10.000 Episoden

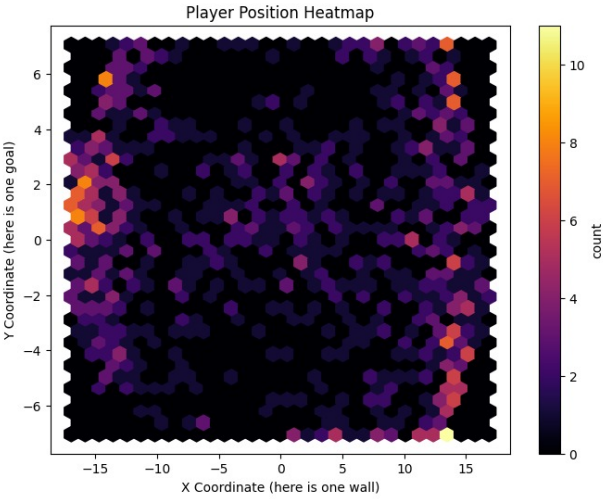

Figure 5. Heatmap

Nichtsdestotrotz muss angemerkt werden, dass das Verhalten der Agenten im Vergleich zu einem echten Fußballspiel noch zu wünschen übrig lässt. Die Anwendung der erlernten RL-Verfahren und das Training der Agenten war in dieser Gruppenarbeit aufgrund der sehr komplexen Umgebung, mit einer Vielzahl an Beobachtungen und der zusätzlichen Schwierigkeit des Multi-Agenten-Szenarios, eine große Herausforderung. Auch die verfügbaren Trainingsressourcen haben den Erfolg der Ergebnisse eingeschränkt.

Auch wenn das Ergebnis dieser Gruppenarbeit somit vom Wunschszenario eines perfekten Fußballteams aus RL-Agenten etwas entfernt ist, wurden fortschrittliche RL-Methoden implementiert und eine einblicksreiche Auswertung mit einem Vergleich der unterschiedlichen Verfahren erstellt. Da vor allem die Vereinfachung auf eine Single-Agent-Umgebung und die Anwendung des PPO-Algorithmus erste, durchaus zufriedenstellende, Ergebnisse erbringt, zeigt die Ausarbeitung Potenzial für

mögliche Erweiterungen auf. So könnte zum einen das Training dieses bestehenden erfolgreichen Single-Agent-RL-Verfahrens ausgeweitet und durch erhöhte Computing-Ressourcen verbessert werden. Zum anderen ist es auch weiter denkbar, das Multi-Agenten-Szenario tiefer zu betrachten und komplexere Verfahren wie MADDPG oder andere zu untersuchen.

References

[1] A. Juliani, V.P. Berges, E. Teng, A. Cohen, J. Harper, C. Elion, C. Goy, Y. Gao, H. Henry, M. Mattar et al., arXiv preprint arXiv:1809.02627 (2020)

[2] B. Oliveira, *A pre-compiled soccer-twos reinforcement learning environment with multi-agent gym-compatible wrappers and human-friendly visualizers.*, https://github.com/bryanoliveira/soccer-twos-env (2021)

[3] P.J.t. Hoen, K. Tuyls, L. Panait, S. Luke, J.A. La Poutré, *An Overview of Cooperative and Competitive Multiagent Learning*, in *Learning and Adaption in Multi-Agent Systems*, edited by K. Tuyls, P.J. Hoen, K. Verbeeck, S. Sen (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 1–46, ISBN 978-3-540-33059-2

[4] M. Tan, *Multi-Agent Reinforcement Learning: Independent versus Cooperative Agents*, in *Proceedings of the Tenth International Conference on Machine Learning (ICML 1993)* (Morgan Kauffman, San Francisco, CA, USA, 1993), pp. 330–337, ISBN 1-55860-307-7, http://web.media.mit.edu/~cynthiab/Readings/tan-MAS-reinfLearn.pdf

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, *Playing atari with deep reinforcement learning* (2013), 1312.5602, https://arxiv.org/abs/1312.5602

[6] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, CoRR **abs/1707.06347** (2017), 1707.06347

[7] V. Mnih, A.P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, CoRR **abs/1602.01783** (2016), 1602.01783

[8] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, N. Dormann, Journal of Machine Learning Research **22**, 1 (2021)