DHBW Mannheim

Hausarbeit

Implementation eines Reinforcement Learning Agents im Videospiel Doom

Projekteinreichung: Machine Learning

Dozent: Prof. Dr. Maximilian Scherer

Vorlesung: Advanced Applied Machine Learning

Kursnummer: WWI-18-DSA

Name: Bulkens, Björn;
Gemmer, Florian;
Gerber, Klemens

Matr.-Nr.: 9936663; 5358290; 4330617

E-Mail: s182146@student.dhbw-mannheim.de;
s180609@student.dhbw-mannheim.de;
s180614@student.dhbw-mannheim.de

Fachsemester: 06

Studiengang: BA Wirtschaftsinformatik - Data Science

Abgabedatum: 27. Juli 2021

Inhaltsverzeichnis

DQN Deep Q-Learning Network

IL Imitation Learning

PPO Proximal Policy Optimization

RL Reinforcement Learning

FPS Frames per Second

ACS Action Code Script

1 Einleitung

Die Herausforderungen im Reinforcement Learning (RL) lassen sich gut durch das Training eines Agents in einer Videospiel Umgebung veranschaulichen. Während RL Ansätze klassischerweise in 2D Spielumgebungen trainiert werden, wurde sich in dieser Arbeit dazu entschieden eine Implementierung im dreidimensionalen Raum anzugehen. Das Spiel *Doom* von 1993 eignet sich gut für die hier angestrebte Umsetzung. Das Spiel wurde von Fans als Open Source Software nachgebaut, und ist trotz 3D Umgebung weniger komplex als moderne First Person Shooter, was die Anzahl an möglichen Aktionen in der Spielumgebung eingrenzt.

Das Ziel ist es, einen RL-Agenten zu trainieren, der in der Lage ist in einer einfachen Spielumgebung annähernd menschliches Verhalten zu zeigen und im besten Fall das gestellte Ziel zu erreichen.

2 Konzeption

Um das Projekt umzusetzen ist eine Implementierung von Doom notwendig, welche es erlaubt mithilfe von Python Skripten in den Spielfluss einzugreifen. Die Bibliothek *ViZDoom* [1] wurde auf Basis eines Open Source Klons des originalen Doom entwickelt und bietet eine Vielzahl von hilfreichen Funktionen zum Entwickeln eines RL-Agents in einer originalgetreuen Doom-Umgebung. Zur Umsetzung eines RL Ansatzes wurden zwei verschiedene, etablierte Algorithmen gewählt: Q-Learning [2] und Proximal Policy Optimization (PPO) [3] .

Nachfolgend wird die Idee hinter ViZDoom erläutert und ein kurzer Überblick über die von der Bibliothek bereitgestellten Funktionen gegeben. Anschließend werden knapp die Grundlagen von Q-Learning und PPO erklärt. Um den Unterschied zwischen den beiden Ansätzen zu verdeutlichen, werden in Kapitel 2.2 On-Policy und Off-Policy Learning voneinander abgegrenzt.

2.1 ViZDoom

Bei ViZdoom handelt es sich um eine Library, die es erlaubt mit Hilfe des Screenbuffers von Doom visuelle Daten abzufangen, welche dann für eine weitere Verarbeitung, beispielsweise durch eine KI, zur Verfügung stehen. ViZdoom basiert auf ZDoom, einem Open Source Port der Doom Engine, und schafft einige Anbindungen um auf eine Vielzahl von Funktionen des Spiels mithilfe von eigenen Skripten Einfluss zu nehmen[1] . So können umfangreiche Daten für das Training unseres Reinforcement Learning Algorithmus genutzt werden.

Die Ausführungen in diesem Kapitel beruhen auf der Arbeit „ViZDoom: A Doom-based AI research platform for visual reinforcement learning“ von Kempka, Wydmuch, Runc et al. [4] in der das ViZDoom Framework zum Ersten mal vorgeschlagen wurde.

Der Grundgedanke von ViZDoom ist es, eine realitätsnähere Lernumgebung für Reinforcement Learning zu schaffen. Vor dem Aufkommen von ViZDoom gab es zwei Arten von spielbasierten Lernumgebungen für RL: 2D Spielumgebungen (beliebt sind hier immer noch Atari 2600 Spiele wie Space Invaders) oder 3D Spielumgebungen [5] . In diesen 3D Spielumgebugen wurde jedoch bis zum Aufkommen von ViZDoom stets auf „High Level“ Informationen, wie die Positionen von Wänden, Gegnern oder Objekten zurückgegriffen. Dem Agenten stehen somit alle (auch eigentlich verborgene) Informationen in leicht zu verarbeitender Form zur Verfügung. ViZDoom wurde entwickelt, um RL Algorithmen rein über visuellen Input zu trainieren. Damit agieren Agenten menschlicher, da die meisten, in anderen Ansätzen genutzten, High Level Informationen einem menschlichen Spieler nicht zur Verfügung stehen würden. Trotzdem gibt ViZDoom die Möglichkeit, Zugriff auf einige spielinterne Variablen, wie zum Beispiel Gesundheit oder Munition zu erlangen.

ViZDoom bietet ferner den Vorteil, sogenannte Szenarien zu nutzen. Diese Szenarien sind hoch anpassbar und beinhalten eigene Karten zu nutzen, dynamische Aktionen zu definieren (z.B. „Gegner erscheint wenn Spieler Waffe aufhebt“ oder „Spiel endet wenn Agent Türe erreicht“) und *Rewards* festzulegen. *Rewards* sind wichtig , um dem Agenten Feedback über die von ihm ausgeführte Aktion zu geben. Rewards können für beliebige Aktionen festgelegt werden, zum Beispiel für das Aufsammeln eines Objekts, das Erreichen des definierten Endes des Szenarios oder auch für jeden Tick den der Agent überlebt. Letzerer Reward wird meist mit negativ Werten versehen, um den Agenten dazu zu bewegen, das Szenario möglichst schnell zu lösen. Um Szenarien zu erstellen werden von den ViZDoom Machern zwei Open-Source Anwendungen empfohlen: Doom

Abbildung 1: Screenshot aus einem ViZDoom Szenario

Builder 2 und SLADE. In dieser Arbeit wurde sich für SLADE entschieden. Das mit SLADE Erstellte Doom Szenario wird in Kapitel 3 Umsetzung genauer erläutert.

Eine Spielinstanz von ViZDoom wird in sogenannte Ticks eingeteilt. 2100 Ticks entsprechen einer Minute in Echtzeit. Beruhend auf Ticks bietet ViZDoom zwei Kontroll-Modi an: Synchron und Asynchron. Asynchron bedeutet, dass das Spiel mit konstanten 35 Frames per Second (FPS) läuft. Ist der angewandte Algorithmus zu langsam, verpasst er Entscheidungen. Im synchronen Modus wartet das Spiel auf die Entscheidungen des Agenten.

2.2 On-Policy vs. Off-Policy Reinforcement Learning

Im RL wird zwischen zwei Arten von Policies unterschieden:

- Die **Behaviour Policy** wird vom Agent verwendet, um zu entscheiden welche der zur Auswahl stehenden Aktionen als nächstes ausgeführt werden soll (Abb. 2 links)

- Die **Target Policy** ist die Policy, die der Agent erlernen soll, um sein Ziel zu erreichen (Abb. 2 rechts)

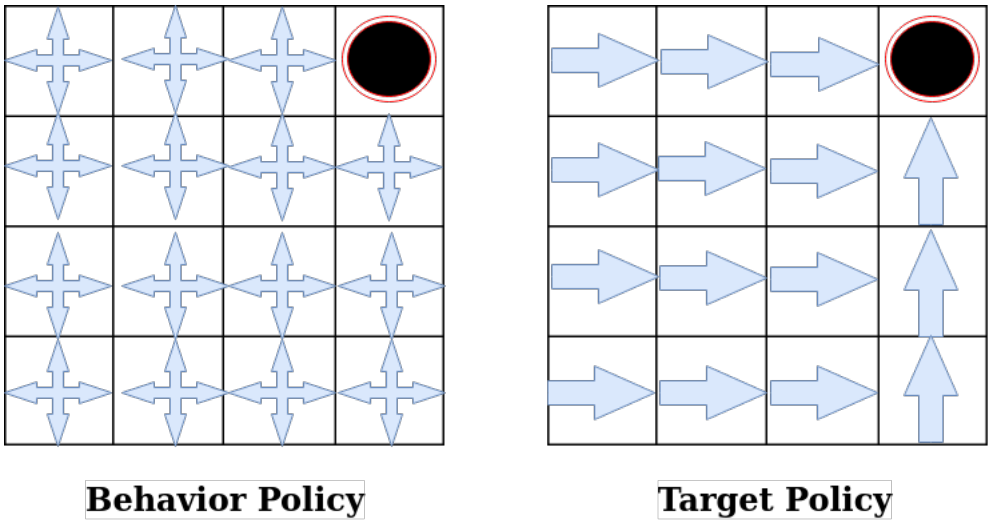

Abbildung 2: Behaviour Policy und Target Policy aus *On-Policy v/s Off-Policy Learning* [6]

Einfach ausgedrückt entspricht im On-Policy Learning die Behaviour Policy der Target Policy. On-Policy Algorithmen evaluieren und verbessern also die Policy, die verwendet wird um Aktionen auszuwählen. Da der neu hinzugefügte Datensatz im On-Policy Learning direkt verwendet wird, um die Policy zu aktualisieren, spricht man hier auch von *Online Learning* [7] . Im Off-Policy Learning entspricht die Behaviour Policy **nicht** der Target Policy. Es wird also eine Policy verbessert, welche nicht der Policy entspricht, die zur Auswahl der Aktionen verwendet wird. Beim Off-Policy Learning werden die Daten in Stapeln (Batches) verarbeitet. Erst wenn der gesamte Batch das Training durchlaufen hat, wird die Policy angepasst. Deshalb spricht man hier auch vom *Offline Learning* [7] .

2.3 Deep Q-Learning

Um einen Agent im Spiel Doom zu trainieren, wird als einer der Ansätze Q-Learning gewählt. Q-Learning ist ein Off-Policy RL Algorithmus, der für jeden möglichen Zustand alle in diesem Zustand möglichen Aktionen erfasst und diesen Aktionen einen Wert zuteilt. Der Wert richtet sich danach, wie hoch die Belohnung ist, welche sich durch Durchführen dieser Aktion erreichen lässt. Q-Learning lässt sich jedoch nur einsetzen, wenn sich alle möglichen Zustands-Aktions Paare in einer Tabelle oder Matrix darstellen lassen. Ein Spiel muss nicht übermäßig komplex sein um die Anzahl dieser Paare in die Millionen gehen zu lassen. Aus diesem Grund wird klassischer Weise ein Deep Q-Learning Network (DQN) gewählt. Neben dem klassischen DQN gibt es zwei auf ihm aufbauende Varianten, welche versuchen, vorhandene Schwächen von DQN zu beseitigen:

- **Double DQN** versucht das Problem zu lösen, dass DQN dazu tendiert, die Höhe der erwarteten Belohnungen zu überschätzen. Die Lösung von Double DQN besteht darin, die Auswahl der Aktion von der Evaluierung der Aktion zu entkoppeln. Dafür werden zwei DQNs verwendet. Das sog. Main DQN entscheidet welche nächste Aktion A die beste ist. Das zweite DQN, Target-DQN genannt, evaluiert die ausgewählte Aktion, um deren Q-Value zu erhalten [8] .

- **Duel DQN** wird speziell für Spiele eingesetzt, in denen es nicht notwendig ist den Wert jeder Aktion zu jedem Zeitpunkt zu kennen. Das gilt vor allem für Spiele, in denen nur eine Reaktion des Spielers auf gewisse Umwelteinflüsse gefordert wird. Zwischen den Momenten in denen reagiert werden muss, gibt es Zeiten, in denen keine Aktion des Spielers einen konkreten Vor- oder Nachteil bringt. Duel DQNs splitten das DQN Netzwerk nach der Verarbeitung der Frames in zwei Streams: Einer, um den State Value (Wie viel ist der momentane Zustand überhaupt wert?) zu kalkulieren und ein weiterer Stream, um die in diesem Zustand beste Aktion zu erhalten [9] .

2.4 Proximal Policy Optimization

Bei PPO handelt es sich um einen On-Policy RL Algorithmus, der 2017 von Open AI in *Proximal Policy Optimization Algorithms* [3] vorgestellt wurde. PPO wurde als direkte *Antwort* zu den drei zu dieser Zeit führenden RL Ansätzen entwickelt: Q-Learning, „Vanilla“ Policy Gradient Methoden und Trust Region Policy Optimization (TRPO). Die Entwickler von PPO bemängeln, dass keine der drei genannten Methoden gleichzeitig skalierbar, dateneffizent und robust ist. PPO soll die besten Eigenschaften der oben genannten Ansätze vereinen und dabei einfacher verständlich sowie einfacher implementierbar sein.

Der konkrete Unterschied zu Q-Learning besteht in dem On-Policy Ansatz von PPO. Im Q-Learning wird ein sogenannter *Replay Buffer* verwendet, in dem sämtliche vergangene Erfahrungen gespeichert werden. PPO im Gegensatz dazu verwendet *Mini Batches*. Dabei werden im Umgang mit der Umgebung kleine Erfahrungs-Batches gesammelt. Diese Batches werden verwendet, um die Policy des PPO Algorithmus zu aktualisieren. Anschließend werden die Batches verworfen. Damit ist PPO im Gegensatz zu Q-Learning eine online-learning Methode, welche nicht auf offline gespeicherte Daten zurückgreifen kann um ihr Training voranzutreiben [10] .

Der Schlüsselbeitrag von PPO liegt darin sicherzustellen, dass sich eine aktualisierte Policy nicht zu weit von ihrem Vorgänger entfernt. Damit wird verhindert, dass der Agent einen Pfad verfolgt, welcher ihn nicht zum Ziel bringt, ohne die Möglichkeit diesen wieder zu verlassen [11].

3 Umsetzung

Für das Training eines Reinforcement Algorithmus werden einige Grundvoraussetzungen benötigt:

1. Eine Umgebung
2. Ein Agent, der in der Umgebung agiert
3. Eine Möglichkeit für den Agent, die Umgebung wahrzunehmen
4. Eine Reward Funktion, die dem Agent eingibt, welche Aktionen gewünscht sind und welche nicht
5. Ein Algorithmus, der den Status des Agents verarbeitet, auf dessen Basis die Aktion berechnet, die den höchsten Reward verspricht und diese Aktion an den Agent weitergibt

Diese Voraussetzungen werden im Folgenden thematisiert.

3.1 Aufbau eines Doom Szenarios

Das ViZDoom Szenario kann auch als Regelwerk der Umgebung gesehen werden, in der der Agent agiert. Es enthält sowohl die Karte, also den Raum in dem der Agent sich bewegen kann, als auch alle Elemente darin (z.B. Gegenstände), sowie die Regeln, die in der Umgebung gültig sind. Diese können über Skripte manipuliert werden. Ein ViZDoom Szenario besteht aus einer *.wad*-Datei, in der alle grundlegenden Daten gespeichert sind, die für die Nutzung des Szenarien nötig sind. Diese Dateien sind gewissermaßen ein Archiv. Ebenfalls Teil des Szenarios ist eine Konfigurationsdatei, in der bestimmte Regeln wie Rewards spezifiziert werden, die nicht in der *.wad* gespeichert werden können. Zusammen ergeben sie ein Szenario, auf das ein Algorithmus zugreifen und in dem ein Agent agieren kann.

Innerhalb eines Szenarios wird, im Falle eines RL-Agents, meist ein Ziel definiert, bei dessen Erreichung eine Iteration des Lernprozesses endet. Dies kann sowohl das Ankommen an einem bestimmten Punkt der Karte, das Einsammeln eines Gegenstand, als auch das Besiegen eines Gegners sein. Somit sind Szenarien nicht vollständig vom Lernprozess entkoppelt, sondern bilden seine Grundzüge und Regeln ab, die zur Erreichung des Ziels beitragen.

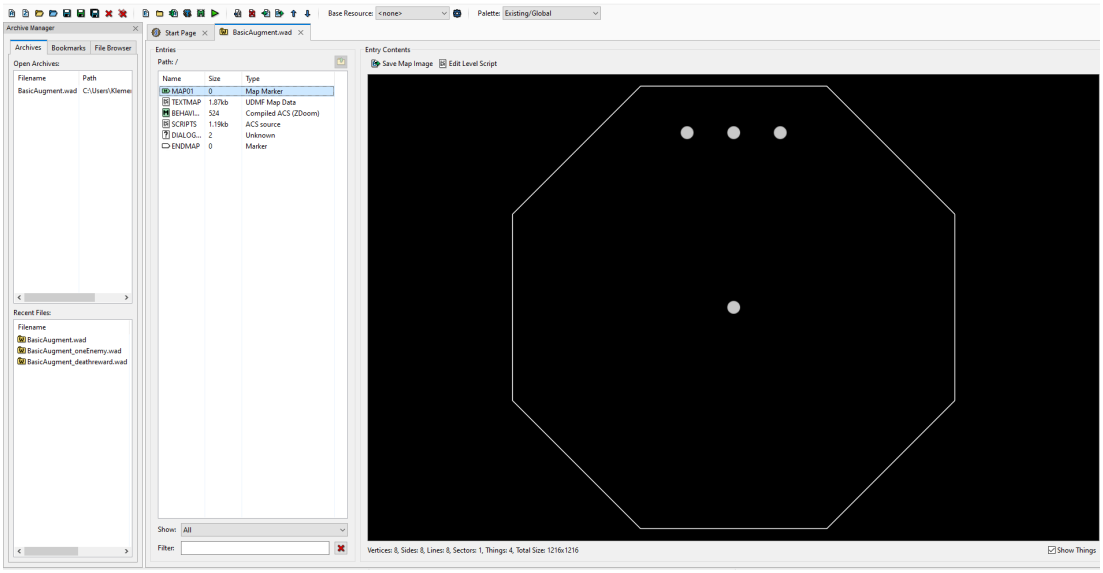

Abbildung 3: SLADE mit Szenario Files und Map Layout

3.2 Erstellung des Levels

Für das Training des Agents wird ein Level erstellt, das die Grundlage unseres Trainings bildet. Dieses Level wird mithilfe der Software *SLADE* erstellt, die es ermöglicht, .wad-Dateien zu verändern und, unter anderem, Doom-Level zu erstellen.

Neben der Änderung des Rewards wird versucht, den aufwändigen Trainingsprozess durch eine stetige Optimierung des Level-Aufbaus effektiver zu gestalten. Hierzu zählen Punkte, wie das Hinzufügen visueller Fixpunkte für eine bessere Orientierung. Dies ist wichtig, da der Agent nur über rein visuellen Input seine Position auf der Karte und die Nähe/Existenz von Gegenständen und Gegnern wahrnehmen kann.
Eine Optimierung in diesem Sinne ist nötig, um unter den gegebenen Ressourcen-einschränkungen, das Training innerhalb einer virtualisierten Linux-Umgebung ohne GPU-Unterstützung umsetzten zu können.
Erstellt wird ein Level, bei dem der Agent zuerst einen Gegenstand (Gewehr) im Raum finden und aufheben muss. Dieser Gegenstand ist mit deutlichen visuellen Fixpunkten versehen und aktiviert beim Aufheben ein Skript. Dieses Skript sorgt dafür, dass in einem festgelegten Bereich an zufälliger Stelle ein Gegner erscheint. Ist ein Gegner besiegt, treten zwei neue an seine Stelle. Das Level kann nicht aktiv beendet werden, das Ziel ist es so lange wie möglich zu überleben unter der Voraussetzung, dass die Waffe aufgehoben wurde. Abbildung 3 zeigt den gewählten Editor SLADE und die Bearbeitung einer Karte.

3.3 Rewards

Um die gewünschten Aktionen im Environment zu finden und diese bestmöglich umzusetzen, muss der Algorithmus die Rewards der Aktionen des Agents verarbeiten können. Hierfür muss die Ausschüttung der Rewards an bestimmte Aktionen geknüpft werden. Rewards können sowohl positiv, als auch negativ sein. Beispielsweise werden klassisch negative Rewards pro Spieltick ausgeschüttet, um den Agent zu einer schnelleren Lösung des gestellten Problems zu treiben. Im Gegensatz dazu, werden bei Erreichung des gesetzten Ziels oder eines Teilziels positive Rewards ausgeschüttet. Diese müssen jedoch in einer entsprechende Höhe ausfallen, die eventuell vorher erhaltene, negative Rewards angemessen ausgleicht, um einen Anreiz zu schaffen den gewählten Weg weiter zu verfolgen.
Der Prozess, hilfreiche und zielführende Rewards zu generieren, ist ein ebenso wichtiger Prozess, wie das spätere Hyperparameter-Tuning, den Agenten den Sachverhalt effizient erlernen zu lassen.
Im Zuge dieses Projekts wurde ein Vielzahl an einzelnen Rewards für bestimmte Ereignisse, sowie deren Kombination ausprobiert. Dadurch haben wir beispielsweise festgestellt, dass ein negativer Reward für das Erhalten von Schaden durch Gegner den Agenten davon abhält die Waffe aufzuheben: Denn Gegner erscheinen nur durch das Aufheben der Waffe. Da der Agent in den ersten Iterationen noch kein Verständnis dafür entwickelt hat, dass er auf diese Gegner schießen muss, um einen positiven Reward zu erhalten, lernt er indirekt, dass das Aufheben der Waffe selbst schlecht ist. Eine Bestrafung für das Abfeuern der Waffe erwies sich hingegen als sinnvoll, um Dauerfeuer zu vermeiden und das gezielte Erkennen von Gegnern zu stimulieren.

4 Herausforderungen

Im Folgenden werden einige Herausforderungen besprochen, die die Umsetzung des Projekts mit sich bringt.

4.1 Scripting und Rewards

Das Bearbeiten der WAD bietet einige Herausforderungen, sowohl in der grundsätzlichen Funktionsweise von Action Code Script (ACS)-Skripten, die in ZDoom umgesetzte Skriptsprache, als auch die technische Umsetzung der angesprochenen Rewards für bestimmte Ereignisse. Diese stellte insofern eine Herausforderung dar, als dass ZDoom und SLADE unter Linux genutzt werden müssen, gängige Compiler für ACS jedoch nur für Windows zur Verfügung stehen.

Weiterhin müssen passende Bedingungen gefunden werden, bei deren Eintritt ein Reward ausgeschüttet wird. Dies wird durch die Wahl der Höhe des Rewards noch verkompliziert, da das Verhältnis zwischen der Höhe der Rewards ausschlaggebend für die Performance des Modells sein kann. Dies ist besonders gut am Beispiel e=130 zu erkennen (Siehe Anhang A). Hier führt eine Kombination aus einem negativen Living-Reward, dieser wird pro Spieltick ausgeschüttet, einem negativen Reward für Schüsse sowie für das Schaden nehmen und Game Over des Agents dazu, dass die negativen Rewards im Verhältnis zu den positiven zu übermächtig sind und das Modell somit nur lernt, dass im Kreis laufen und weder positive, noch zusätzliche negative Rewards erhalten als valide Strategie erscheint.

4.2 Performance

Da das Training der Modelle aufgrund der relativ schwachen Rechenleistung der Systeme auf denen das Training stattfindet eine vergleichsweise lange Zeit benötigt (ca. 10-15 Epochen pro Stunde) werden Fehler und Herausforderungen, wie die im vorherigen Kapitel besprochenen, erst relativ spät bemerkt und kosten Zeit.
Im Vergleich zu ähnlichen Projekten die eine Gesamtzahl von 300.000 bis 500.000 Trainingsschritten mit insgesamt 20 Workereinheiten einsetzen (siehe GutHubs zu ähnlichen Problemstellungen), ist die hier verwendete Konfiguration marginal. Um ähnliche Operationen durchzuführen, müssten bei einer derart minimalen Rechenleistung unverhältnismäßige Trainingszeiten bewältigt werden. Dies ist als eine der hauptsächlichen Herausforderungen dieses Projektes zu betrachten und führte zu einer zwingenden Simplifizierung des Environments und zu eher unzufriedenstellenden Ergebnissen.

4.3 Implementieren von Imitation Learning

Eine Möglichkeit, um das oben genannte Performance Problem zumindest zu verringern, stellt Imitation Learning (IL) dar. Im Gegensatz zu reinem RL startet der Agent im IL mit vorinitialisierten Gewichten, die durch das Beobachten eines menschlichen Spielers zustande kommen. ViZDoom bietet mit dem *Spectator Mode* auch einen Modus an, mit dem Imitation Learning umgesetzt werden kann. Der Spectator Modus gibt die Spieleraktionen jedoch in einem anderen Format zurück, als das Format das vom Agent beim selber spielen erzeugt wird. Trotz ausführlichem Debbuging ist es dem Team nicht gelungen einen IL Ansatz in ViZDoom umzusetzen. Ein solcher Ansatz könnte jedoch die Baseline für ein Training mit Reinforcement Learning darstellen und die Ergebnisse verbessern.

5 Ergebnisse und Bewertung

Insgesamt wurden 20 verschiedene Hyperparameterkombinationen für das Duel DQN getestet. Oft wurden dabei durch die Beschränkung der Ressourcen Modelle denen eine gute zufällige Initalisierung gelang als Basis genutzt, um von diesen aus weiter zu trainieren. Die Bewertung der Kombinationen erfolgte hierbei sowohl anhand der gesammelten Rewards, als auch anhand bestimmter *Zieloperationen*, deren Erfüllung das Ziel des Agents sein soll. Diese Operationen sind "Waffe aufnehmen", "Gegner töten" und "Game Over". Da die Handlungen des Agentes immer noch zu einem gewissem Grad abhängig von Zufällen ist, werden jeweils 10 Iterationen beobachtet und zusammengefasst dokumentiert. Diese Eckpunkte wurden hinsichtlich ihrer einfachen und ihrer verlässlichen Durchführung untersucht. Einen zusätzlichen Performanceindex gibt die maximale Anzahl von getöteten Gegnern. Die Ergebnisse ausgewählter Varianten sind in Anhang Dokumentation der Ergebnisse auf Seite 17 gelistet.

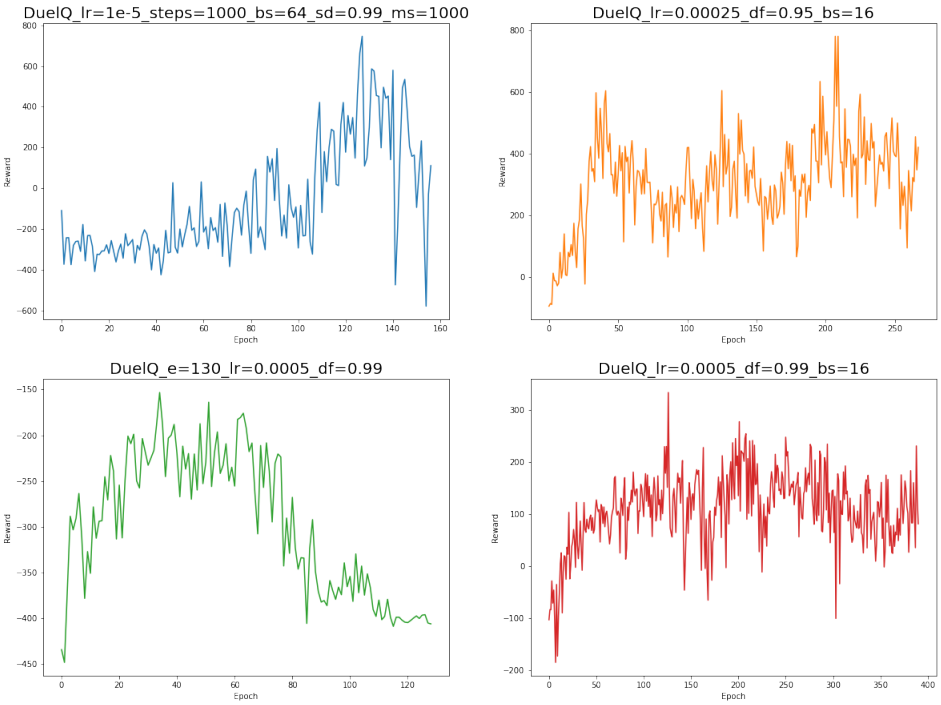

Abbildung 4: Rewards verschiedener Hyperparameter Kombinationen

Nach diesen Kriterien beurteilt, konnte keine Kombination gefunden werden, die eindeutig die besten Ergebnisse liefert. Zwar konnten einige Parameterkombinationen gefunden

werden, die die gesetzten Ziele angemessen erreichen, jedoch konnte die Auswahl der Parameter unzureichend oft neu trainiert werden, um aussagekräftige Stichproben darzustellen. Dies ist nötig aufgrund der anfänglich zufälligen Wahl der Aktionen.

Aufgrund der bereits thematisierten Herausforderungen konnten lediglich stark schwankende Ergebnisse erzielt werden. Einzelne Modelle können die Waffe verlässlich finden, Gegner besiegen und zeigen Ansätze von Strategien, den Schüssen der Gegner auszuweichen. Trotzdem neigen auch die Besten der trainierten Modelle dazu in einzelnen Iterationen im Kreis zu laufen, oder die Waffe um wenige Pixel zu verpassen was zu einer schlechten Performance in der jeweiligen Iteration führt. Die inkonsistente Performance verschiedener Modelle ist in Abb. 4 dargestellt. Zwar konnten grundsätzlich annehmbare Ergebnisse erzielt werden, diese weisen jedoch eine Varianz in ihrer Qualität auf und können in den wenigsten Fällen mit einem menschlichen Spieler mithalten.

6 Fazit

Reinforcement Learning stellt auch im Kontext eines Videospiels eine Herausforderung dar. Zwar ist das Environment hier weit weniger Komplex und die Anzahl der möglichen Aktionen geringer als in vielen anderen Anwendungsfällen wie beispielsweise Robotik, jedoch ist auch hier der Umfang der Challenge Reinforcement Learning nicht zu leugnen. ViZDoom erleichtert die Implementierung eines Machine Learning Algorithmus zwar signifikant indem eine Flexible Anbindung an das Spiel geschaffen wird, jedoch ist das erreichen einer annehmbaren Performance nur mit starker Rechenleistung und einem entsprechenden zeitlichen Rahmen möglich. Zudem stellt die Nutzung von Tools wie Slade unter Linux für unerfahrene Benutzer eine Herausforderung dar, deren Bewältigung einige Zeit in Anspruch nehmen kann. Ein Ansatz, der zu besseren Ergebnisse führen könnte, wäre der Aufbau auf Imitation learning. Die Implementierung dieses vielversprechenden Konzepts wurde jedoch von Bugs verhindert, die nicht in angemessener Zeit gefixt werden konnten.

Letztendlich stellt das Projekt für alle Teilnehmer eine interessante Lernerfahrung dar, da sowohl Reinforcement Learning besser verstanden werden konnte als auch der Einfallsreichtum gefördert wurde, der nötig war, um trotz begrenzter Ressourcen ein annehmbares Ergebnis zu erzielen.

Literatur

[1] M. Kempka, M. Wydmuch, G. Runc, J. Toczek und W. Jaśkowski, „ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning," in *IEEE Conference on Computational Intelligence and Games*, The best paper award, Santorini, Greece: IEEE, Sep. 2016, S. 341–348. Adresse: http://arxiv.org/abs/1605.02097.

[2] C. J. Watkins und P. Dayan, „Q-learning," *Machine learning*, Jg. 8, Nr. 3-4, S. 279–292, 1992.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford und O. Klimov, *Proximal Policy Optimization Algorithms*, 2017. arXiv: 1707.06347 [cs.LG].

[4] M. Kempka, M. Wydmuch, G. Runc, J. Toczek und W. Jaskowski, „ViZDoom: A Doom-based AI research platform for visual reinforcement learning," *IEEE Conference on Computatonal Intelligence and Games, CIG*, Jg. 0, S. 0–7, 2016, ISSN: 23254289. DOI: 10.1109/CIG.2016.7860433. arXiv: 1605.02097.

[5] R. S. Sutton und A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[6] A. Suran, *On-Policy v/s Off-Policy Learning*, 2020. Adresse: https://towardsdatascience.com/on-policy-v-s-off-policy-learning-75089916bc2f.

[7] B. Aunkofer, *Machine Learning: Online vs Offline*, 2018. Adresse: https://data-science-blog.com/blog/2018/03/08/machine-learning-online-vs-offline-lernen/.

[8] M. S. Ausin, *Introduction to Reinforcement Learning. Part 4: Double DQN and Dueling DQN*, 2020. Adresse: https://markelsanz14.medium.com/introduction-to-reinforcement-learning-part-4-double-dqn-and-dueling-dqn-b349c9a61ea1.

[9] C. Yoon, *Dueling Deep Q Networks*, 2019. Adresse: https://towardsdatascience.com/dueling-deep-q-networks-81ffab672751.

[10] S. Saikia, *Deep Reinforcement learning using Proximal Policy Optimization*, 2020. Adresse: https://medium.com/analytics-vidhya/deep-reinforcement-learning-using-proximal-policy-optimization-7555280ef941.

[11] C. Trivedi, *Proximal Policy Optimization Tutorial (Part 1/2: Actor-Critic Method)*, 2019. Adresse: https://towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9afffbf6.

Eigendarstellung

Innerhalb unseres Projektes einen DOOM-Bot über Reinforcement Learning zu erstellen, gab es nicht immer eine klare Aufgabentrennung. Die Gruppenmitglieder versuchen im Folgenden ihren Eigenanteil herauszustellen, jedoch sei gesagt, dass an jedem der genannten Prozesse auch immer die restlichen Mitglieder Teil hatten und jeweils nur die zeitaufwendigsten und zentralen Aufgabenaspekte genannt werden.

A Anteil Björn Bulkens

In diesem Projekt war ich für die Planung und Umsetzung der unterschiedlichen Reinforcement Learning Algorithmen als Senior Developer hauptverantwortlich. Hierzu gehört die Umsetzung der Abstract Base Classes aus denen die unterschiedlichen RL-Algorithmen erben sollen, sowie die Implementierung und Anpassung des DQN-Agents und -Modells. Die Umsetzung eines Transfer Learning Approaches und des PPO-Algorithmus sind leider am zeitlichen Aufwand gescheitert, durch diverse Probleme in der initialen Aufsetzung des Projektes.
Zusätzlich habe ich eine Funktionalität umgesetzt, mit Hilfe deren man über Terminal bei Start des Programmes alle wichigen Parameter des Trainings einschließlich der Auflösung der Bilder, welche für das Training verwendet werden, um ohne Programmierkenntnisse komfortabel Hyperparameter-Tuning selbst umsetzen zu können.
Mit einbezogen in diese Aufgabe ist die vollständige Dokumentation des in Python geschriebenen Quelltextes und Code-Reviews für die Beiträge anderer Gruppenmitglieder zu diesem.
Zusätzlich war ich unterstützend bei der Erstellung des Enviroments und der Reward-Funktion tätig, durch eine ständige Evaluierung der Modelle und Fehleranalyse zu Fehlverhalten des Agenten. Die gefundenen Fehler wurden dann entweder über eine Anpassung des Modells oder eine Anpassung des Levels und der zugehörigen Skripte gelöst. Die Modelle wurden anschließend in weiteren Iterationen evaluiert.
Mit meinen Teamkollegen zusammen habe ich das Paper und die zugehörige Präsentation verfasst. Hier lässt sich keine speziellere Teilung der Aufgaben dokumentieren.

B Anteil Florian Gemmer

Zu Beginn des Projekts habe ich mich mit der der Findung des passenden Environments beschäftigt. Dazu gehören die Entscheidung für ViZDoom, das Aufsetzen in einer Linux VM und das Installieren des zugehörigen Base Games (ZDoom). Anschließend habe ich zusammen mit Klemens das Environment konzipiert, wozu das Erstellen der Karte, positionieren von Objekten in der Spielwelt und dynamisches Spawning von Gegnern gehören. In der Umsetzung der RL Algorithmen habe ich eine unterstützende Rolle in der Fehleranalyse übernommen, jedoch nicht aktiv die Modelle codiert beziehungsweise trainiert. Ich habe mit den von Klemens und Björn erhaltenen Daten den Trainingsfortschritt unserer mit verschiedenen Hyperparamtern getuneten Modelle visualisiert. Schlussendlich habe ich mich mit den theoretischen Grundlagen von ViZDoom, PPO und Q-Learning befasst, um eine theoretische Grundlage für die wissenschaftliche Arbeit zu schaffen.

Mit meinen Teamkollegen zusammen habe ich das Paper und die zugehörige Präsentation verfasst. Hier lässt sich keine speziellere Teilung der Aufgaben dokumentieren.

C Anteil Klemens Gerber

In unserem Projekt habe ich die Position des Junior Developers übernommen. Meine Aufgaben umfassten unter anderem das Organisieren, Durchführen und Dokumentieren des Hyperparameter-Tunings, sowie die Unterstützung bei der Anpassung der Lernalgorithmen. Dazu gehörten das Einpflegen der Ergebnisse in die dafür vorgesehene Tabelle, als auch das Organisieren und Bereitstellen von Modellen und ihren zugehörigen Meta-Daten. Diese wurden anschließend von meinen Teamkollegen verarbeitet. Weiterhin habe ich beim allgemeinen Coding unterstützt.

Ebenfalls habe ich mich mit der Umsetzung der Reward Funktionen mit ACS und den damit verbundenen Challenges befasst, die im Paper bereits beschrieben wurden. Besonders ist hierbei das Scripting hervorzuheben, das aufgrund von Unerfahrenheit mit C-Ähnlichen Sprachen einige Zeit beanspruchte. Florian und ich haben gemeinsam das Environment konzipiert und umgesetzt. Zudem habe ich durch weitreichende Tests in Standardenvironments Challenges wie das Sparse Reward Problem und deren mögliche Lösungen untersucht. Hier ist vor allem das Beispiel ”mywayhome.wad” zu nennen, dessen Bewältigung unser ursprünglicher Plan war. Diesbezüglich habe ich einige Tests durchgeführt und die Machbarkeit evaluiert.

Mit meinen Teamkollegen zusammen habe ich das Paper und die zugehörige Präsentation verfasst. Hier lässt sich keine speziellere Teilung der Aufgaben dokumentieren.

D Dokumentation der Ergebnisse

Epoch	10	10	60	10	130	40	80	110	270	130	170	410	400
LR	1e-3	1e-3	1e-3	5e-4	5e-4	25e-5	25e-5	25e-5	25e-5	1-e5	2e-4	2e-4	0,0005
Discount	0,99	0,95	0,95	0,95	0,99	0,99	0,99	0,99	0,95	0,99	0,96	0,96	0,99
Steps	2k	2k	2k	2k	2k	2k	2k	2k	1k	1k	1k	1k	1k
Memory Size	10k	10k	10k	10k	10k	10k	10k	10k	10k	1k	10k	10k	20k
BS	64	64	64	64	64	32	64	32	16	64	16	16	16
min Reward	-210	-151	N/A	-206	-412	N/A	34,7	-50	40	-477	-55	165	-75
max Reward	35	-76	N/A	-21	-369	N/A	245	465	1450	936	740	735	350
mean	-156,8	-106,3	N/A	-173,2	-406,1	N/A	121	143	348	221	304,5	333	83,5
sigma	37,2	25,5	N/A	23,1	-382	N/A	77,6	148	393,5	410,8	262,7	166,9	132,8
Gun Pickup	yes	yes	no	yes	no	yes	yes	yes	yes	yes	yes	yes	no
Once Gun Pickup		yes	no	yes	no	yes	yes	yes	yes	yes	yes	yes	no
Reliable Kill	no	no	no	no	no	yes	no	no	yes	yes	yes	yes	no
Enemy Kill	no	yes	no	no	no	yes	yes	yes	yes	yes	yes	yes	no
Enemy Reliable	no	no	no	no	no	no	no	no	yes	yes	yes	yes	no
Survive Once	yes	no	yes	yes	yes	yes	no	no	no	yes	no	no	yes
Always Survive	yes	no	yes	no	yes	no	no	no	no	yes	yes	no	yes
Max Kills	0	1	0	0	0	1	5	0	6	5	6	6	0

Tabelle 1: Auswahl einiger Varianten und den beobachteten Performance über zehn Versuche

Legende: Grün markiert für besonders gut befundene Agenten, rot Fehlschläge in Relation zum Trainingsaufwand. Alle farblich hervorgehobenen Agenten können sind als Video per Link im GitHub bereitgestellt. Varianten, die nicht von einander abgegrenzt sind bauen auf einander auf.

Eigenständigkeitserklärung

Hiermit versicheren wir, dass wir die Hausarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, alle Ausführungen, die anderen Schriften wörtlich oder sinngemäß entnommen wurden, kenntlich gemacht sind und die Arbeit in gleicher oder ähnlicher Fassung noch nicht Bestandteil einer Studien- oder Prüfungsleistung war.

Mannheim, den 27. Juli 2021

Björn Bulkens, Florian Gemmer, Klemens Gerber