Duale Hochschule Baden-Württemberg Mannheim

Schriftliche Ausarbeitung

Atari-Freeway-Game

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

https://github.com/JanMuehlnikel/Atari-Freeway-Reinforcement-Learning

Verfasser:	Franziska Marb, Jan Mühlnikel
Matrikelnummern:	5288260, 2235021
Kurs:	WWI21DSA
Studiengangsleiter:	Prof. Dr.-Ing. habil. Dennis Pfisterer
Dozent:	Dr. Manuel Zeise
Modul:	Aktuelle Data Science Entwicklungen II
Bearbeitungszeitraum:	08.05.2024 – 10.07.2024
Eingereicht:	10.07.2024

Disclaimer

Die nachfolgende Arbeit wurde als Gruppenarbeit von allen Verfassern geschrieben und soll daher mit einer Gesamtnote für alle Verfasser bewertet werden.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

In der Forschung zum maschinellen Lernen, insbesondere im Bereich des Reinforcement Learning, haben sich klassische Atari-Spiele als wertvolle Testumgebungen etabliert. Die Spiele bieten komplexe und dynamische Szenarien, in denen Agenten durch Interaktionen mit der Umgebung optimale Entscheidungen treffen können. Eine der wesentlichen Herausforderungen im Bereich des maschinellen Lernens ist die Entwicklung von Agenten, die in komplexen Umgebungen effizient lernen und handeln können [1] [2] [3].

In diesem Projekt wird die Anwendung von Reinforcement Learning im Kontext des Atari-Spiels „Freeway“ untersucht. Dabei erfolgt ein Vergleich der Implementierung eines Reinforcement-Learning-Algorithmus mit einer Baseline-Methode. Das Hauptziel besteht darin, zu evaluieren, ob die Reinforcement-Learning-Methode erfolgreich lernt, im Spiel zu bestehen.

1.1 Problem

Das Atari-Spiel „Freeway“ stellt ein klassisches Problem aus der Domäne des Reinforcement Learnings dar. Im Spiel steuert der Spieler einen Avatar, der einem Huhn nachempfunden ist. Das Ziel des Spiels besteht darin, eine stark frequentierte Straße zu überqueren, um auf die andere Seite zu gelangen. Die Straße ist in zehn Fahrspuren unterteilt, auf denen Fahrzeuge mit unterschiedlichen Geschwindigkeiten fahren. Der Spieler muss dem Verkehr ausweichen, um sicher auf die andere Seite zu gelangen. Sollte das Huhn mit einem Fahrzeug kollidieren, wird es je nach gewählter Schwierigkeit mehrere Fahrbahnen zurückgeworfen. Jeder erfolgreiche Übergang wird mit einem Punkt belohnt. Ziel ist es, die Fahrbahn so oft wie möglich zu überqueren, um möglichst viele Punkte zu sammeln. Der Spieler hat dafür 2 Minuten und 16 Sekunden Zeit [4].

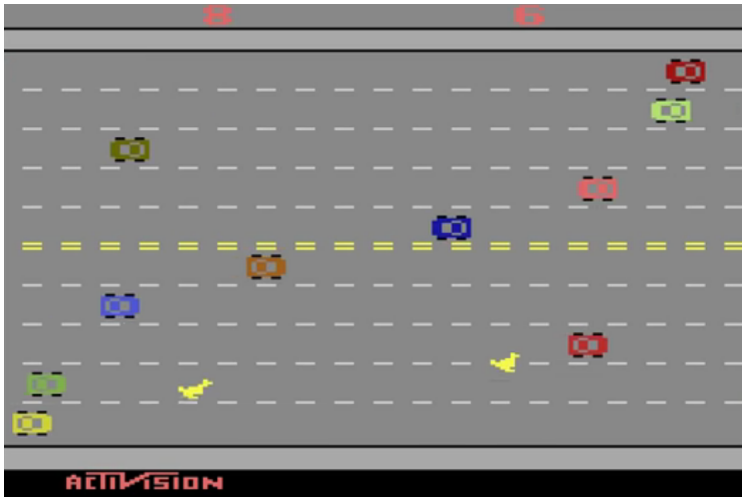

Abbildung 1.1: Beispielhafte Darstellung einer Spielsituation des Atari Freeway Spiels
Quelle: [5]

1.2 Umgebung

Das „Freeway“ Spiel wird in diesem Projekt durch das Gymnasium Framework implementiert. Das Spielfeld in „Freeway“ wird durch ein zweidimensionales Raster dargestellt, wobei der Spielcharakter von unten nach oben über das Spielfeld bewegt wird. Das Gymnasium Framework ermöglicht es, für das Atari-Freeway-Spiel zwischen drei verschiedenen Zustandsrepräsentationen zu wählen. Erstens kann der Zustand des Spiels als Bild ausgegeben werden, wobei zwischen einer RGB- und einer Graustufen-Darstellung unterschieden wird. Zweitens kann der Zustand als RAM-Version ausgegeben werden, in der er durch ein 128 Byte großes Array dargestellt wird.

Der Spieler startet das Spiel am unteren Rand der Straße. Während des Spiels bewegen sich Fahrzeuge kontinuierlich horizontal über die Spielfläche. Das Gymnasium Framework bietet acht verschiedene Spielmodi an. Die Modi unterscheiden sich in der Frequenz und Geschwindigkeit der Fahrzeuge. Es ist zu beachten, dass die Statik innerhalb eines Modus gleich bleibt und sich in verschiedenen Durchläufen nicht verändert. Bei einer Kollision mit einem Fahrzeug wird der Spielercharakter zurückgesetzt, wobei die Entfernung, um die der Charakter zurückgesetzt wird, von der gewählten Schwierigkeitsstufe abhängt. Punkte können nur erzielt werden, wenn der Spieler

erfolgreich die gesamte Straße überquert. Um den Avatar zu bewegen, stellt das Framework drei verschiedene Aktionen zur Verfügung: Verharren an einer Position, Aufwärtsbewegung und Abwärtsbewegung. Eine detaillierte Darstellung der Aktionen und ihrer zugehörigen Werte ist in Tabelle 1.1 aufgeführt. [6]

Tabelle 1.1: Aktionen, welche durch das Gymnasium Framework zur Verfügung gestellt werden

Beschreibung	**Wert**
Verharren	0
Aufwärtsbewegen	1
Abwärtsbewegen	2

1.3 Vereinfachungen & Einstellungen

Zur besseren Nachvollziehbarkeit der Ergebnisse sowie zur erleichterten Durchführung des Projekts wurden Vereinfachungen vorgenommen. Zum einen wurde die Gymnasium-Umgebung „ALE/Freeway-v5“ verwendet, um eine standardisierte und reproduzierbare Umgebung zu gewährleisten. Um das Training zu beschleunigen, wurde das Spielziel leicht modifiziert. In jeder Spielsitzung wird nur ein einziger Versuch unternommen. Ein Versuch gilt als erfolgreich, wenn das Huhn das gegenüberliegende Ende der Fahrbahn erreicht, und als nicht erfolgreich, wenn das Huhn mit einem Fahrzeug kollidiert. Zudem wird der Spielmodus zufällig gewählt, um die Statik des Spiels variabel zu halten. Da das Spiel bei einer Kollision endet, kann der Schwierigkeitsgrad vernachlässigt werden.

Die Konfiguration und die Verwendung der standardisierten Umgebung erleichtern die Durchführung von Tests und die Vergleichbarkeit der Ergebnisse mit anderen Experimenten. Detaillierte Informationen zu verwendeten Umgebung sind der Gymnasium Dokumentation [6] zu entnehmen, während spezifische Spielregeln und Hintergründe auf der offiziellen Atari-Seite [4] bereitgestellt werden.

2 Methoden

2.1 Repräsentation des Zustands

Bei der Wahl der Repräsentation des Zustandes ist davon auszugehen, dass die RGB-Zustandsrepräsentation die komplexeste Form der Darstellung darstellt. Darauf folgt die Graustufen-Zustandsrepräsentation, während die RAM-Repräsentation die geringste Komplexität aufweist [6].

Um die Komplexität des Modells und die Trainingsdauer möglichst gering zu halten, wurde in dieser Arbeit die RAM-Repräsentation gewählt. Diese Entscheidung basiert auf dem geringeren Ressourcenbedarf im Vergleich zu den bildbasierten Repräsentationen. Darüber hinaus wurde die RAM-Repräsentation weiter spezifiziert, indem nur die relevantesten Bytes betrachtet werden. Zur Identifikation dieser relevanten Bytes werden die gängige Atari RAM-Annotierungen herangezogen. Der Zustand in der RAM-Repräsentation wird durch einen Array mit 128 Bytes dargestellt. Jedes Byte in diesem Array kann 256 (zwischen 0 und 255) verschiedene Werte annehmen. [7] [8]

Tabelle 2.1: Relevante RAM-Bytes für das Atari-Freeway-Spiel

Beschreibung	**Index**
Position des Spielers (y-Koordinate)	14
Kollision	16
Punktestand	103
Position der gegnerischen Autos (x-Koordinate)	108-117

Außerdem wurden die möglichen Zustände innerhalb der Bytes weiter eingegrenzt, indem das Kollisionsbyte binär umgewandelt wird. Der Wert 255 signalisiert dabei eine Kollision, während alle anderen Werte keine Kollision anzeigen. Weiterhin wurde das Byte der y-Koordinate des Spielers, welches eine Position zwischen 0 und etwa 170 repräsentiert, gedrittelt. Diese Unterteilung führt zwar zu einer geringeren Genauigkeit bei der Positionsangabe, reduziert jedoch die Anzahl der möglichen Zustände erheblich. Selbiges wurde für die Position der gegnerischen Autos durchgeführt. [8]

2.2 Baseline-Methodik

Die Baseline-Methode wird verwendet um den Erfolg des implementierten Reinforcement-Algorithmus zu messen. Die hier gewählte Methode stellt einen naiven Lösungsansatz dar, der keine Lerneigenschaften besitzt.

In Anbetracht des Ziels, die Fahrbahn so schnell wie möglich zu überqueren, um eine maximale Punktzahl zu erzielen, erscheint es intuitiv, das Huhn kontinuierlich aufwärts zu bewegen. Dieser Ansatz basiert auf der Annahme, dass eine gleichmäßige und ununterbrochene Bewegung nach vorne die effektivste Strategie darstellt, um eine möglichst große Strecke in kürzester Zeit zurückzulegen.

Diese Strategie birgt jedoch Risiken. Eine kontinuierliche Aufwärtsbewegung des Huhns minimiert zwar die Zeit, des Huhns auf der Fahrbahn, erhöht jedoch die Wahrscheinlichkeit einer Kollision. Das Huhn hat dementsprechend keine Möglichkeit, seinen Weg zu ändern oder auf Fahrzeuge zu reagieren.

Die zurückgelegte Strecke bei dieser Methode ist daher zufällig und hängt davon ab, wie die Fahrzeugdichte und -geschwindigkeit eines Spielmodus ist. So kann ein versuch schnell enden, wenn das erste Fahrzeug unmittelbar den Weg des Huhns kreuzt. Dies zeigt die Unsicherheit und Variabilität dieser Strategie. Diese hängt dementsprechen stark von externen, nicht kontrollierbaren Faktoren ab.

2.3 Deep Q-Learning

2.3.1 Grundlagen

Um den Reinforcement-Learning-Algorithmus zu implementieren wurde sich in dieser Ausarbeitung für das Deep Q-Learning entschieden. Bei diesem Ansatz werden die Eigenschaften des Q-Learnings mit denen von Tiefen Neuronalen Netzwerken verbunden.

Q-Learning basiert auf der Aktualisierung von Q-Werten für Zustand-Aktions-Paare. Diese Q-Werte werden iterativ mithilfe der Bellman-Gleichung angepasst, um optimale Entscheidungsstrategien zu erlernen. [9]

Erweitert wird diese Methode durch das Deep Q-Learning welches die Q-Funktion mittels eines neuronalen Netzes approximiert, wodurch die tabellarische Speicherung der Q-Werte ersetzt wird. Außerdem wird durch die Einführung von Experience Replay und eines Zielnetzwerks die Stabilität und Effizienz des Lernprozesses im Vergleich zu klassischem Q-Learning erheblich verbessert. Experience Replay ist eine Methode, bei der Agenten vergangene Erfahrungen in einem Puffer speichern und daraus zufällig Stichproben entnehmen, um das neuronale Netzwerk zu trainieren. Das Zielnetzwerk ist eine separate Kopie des Q-Netzwerks, welche verwendet wird, um die Q-Werte zu berechnen und so die Instabilität während des Trainings zu reduzieren, indem die Gewichte dieses Netzwerk nur in größeren Abständen aktualisiert werden. Die Wahl und Strukturierung der Rewards spielt ebenfalls eine entscheidende Rolle, da sie die Anreize definieren, welche die Agenten zur optimalen Handlung führen. [10]

2.3.2 Netzwerkarchitektur

Die Architektur des neuronalen Netzwerks ist sequenziell aufgebaut und umfasst drei verborgene Schichten. Der Eingabevektor mit der Form (13, 1) wird durch drei vollständig verbundene Schichten geleitet, die jeweils 256, 128 und 64 Neuronen enthalten. Jede dieser Schichten verwendet die ReLU-Aktivierungsfunktion. Die Ausgabeschicht ist ebenfalls vollständig verbunden und besteht aus drei Neuronen, entsprechend der Anzahl der möglichen Aktionen. Hierbei wird eine lineare Aktivierungsfunktion verwendet, um die Q-Werte zu approximieren. Im Rahmen der Greedy-Strategie wird der höchste Q-Wert ausgewählt, um die nächste Aktion vorherzusagen.

2.3.3 Rewards

Die Rewards, wie in Tabelle 2.2 dargestellt, wurden so gewählt, dass ein Aufwärtsbewegen des Agenten belohnt wird, um das Ziel schnellstmöglich zu erreichen. Dafür

Tabelle 2.2: Deep Q-Learning Rewards

Bedingung	**Reward**
Verharren	-0.05
Aufwärtsbewegen	0.5
Abwärtsbewegen	-0.2
Reward je erreichten Checkpoint	0.01
Ziellinie erreichen	1
Kollision	-1

wurde ein positiver Reward für die Aufwärtsbewegung vergeben, während eine Abwärtsbewegung mit einem negativen Reward bestraft wurde. Auch das Verharren wird leicht bestraft, da das Ziel darin besteht, die Straße so schnell wie möglich zu überqueren. Zusätzlich werden positive Rewards für das Erreichen von Checkpoints vergeben, die sich an jeder fünften Y-Koordinate befinden, um das Aufwärtsbewegen weiter zu belohnen. Das Erreichen der Ziellinie wird mit einem großen Reward belohnt, während eine Kollision mit einem großen negativen Reward bestraft wird. Dies soll dazu führen, dass der Agent lernt, den Fahrzeugen auszuweichen.

Die Rewards wurden so angepasst, dass sie im positiven Fall 1 und im negativen Fall -1 nicht überschreiten. Diese Methode, bekannt als Reward Clipping, soll das Training stabilisieren, indem die Skala der Fehlerableitungen im neuronalen Netzwerk begrenzt wird. [10]

3 Anwendung

3.1 Baseline

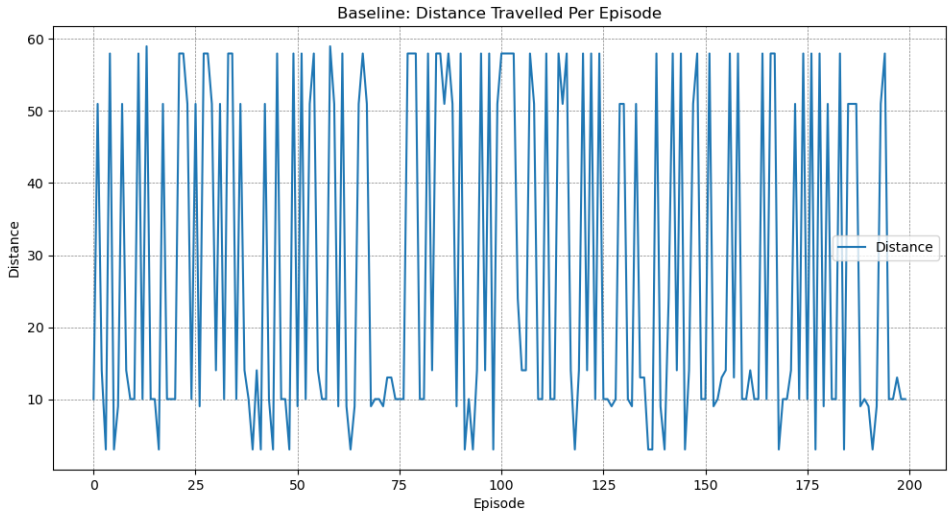

Abbildung 3.1: Zurückgelegte Distanz bei Anwendung der Baseline-Methode. Die zurückgelegte Strecke variiert stark und ist extrem vom Glück abhängig. Erwartungsgemäß ist kein Lernverhalten des Agenten erkennbar.

Die Anwendung der Baseline-Methode, dargestellt durch Abbildung 3.1, zeigt, dass ein Agent, der sich nur aufwärts bewegt, durchaus erfolgreich sein kann. Es ist zu sehen, dass der Agent mehrmals die Ziellinie bei der Distanz 58 erreichen konnte. Andererseits kann der Agent in vielen Fällen eine Distanz von 20 nicht überschreiten. Diese Variabilität hängt mit der unterschiedlichen Statik der Spielmodi zusammen. Es wird deutlich, dass einige Spielmodi besser für die Baseline-Methode geeignet sind als andere. Tabelle 3.1 zeigt, wie die Baseline-Methode für die verschiedenen Spielmodi im Detail abschneidet. Es ist zu beachten, dass durch eine konstante Statik innerhalb der Spielmodi die erreichte Entfernung des Agenten der Baseline-Methode für einen Spielmodus über alle Episoden gleich bleibt.

Tabelle 3.1: Zurückgelegte Distanz des Agenten der Baseline-Methode nach Spielmodi unter statischen Voraussetzungen

Modus	**Distanz (y-Koordinate)**
Modus 0	58
Modus 1	10
Modus 2	10
Modus 3	51
Modus 4	58
Modus 5	13
Modus 6	9
Modus 7	10

3.2 Deep Q-Learning

Die Anwendung des Deep Q-Learning, zeigt wie in Abbildung 3.2 dargestellt, dass der Agent mit fortschreitender Zeit dazulernt. Durch das Reward Clipping ist ein stetiger Lernerfolg zu erkennen.

Weiter ist festzustellen, dass eine Strategie erlernt wird mit welcher der Agent einigermaßen zuverlässig die Ziellinie überqueeren kann. Mit Blick auf das Verhalten des Agenten bei Episode 300 zeigt sich, dass der Agent eine Strategie ähnlich zur Baseline-Methode erlernt hat. In der überwiegenden Anzahl der Fälle entscheidet sich der Agent dazu aufwärts zu gehen. Außerdem hat der Agent gelernt in einigen Fällen zu Beginn des Spiels abzuwarten, um dann nur aufwärts zu gehen, um so das Ziel zu erreichen.

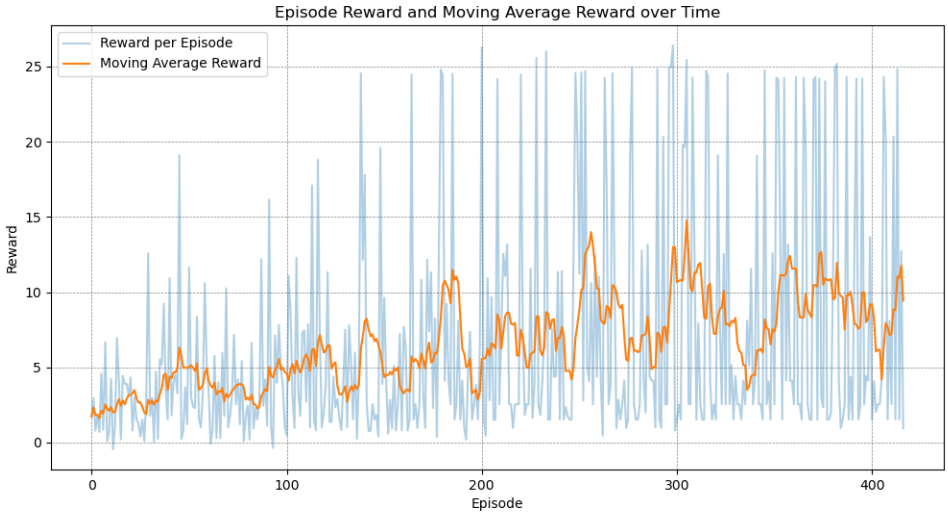

Abbildung 3.2: Reward pro Episode des Deep Q-Learnings mit 3 Aktionen. Der Moving Average Reward über die letzten 10 Episoden wird durch die orange Linie dargestellt. Der Reward in der aktuellen Episode wird in blau gezeigt. Es stetiges Lernverhalten ist bis zu Episode 200 zu erkennen. Danach kann der Agent keinen signifikanten Lernerfolg mehr verzeichnen.

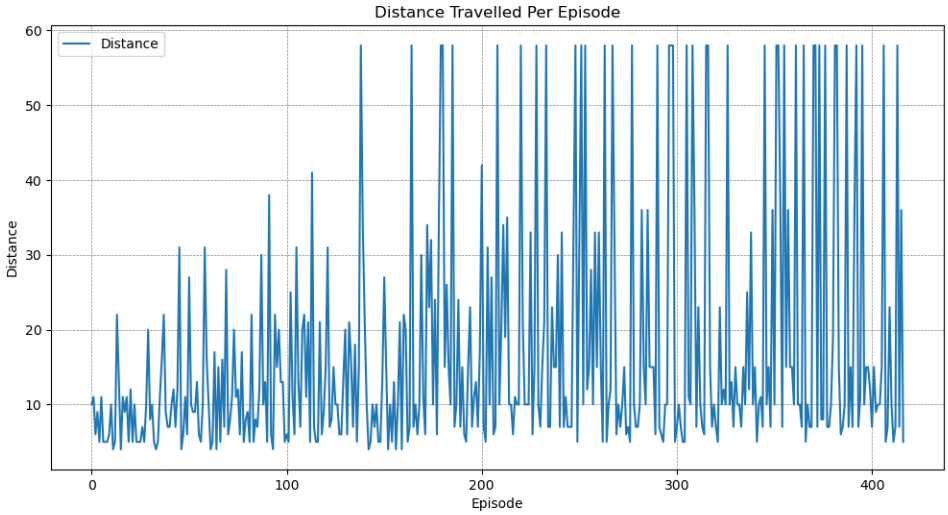

Abbildung 3.3: Zurückgelegte Distanz pro Episode des Deep Q-Learnings. Das erste Mal erreicht der Agent die Ziellinie bei Distanz (y-Koordinate) um die Episode 150. Ab Episode 250 kann die Zielline relativ Erfolgreich erreicht werden. Es ist ein Lernerfolg zu beobachten.

4 Abschluss

In dieser Ausarbeitung konnte gezeigt werden, dass mit dem Deep Q-Learning-Ansatz eine Strategie erlernt werden kann, die das Atari-Freeway-Spiel mit teilweisem Erfolg spielen kann. Besonders im Vergleich zur Baseline-Methode ist das trainierte Modell zuverlässiger. Auffällig ist, dass die Strategie des trainierten Modells der der Baseline-Methode ähnelt. Es ist festzustellen, dass die Baseline-Methode bereits eine gute Benchmark darstellte, welche durch das Modell nur mit Mühe übertroffen werden konnte. Das Modell hebt sich von der Baseline-Methode ab, indem es zögert, bevor es das erste Mal nach vorne geht, oder in seltenen Fällen auf der Spur stehen bleibt, um ein Fahrzeug abzuwarten. Allgemein hat der Agent aber erlernt, dass ihm das alleinige Aufwärtsbewegen den größten Reward gibt. Die hohe Variabilität bei den Ergebnissen des Modells kann zudem auf die unterschiedliche Spielmodi zurückzuführen sein. Es ist davon auszugehen, dass das Modell in einigen Modi besser zurecht kommt als in anderen.

Tabelle 4.1: Computing-Ressourcen

Ressource	Spezifikation
CPU	Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz
Arbeitsspeicher	8 GB
GPU	nicht Vorhanden

Die Herausforderung beim Training bestand darin, einen Trade-Off zwischen dem Aufwärtsbewegen und dem Vermeiden von Kollisionen zu finden. Dabei ist es nicht gelungen, den Agenten ausreichend zu belohnen, damit er Kollisionen mit hoher Wahrscheinlichkeit aktiv ausweicht. Hier könnte eine kleinere Bestrafung für das Abwärtsbewegen und das Verharren sich positiv auswirken. Außerdem könnte eine höhere Bestrafung für Kollisionen den Agenten dazu bringen, diese zu vermeiden.Eine zusätzliche Erweiterung könnte darin bestehen, den Zustand als Graustufen- oder RGB-Bild auszugeben. Solche Bilder enthalten mehr Informationen als eine einfache RAM-Darstellung. Dies könnte dem Netzwerk helfen, in einigen Situationen bessere Entscheidungen zu treffen. Es ist zu beachten, dass für diese Methoden voraussichtlich

mehr Ressourcen und eine längere Trainingszeit benötigt werden. Auch eine Betrachtung des gesamten RAM-Arrays könnte das Ergebnis positiv beeinflussen.

Die Komplexität und das Training des Modells wurden in diesem Projekt durch die in Tabelle 4.1 angegebenen Ressourcen begrenzt.

Literaturverzeichnis

[1] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht und M. Bowling, „Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents," *Journal of Artificial Intelligence Research*, Jg. 61, S. 523–562, März 2018, ISSN: 1076-9757. DOI: 10.1613/jair.5699. Adresse: http://dx.doi.org/10.1613/jair.5699.

[2] V. Mnih, K. Kavukcuoglu, D. Silver et al., „Human-level control through deep reinforcement learning," *Nature*, Jg. 518, Nr. 7540, S. 529–533, Feb. 2015, ISSN: 1476-4687. DOI: 10.1038/nature14236. Adresse: http://dx.doi.org/10.1038/nature14236.

[3] M. G. Bellemare, Y. Naddaf, J. Veness und M. Bowling, „The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, Jg. 47, S. 253–279, Juni 2013, ISSN: 1076-9757. DOI: 10.1613/jair.3912. Adresse: http://dx.doi.org/10.1613/jair.3912.

[4] *AtariAge - Atari 2600 Manuals (HTML) - Freeway (Activision) — atariage.com*, https://atariage.com/manual_html_page.php?SoftwareLabelID=192, [Accessed 07-07-2024].

[5] jumpropeman, *Freeway (Atari 2600) - The Game Hoard — thegamehoard.com*, https://thegamehoard.com/2021/03/20/freeway-atari-2600/, [Accessed 07-07-2024].

[6] *Gymnasium Documentation — gymnasium.farama.org*, https://gymnasium.farama.org/environments/atari/freeway/, [Accessed 07-07-2024].

[7] A. Anand, E. Racah, S. Ozair, Y. Bengio, M.-A. Côté und R. D. Hjelm, „Unsupervised State Representation Learning in Atari," *arXiv preprint arXiv:1906.08226*, 2019.

[8] D. Mayr, *FreewayGame*, https://github.com/DionisiusMayr/FreewayGame, Accessed: 2024-07-09, 2024.

[9] C. J. C. H. Watkins, *Learning from delayed rewards*, 1989.

[10] V. Mnih, K. Kavukcuoglu, D. Silver et al., „Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.