EVIDEN

DHBW
Duale Hochschule
Baden-Württemberg
Mannheim

Duale Hochschule Baden-Württemberg Mannheim

Projektarbeit

Entwicklung eines Reinforcement Learning-Agenten für die CarRacing-Umgebung

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser(in):	Fabian Banovic
Matrikelnummer:	6787994
Firma:	Eviden
Kurs:	WWI21DSB
Bearbeitungszeitraum:	08.05.2024 – 26.07.2024

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

RL	Reinforcement Learning
DQN	Deep Q-Network

1 Einführung

1.1 Motivation

Wie bereits der Titel der Arbeit verrät, wurde sich für das Gym-Environment Car Racing entschieden, um einen Reinforcement Learning (RL) Agenten zu programmieren. Die Entscheidung fiel sowohl aufgrund von persönlichen Präferenzen, als auch wegen der Einfachheit des Spiels. Mein Interesse an der Umgebung wurde durch einige Videos zum Thema Car Racing und RL geweckt. Besonders interessant fand ich dabei zu sehen, wie im Laufe des Trainings das Auto nicht nur weiter voran, sondern auch immer schneller wurde und damit beispielsweise das Driften erlernt hat. Zudem habe ich eine große Leidenschaft für Autos und schnelles Fahren, was die Arbeit mit dieser Umgebung besonders sinnvoll gemacht hat.

1.2 Umgebung

Die CarRacing-v2-Umgebung bietet eine Rennsimulation an, die es ermöglicht meine Interessen mit RL-Algorithmen zu kombinieren und zu testen. Die Umgebung zeichnet sich durch verschiedene technische Merkmale aus, die sie besonders geeignet für praktische Anwendungen machen. Der State-Space der CarRacing-v2-Umgebung besteht aus einer 96x96x3 RGB-Bildsequenz, die dem Agenten als Eingabe dient und das visuelle Feedback repräsentiert, das zur Steuerung des Fahrzeugs genutzt wird. Der Action-Space erlaubt es dem Agenten, kontinuierliche Aktionen auszuführen, die aus Lenken, Gas geben und Bremsen bestehen, als auch diskrete Aktionen, die wiederum aus nichts tun, nach links lenken, nach rechts lenken, Gas geben und aus Bremsen bestehen. Die Belohnungsfunktion in dieser Umgebung basiert auf der zurückgelegten Strecke. Der Agent erhält positive Belohnungen für das besuchen eines neuen Feldes auf der Strecke und eine negative Belohnung von -0.1 für jeden Schritt, als auch eine negative Belohnung von -100, wenn der Agent aus dem Spielfeld heraus fährt. Der Agent startet immer in der Mitte einer zufällig generierten Straße. Eine Episode endet, wenn der Agent das Spielfeld verlässt oder eine bestimmte Rundenanzahl erreicht ist. Der Terminal State kann also auf zwei Arten erreicht werden. (Vgl. *Car Racing - Gym Documentation* 2024)

2 Reinforcement Learning Agent

2.1 Architektur

Die Architektur wurde inspiriert von einem GitHub-Repo, welches eine ältere Version des Spiels genutzt hat. (Vgl. Wu 2024) Die Architektur des RL-Agenten basiert auf einem Deep Q-Network (DQN), das darauf abzielt, optimale Strategien für die Steuerung des Fahrzeugs in der CarRacing-v2-Umgebung zu erlernen. Dieser Ansatz verwendet ein neuronales Netz, dass darauf ausgelegt ist, die Obersvations des Agenten, die Bilddaten, in Aktionen umzuwandeln. Das Netzwerk besteht dabei aus zwei Convolutional Layers und zwei Fully Connected Layers. Da der DQN für einen diskreten Actionspace verwendet wird, wurde dafür das Environment hinsichtlich der Variable continuous auf False gesetzt, um dies zu ermöglichen. Der Agent speichert seine Erfahrungen in einem Replay Memory und verwendet eine Epsilon-Greedy-Strategie, um im Laufe des Trainings die Exploitation zu verringern und den Fokus langsam auf Exploitation zu legen. (Vgl. Wu 2024) Der Agent benutzt als Optimizer, den in der Vorlesung besprochenen RMSprop und als Loss den Huberloss. Desweiteren wurde der Agent so eingerichtet, dass er sowohl auf einer CPU als auch ein einer GPU laufen kann, falls diese vorhanden ist, um den Trainingsprozess zu beschleunigen.

2.2 Training

Der Trainingsprozess beginnt mit der Initialisierung der Umgebung und des Agenten, wobei wichtige Parameter wie Seed, Episodenanzahl, Speichergröße sowie Gamma, Epsilon und die Lernrate, als auch noch weitere Variablen, festgelegt werden. Neben einer Helperfunction, die Überprüft, ob das Auto die Straße verlassen hat oder nicht, wird außerdem für jeden gestarteten Trainingprozess ein neuer Ordner angelegt, in welchem alle Daten, die während des Trainings gesammelt worden sind, gespeichert werden. Der Trainingsschritt an sich ist sehr simpel. Begonnen wird, indem das Environment zurückgesetzt wird. Im nächsten Schritt werden immer vier Frames, also vier Beobachtungen, gestapelt. Dabei werden die Observations zunächst auf Schwarz-weiß geändert, um die Ausgabe von 96x96x3 auf 96x96

zu bringen. Im letzten Schritt werden vier von diesen aneinander gehangen und dem Modell übergeben. Basierend darauf wird die Act-Funktion des Models aufgerufen, um eine Aktion zu bekommen. Da mehrere Frames zusammengelegt wurden, wird die Aktion für jede von diesen ausgeführt. Dabei wird der Reward und der nächste State getrackt und geschaut ob es zu einem Abbruch kommt. Sollte dies nicht der Fall sein, wird der Agent extra belohnt, sofern dieser Gas gegeben hat oder bestraft, wenn bestimmte Fälle eingetreten sind, wie außerhalb der Fahrbahn sein, oder zu oft negative Rewards bekommen. Schließlich werden diese Erfahrungen in dem Replay Buffer gespeichert und das Training des Models gestartet. Ist das Training vorbei, wird sowohl das Model in dem erstellten Ordner gespeichert, als auch der Reward und der Loss für die Visualisierung.

2.3 Optimierung

Begonnen wurde mit den Standardeinstellungen und der Konfiguration, die in der Quelle festgelegt wurde. Diese Konfiguration führte bereits zu einem ersten durchschnittlichen Ergebnis (siehe Abbildung 2.1). Um die Performance des Agents zu verbessern, wurde zunächst die Architektur angepasst. Anfangs war der Adam-Optimizer sowie die Mean Squared Error-Lossfunktion gewählt worden. Um die Lernqualität zu optimieren, wurden diese Komponenten auf RMSprop als Optimizer und Huber Loss als Lossfunktion umgestellt, entsprechend wie in der Vorlesung. Die Veränderung kann in Abbildung 2.2 gesehen werden.

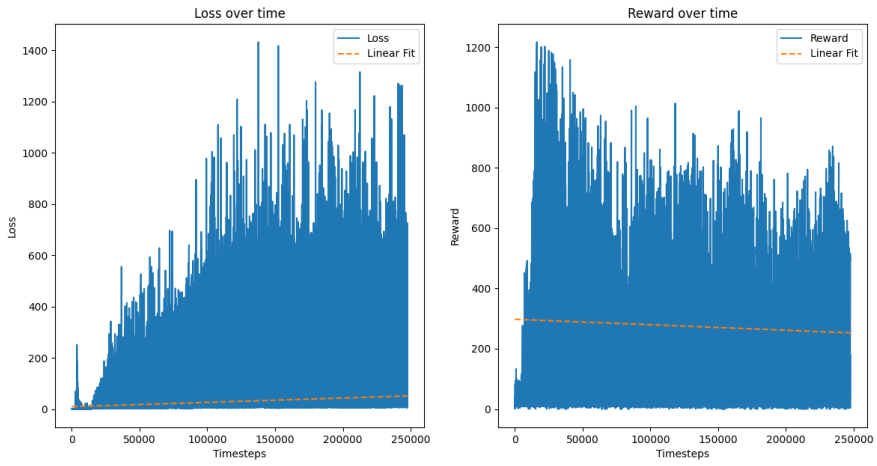

Abbildung 2.1: Loss und Reward über den Trainingsverlauf ohne Konfigurationen

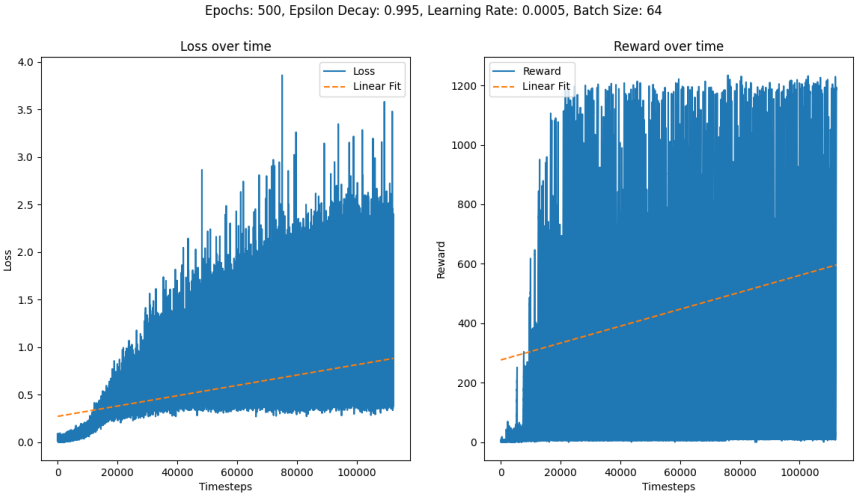

Abbildung 2.2: Loss und Reward über den Trainingsverlauf mit einer geänderten Lossfunktion und einem Optimizer

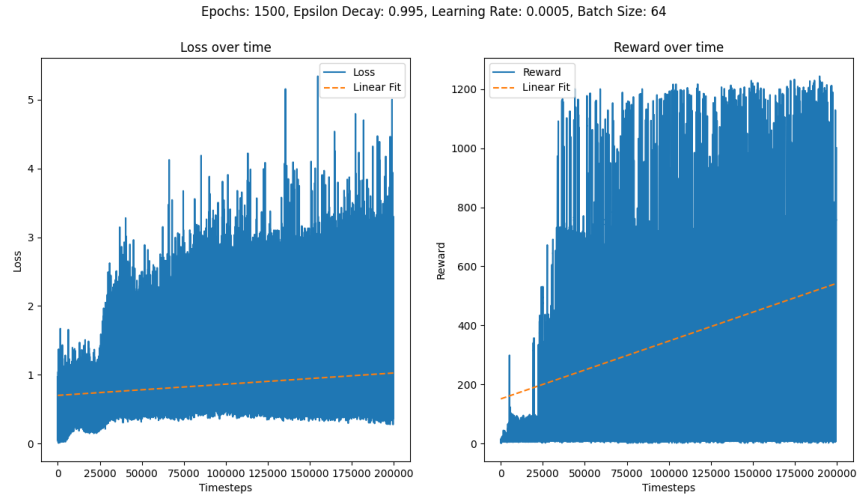

Abbildung 2.3: Loss und Reward über den Trainingsverlauf mit der Funktion, die die Position des Fahrzeugs auf der Strecke überprüft

Basierend auf dieser neuen Ausgangslage wurden weitere kleinere Anpassungen vorgenommen, wie etwa die Erhöhung der Episodenanzahl, was zu deutlich besseren Ergebnissen führte. Je länger der Agent trainiert wird, desto mehr Erfahrungen kann er sammeln und sich verbessern. Um die Modelle auch visuell vergleichen zu können, wurden sie zum Testen geladen und einzelne Durchläufe aufgenommen. Diese Videos sind teilweise auf GitHub verfügbar. (Vgl. Fabian 2024) In den Videos zeigte sich, dass das Modell zwar weit kommt,

aber häufig den Fahrbahnrand überschreitet.

Um dieses Problem zu lösen, wurden die vom Autor festgelegten Werte parametrisiert und getestet, allerdings ohne signifikante Verbesserungen. Als letzten Verbesserungsschritt wurde eine eigene Funktion implementiert, die den Agenten sofort bestraft und die Episode beendet, sobald er die Fahrbahn verlässt. Da das Ziel des Agenten darin besteht, auf der Straße zu bleiben, führte das Hinzufügen dieser Funktion zu dem in Abbildung 2.3 gezeigten Ergebnis. Hierbei lassen sich nur kleine Veränderungen erkennen, wie eine flachere Trendlinie beim Loss und eine steilere beim Reward, was jedoch mit der Episodenanzahl zu begründen ist. Eine deutliche Verbesserung hingegen lässt sich in den Testvideos sehen. Das Modell hat gut gelernt, dass das Überschreiten der Fahrbahn zu einem negativen Reward führt, weshalb es nahezu perfekt auf der Fahrbahn bleibt.

3 Fazit

3.1 Bewertung

Das zuletzt trainierte Modell zeigt insgesamt eine gute Leistung und kann die Fahraufgabe in der CarRacing-v2-Umgebung erfolgreich bewältigen. Es hat sich gezeigt, dass das Modell in der Lage ist, die Fahrbahn weitgehend zu folgen und angemessene Entscheidungen zu treffen. Ein wesentlicher Faktor für die Verbesserung des Fahrstils war die Implementierung der Überprüfungsfunktion, die bei Verlassen der Fahrbahn einen negativen Reward vergibt. Diese Änderung hat maßgeblich dazu beigetragen, das Verhalten des Modells zu optimieren, auch wenn es dadurch mehr Episoden benötigt, um hohe Rewards zu erreichen. Zusätzlich hat die Anpassung der Loss-Funktion und des Optimizers eine entscheidende Rolle gespielt. Die Umstellung auf den Huber Loss und den RMSprop-Optimizer, wie in der Vorlesung empfohlen, hat zur Stabilität und Effizienz des Trainings beigetragen. Diese Entscheidungen basierten auf den in der Vorlesung dargestellten Vorteilen und haben sich in der Praxis als vorteilhaft erwiesen.

3.2 Kritische Reflexion

Die Verbesserungen des Modells beruhten ausschließlich auf Entscheidungen, die anhand der Auswertung der Ergebnisse basierten. Um noch bessere Ergebnisse zu erzielen, wäre ein Hyperparametertuning, wie beispielsweise mit Optuna (*Optuna - A hyperparameter optimization framework* 2024), sehr hilfreich gewesen. Durch systematisches Variieren und Optimieren der Hyperparameter wie Lernrate, Batch-Größe, und Epsilon-Decay, sowie einigen anderen Parametern, könnte die Leistung des Modells weiter verbessert werden.
Ein weiterer Aspekt ist, dass alternative beziehungsweise fortgeschrittenere Architekturen genutzt werden können, wie in etwa Double-DQN oder auch Dueling-DDQN. Diese Architekturen bieten erweiterte Mechanismen zur Stabilisierung des Lernprozesses und zur Verbesserung der Entscheidungsfindung des Agenten, was zu einer höheren Leistung und Effizienz führen kann.

Literaturverzeichnis

Car Racing - Gym Documentation (2024). URL: https://www.gymlibrary.dev/environments/box2d/car_racing/ (besucht am 07. 07. 2024).

Fabian (5. Juli 2024). *FabianDHBW/RL_CarRacing*. original-date: 2024-06-29T18:07:02Z. URL: https://github.com/FabianDHBW/RL_CarRacing (besucht am 07. 07. 2024).

Optuna - A hyperparameter optimization framework (2024). Optuna. URL: https://optuna.org/ (besucht am 07. 07. 2024).

Wu, Andy (1. Juli 2024). *andywu0913/OpenAI-GYM-CarRacing-DQN*. original-date: 2020-04-19T17:30:31Z. URL: https://github.com/andywu0913/OpenAI-GYM-CarRacing-DQN (besucht am 07. 07. 2024).