Duale Hochschule Baden-Württemberg Mannheim

Portfolioarbeit zum Thema

Vergleichende Analyse von Deep Q-Network und Proximal Policy Optimization Algorithmen für autonome Fahrsysteme in der Highway-v0 Simulationsumgebung

Fakultät Wirtschaft

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser:	Jan Rüdt
Matrikelnummer, Kurs:	1737304, WWI20DSA

Modul:	Aktuelle Data Science Entwicklungen II
Fach:	Reinforcement Learning
Dozentin:	Janina Patzer

Abstract

Diese Arbeit präsentiert eine vergleichende Analyse der Deep Q-Network (DQN) und Proximal Policy Optimization (PPO) Reinforcement Learning Algorithmen im Kontext autonomer Fahrsysteme. Die Evaluation erfolgt innerhalb der Highway-v0 Simulationsumgebung, bereitgestellt von der Farama Foundation. Dabei steht DQN für einen wertebasierten Ansatz, während PPO ein Policy-basierter Ansatz ist. Beide Algorithmen werden auf ihre Leistungsfähigkeit hin untersucht, indem sie in einer künstlichen Umgebung, die eine vereinfachte Autobahn darstellt, angewendet werden. Die Entwicklung der Agenten erfolgt in der speziell entwickelten "highway-fast-v0" Umgebung und wird durch 20.000 Trainingsschritte ausgeführt. Die Leistungsbewertung erfolgt durch Vergleich der durchschnittlichen Episodenlänge und Belohnung pro Episode für beide Agenten. Die Ergebnisse deuten darauf hin, dass der PPO-Agent in dieser speziellen Umgebung eine bessere Leistung erzielt. Allerdings wurden dabei nur bestimmte Kennzahlen und Eigenschaften berücksichtigt und die Modelle stellen eine starke Vereinfachung der hochkomplexen Thematik des autonomen Fahrens dar. Daher sind weitere Untersuchungen und Anpassungen nötig, um die Ergebnisse zu verbessern und einen realitätsnahen Einsatz zu ermöglichen.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS

1 Einleitung

In der vorliegenden Arbeit wird eine vergleichende Analyse zweier prominenter Reinforcement Learning Algorithmen, dem Deep Q-Network (DQN) und der Proximal Policy Optimization (PPO), im Kontext autonomer Fahrsysteme durchgeführt. Die Evaluierung dieser Algorithmen erfolgt in der spezifischen Simulationsumgebung Highway-v0, die von der Farama Foundation bereitgestellt wird. Der dazugehörige Code wird in diesem GitHub-Repository dokumentiert: https://github.com/ruedel/RL-Agent-Comparison/

2 Beschreibung der Umgebung

Die Highway-v0-Umgebung ist eine speziell entwickelte Simulationsumgebung, die für das Training von autonomen Fahrzeugen mit Hilfe von Multi-Agent Reinforcement Learning Algorithmen (MARL) konzipiert wurde. Sie bietet eine vereinfachte Darstellung einer Autobahn, auf der mehrere Fahrzeuge interagieren. In dieser Umgebung ist jedes Fahrzeug ein Agent, der seine Geschwindigkeit und Fahrspur kontrolliert. Der Agent kann eine Reihe von Aktionen ausführen, darunter das Lenken des Fahrzeugs, das Beschleunigen oder Verlangsamen und das Wechseln der Fahrspuren. Die Belohnungsfunktion belohnt den Agenten dafür, dass er auf der Straße bleibt, eine bestimmte Geschwindigkeit hält und Kollisionen vermeidet (Dinneweth et al. 2022, S. 6; Farama Foundation 2023).

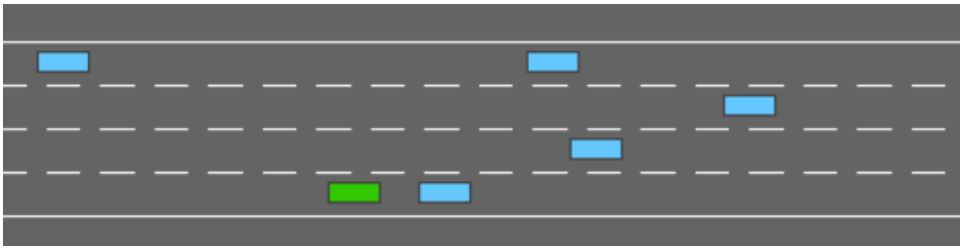

Abbildung 1: Die Highway-v0-Umgebung

Die Architektur der Highway-v0-Umgebung ist so konzipiert, dass sie die Komplexität der realen Welt auf ein handhabbares Niveau reduziert, ohne die wesentlichen Aspekte des autonomen Fahrens zu verlieren. Sie besteht aus einer Reihe von Fahrspuren und Fahrzeugen, die sich mit unterschiedlichen Geschwindigkeiten bewegen. Jedes Fahrzeug wird durch seine Position, Geschwindigkeit und Fahrspur repräsentiert. Die Fahrzeuge interagieren miteinander und mit der Umgebung, um Kollisionen zu vermeiden und ihr Ziel zu erreichen (Leurent 2018).
Die Highway-v0-Umgebung bietet mehrere Vorteile. Sie ermöglicht das Training von autonomen Fahrzeugen in einer kontrollierten und sicheren Umgebung, ohne dass ein reales Fahrzeug oder eine reale Infrastruktur benötigt werden. Außerdem verwendet sie MARL-Algorithmen, welche die Fahrzeuge dazu befähigen, kooperative oder wettbewerbsfähige Verhaltensweisen zu erlernen und robustere und überzeugendere Strategien zu entwickeln (Schmidt et al. 2022, S. 5–6).
Die Wahl der Highway-v0-Umgebung wurde durch mehrere Faktoren bestimmt. Sie bietet eine ausreichende Komplexität, um realistische Fahrbedingungen zu simulieren, ohne übermäßig komplex oder rechenintensiv zu sein. Zudem ist sie besonders effektiv durch die Verwendung der MARL-Algorithmen für das Training von autonomen Fahrzeugen geeignet. Ein weiterer Punkt ist, dass die gute Dokumentation und weite Verbreitung die Implementierung und das Debugging erleichtern. Weitere populäre Umgebungen sind beispielsweise die CARLA-Umgebung, die auf dem Unreal Engine basiert oder das Flow-Framework. Diese Umgebungen bieten eine höhere Komplexität und mehr Realismus, sind aber auch rechenintensiver und schwieriger zu handhaben (Dinneweth et al. 2022, S. 6).

Das autonome Fahren ist heute von großer Bedeutung, da es das Potenzial hat, die Sicherheit und Effizienz des Verkehrs zu verbessern, die Umweltauswirkungen des Verkehrs zu reduzieren und die Mobilität für Menschen, die nicht selbst fahren können, zu erhöhen. Darüber hinaus ist es ein aktives Forschungsgebiet mit vielen offenen Fragen und Herausforderungen, was es zu einem attraktiven Thema für wissenschaftliche Untersuchungen macht.

3 Entwicklung der Reinforcement Learning Agenten

Die Highway-v0-Umgebung kann als ein Markov-Entscheidungsprozess (MDP) konzeptualisiert werden, in dem der Zustand durch die Position und Geschwindigkeit des Agentenfahrzeugs sowie der anderen Fahrzeuge repräsentiert wird. Die Aktionen des Agenten werden durch seine Fahrmanöver definiert, während die Belohnungen durch eine spezifische Belohnungsfunktion gegeben werden (Elallid et al. 2022, S. 7381).

In dieser Arbeit werden die beiden Reinforcement Learning Algorithmen Deep Q-Network (DQN) und Proximal Policy Optimization (PPO) im beigefügten Jupyter Notebook angewendet, um zu bestimmen, welcher der vorgestellten Algorithmen die beste Leistung in der Highway-v0-Simulationsumgebung erzielt. Die Entwicklung hat in Google Colab Pro stattgefunden, da dies auf einer Linux-Distribution ausgeführt wird, was eine Voraussetzung für die gymnasium Bibliothek ist. Die Verwendung von Google Colab Pro ermöglicht es zudem, flexibel Rechenleistung hinzuzubuchen.

Der DQN-Algorithmus, ein wertebasierter Ansatz, zielt darauf ab, eine Q-Funktion zu konstruieren, die den erwarteten Nutzen jeder Aktion in jedem Zustand abschätzt. Dies ermöglicht dem Agenten, Verhaltensweisen zu implementieren, die zum besten erwarteten Ergebnis führen. Im Gegensatz dazu ist der PPO-Algorithmus ein Policy-basierter Ansatz, der direkt eine Strategie optimiert, die Aktionen auswählt. Er verwendet eine spezielle Art von objektiver Funktion, die dazu ermutigt, die aktuelle Strategie nicht zu stark von der vorherigen Strategie abzuweichen. Dieser Ansatz zielt darauf ab, eine stabile Lernleistung zu gewährleisten (Dinneweth et al. 2022, S. 2–3).
Auch eine spezielle Variante des DQN-Algorithmus, die ein Convolutional Neural Network (CNN) zur Verarbeitung von Bildbeobachtungen verwendet, wäre hier denkbar. Sie ist besonders nützlich in Umgebungen, in denen visuelle Informationen für die Entscheidungsfindung des Agenten von Bedeutung sind. Durch die Verwendung eines CNN kann der Agent visuelle Eingaben verarbeiten und auf dieser Grundlage Entscheidungen treffen. Diese Variante ist jedoch sehr rechenintensiv und schwierig zu implementieren, weshalb sie an dieser Stelle nicht weiter berücksichtigt wird (Elallid et al. 2022, S. 7370).
Um die beiden Algorithmen anwenden zu können, werden zunächst die erforderlichen Bibliotheken installiert und Abhängigkeiten importiert. Die Bibliotheken umfassen unter anderem highway-env, stable-baselines3, tensorboardx, gymnasium und pyvirtualdisplay. Zudem wird das highway-env-Repository geklont. Im nächsten Schritt wird der Agent mit den beiden Algorithmen trainiert. Als Trainingsumgebung dient die highway-fast-v0-Umgebung, eine bis zu 15-mal schnellere Variante der Highway-v0-Umgebung (Farama Foundation 2023).
Zunächst wird der DQN-Agent trainiert, welcher ein neuronales Netzwerk verwendet, um die Q-Wert-Funktion zu approximieren. Die Q-Wert-Funktion ist eine Funktion, die den erwarteten zukünftigen Nutzen einer bestimmten Aktion in einem bestimmten Zustand abschätzt. Der Wiederholungspuffer speichert frühere Erfahrungen des Agenten, um die Datenkorrelation zu minimieren. Als Batchgröße wird 32 bestimmt, der Agent durchläuft ein Training von 20.000 Schritten. Diese Schrittanzahl wurde gewählt, da sie einen sinnvollen Kompromiss aus Lernfähigkeit und Rechenbedarf darstellt. Als Lernrate empfiehlt die Farama Foundation 5e-4 für den DQN-Agenten und 3e-4 für PPO-Agenten (Farama Foundation 2023).

Anschließend wird der PPO-Agent trainiert. Dieser verwendet eine Policy-Gradientenmethode für das Reinforcement Learning. Es verwendet eine Ersatzziel-Funktion und fügt der Ziel-Funktion einen Strafterm hinzu, um zu verhindern, dass die neue Politik zu stark von der alten Politik abweicht. Für das Training des PPO-Agenten werden die gleichen Hyperparameter wie für den DQN-Agenten verwendet, um die Ergebnisse miteinander vergleichen zu können.
Um die Lernleistung des DQN-Agenten beispielhaft visualisieren zu können, werden 30 Episoden des Trainingsprozesses als Videofrequenz dargestellt. Hierfür werden die Hilfsprogramme record_videos und show_videos verwendet. Es ist deutlich erkennbar, dass sich die Trainingsleistung des Agenten bereits nach wenigen Episoden wesentlich verbessert.
Für die Vergleichbarkeit der Trainingsergebnisse wird die durchschnittliche Episodenlänge und die durchschnittliche Belohnung pro Episode während des Trainings der beiden Agenten mithilfe von TensorBoard visualisiert. Eine Episode in Reinforcement Learning ist eine Sequenz von Zuständen, Aktionen und Belohnungen, die mit einem Anfangszustand beginnt und endet, wenn ein Endzustand erreicht wird oder eine maximale Anzahl von Schritten ausgeführt wurde. Die Länge einer Episode kann ein Indikator für die Leistung des Agenten sein, in dieser Trainingsumgebung beispielsweise die Zeit bis zu einem Zusammenstoß. Eine Belohnung ist das Signal, das der Agent erhält, um zu lernen, welche Aktionen gut oder schlecht sind. Eine höhere durchschnittliche Belohnung pro Episode deutet darauf hin, dass der Agent besser darin wird, Aktionen auszuführen, die hohe Belohnungen erzielen. Für den DQN-Agenten kann zusätzlich die Erkundungsrate grafisch dargestellt werden. Eine hohe Erkundungsrate bedeutet, dass der Agent verstärkt neue Aktionen ausprobiert, während eine niedrige Erkundungsrate bedeutet, dass der Agent eher die besten bekannten Aktionen ausführt (Jurvelin Olsson 2021, S. 6–7).

4 Bewertung der Ergebnisse und Herausforderungen bei der Umsetzung

Das Training der beiden Agenten nach jeweils 20.000 Schritten liefert die nachfolgenden Ergebnisse. Bezogen auf die durchschnittliche Episodenlänge zeigt der PPO-Agent eine bessere Leistung:

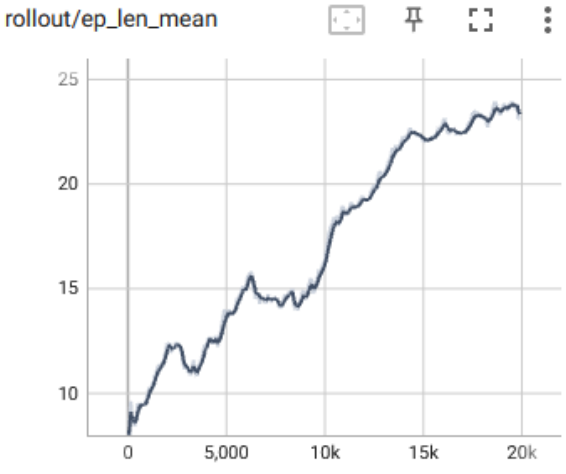

Abbildung 2: Durchschnittliche Episodenlänge des DQN-Agenten nach 20.000 Schritten

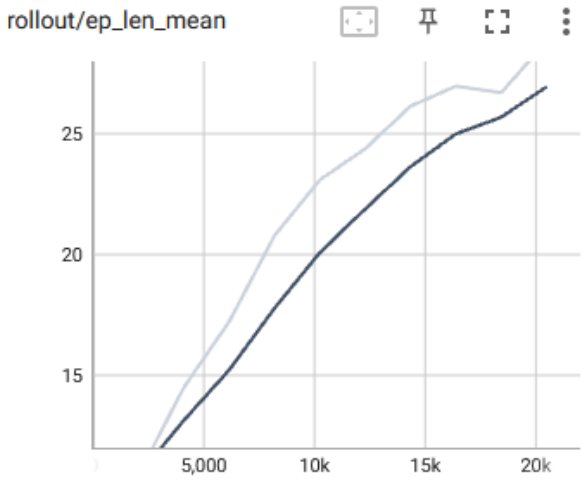

Abbildung 3: Durchschnittliche Episodenlänge des PPO-Agenten nach 20.000 Schritten

Während die durchschnittliche Episodenlänge des DQN-Agenten bei etwa 24 liegt, beträgt die des PPO-Agenten circa 27. Die Episodenlänge des DQN-Agenten steigt im Trainingsverlauf unterschiedlich schnell und sinkt sogar an einigen Stellen wieder. Im Gegensatz dazu steigt die Episodenlänge des PPO-Agenten fast linear an. Die erzielten Werte deuten darauf hin, dass in diesem Szenario der PPO-Agent bessere Leistungen, bezogen auf die durchschnittliche Episodenlänge, liefert.

Beim Vergleichen der durchschnittlichen Belohnung pro Episode fallen ähnliche Beobachtungen auf. Auch hier steigt die Belohnung des DQN-Agenten ungleichmäßiger als beim PPO-Agenten an. Der durchschnittliche Belohnungswert nach 20.000 Schritten liegt bei beiden Agenten etwa bei etwa 19, wobei der Wert beim PPO-Agenten geringfügig höher ist.

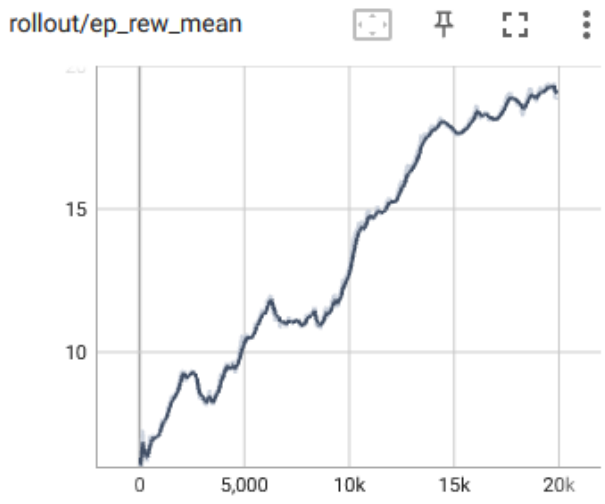

Abbildung 4: Durchschnittliche Belohnung pro Episode des DQN-Agenten nach 20.000 Schritten

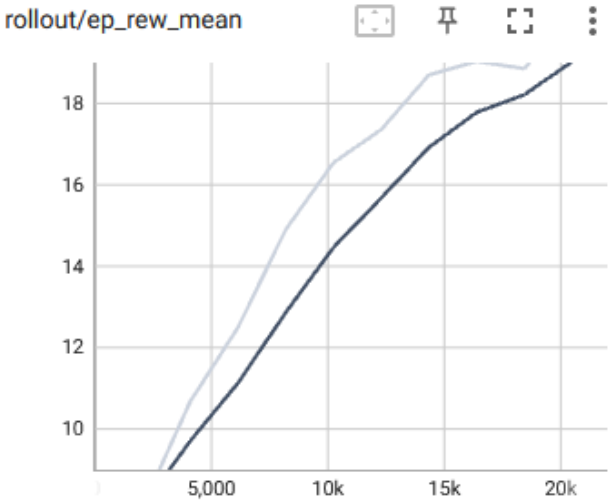

Abbildung 5: Durchschnittliche Belohnung pro Episode des PPO-Agenten nach 20.000 Schritten

Die Erkundungsrate des DQN-Agenten verringert sich im Laufe des Trainings stetig, bis sie bei etwa 14.500 Schritten den Wert 0 erreicht. Dies deutet darauf hin, dass der DQN-Agent zu Beginn des Trainings vermehrt neue Aktionen ausprobiert hat, während er gegen Ende des Trainings hauptsächlich auf bewehrte Aktionen zurückgreift, um seine Belohnung zu maximieren.

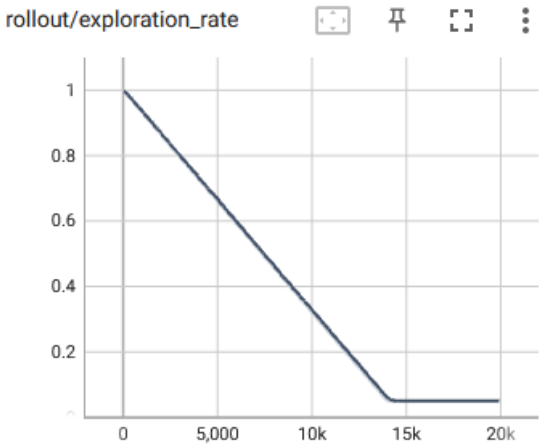

Abbildung 6: Erkundungsrate des DQN-Agenten nach 20.000 Schritten

Zusammenfassend lässt sich festhalten, dass der PPO-Agent in dieser Trainingsumgebung bessere Ergebnisse als der DQN-Agent erzielen kann, insbesondere bei der Betrachtung der durchschnittlichen Episodenlänge nach 20.00 Schritte. Die Werte lassen sich unter anderem durch Anpassen der Hyperparameter oder des Observationstyps verbessern. Auch durch die Verwendung komplexer Algorithmen wie dem DQN-Algorithmus in Verbindung mit einem Convolutional Neural Network oder einem Double DQN-Algorithmus lassen sich bessere Ergebnisse erzielen, allerdings würde dies auch einen höheren Rechenaufwand erfordern.

Die Ergebnisse können im Rahmen der gegebenen Beschränkungen als zufriedenstellend eingestuft werden. Es ist jedoch zu beachten, dass hier nur einige wenige Eigenschaften und Kennzahlen untersucht wurden. Aspekte wie beispielsweise die Robustheit gegenüber Veränderungen in der Umgebung oder die erforderliche Trainingszeit konnten in dieser Untersuchung nicht berücksichtigt werden. Auch stellen diese Modelle starke Vereinfachungen der hochkomplexen Thematik des autonomen Fahrens dar. In der Realität müssen deutlich mehr Einflüsse und Aspekte beachtet werden, um realitätsnahe Ergebnisse zu erhalten.

Im Zuge der Bearbeitung dieser Thematik traten verschiedene Herausforderungen auf. Unter anderem gab es Kompatibilitätsprobleme zwischen den einzelnen Bibliotheken sowie mit dem Betriebssystem. Beispielsweise ist Stable Baselines3 derzeit nicht kompatibel mit der gymnasium Bibliothek, weshalb eine angepasste Version von Stable Baselines3 verwendet wurde (Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR)). Auch wird, wie bereits erwähnt, gymnasium nur auf Linux and macOS unterstützt, weshalb eine lokale Ausführung nicht möglich war und auf eine Cloud-Ressource mit Linux-Distribution zurückgegriffen werden musste (OpenAI 2023). Aufgrund der langen Ausführzeiten und des hohen Rechenbedarfs wurde häufig die Laufzeitverbindung während der Ausführung getrennt, weshalb Google Colab Pro für längere Laufzeitverbindungen und zusätzlicher Rechenressourcen erworben wurde. Schließlich stellte die visuelle Darstellung der Ergebnisse eine Herausforderung dar, da die Trainingsergebnisse in einer Log-Datei mit einem sehr spezifischen Dateiformat gespeichert wurden, was die Extraktion der Daten und damit auch die grafische Darstellung erschwerte.

5 Literaturverzeichnis

Deutsches Zentrum für Luft- und Raumfahrt e. V. (DLR): Add Gymnasium support for Stable Baselines3. Online verfügbar unter https://github.com/DLR-RM/stable-baselines3/tree/master/scripts, zuletzt geprüft am 28.07.2023.

Dinneweth, Joris; Boubezoul, Abderrahmane; Mandiau, René; Espié, Stéphane (2022): Multi-agent reinforcement learning for autonomous vehicles. In: *Auton. Intell. Syst.* (2), S. 1–12. DOI: 10.1007/s43684-022-00045-z.

Elallid, Badr Ben; Benamar, Nabil; Hafid, Abdelhakim Senhaji; Rachidi, Tajjeeddine; Mrani, Nabil (2022): A Comprehensive Survey on the Application of Deep and Reinforcement Learning Approaches in Autonomous Driving. In: *Journal of King Saud University - Computer and Information Sciences* (34), S. 7366–7390. DOI: 10.1016/j.jksuci.2022.03.013.

Farama Foundation (2023): highway-env Documentation. Online verfügbar unter http://highway-env.farama.org/environments/highway/, zuletzt geprüft am 20.07.2023.

Jurvelin Olsson, Mikael (2021): MULTI-AGENT REINFORCEMENT LEARNING WITH APPLICATION ON TRAFFIC FLOW CONTROL. Online verfügbar unter http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1573441&dswid=1317.

Leurent, Edouard (2018): highway-env: An Environment for Autonomous Driving Decision-Making. GitHub. Online verfügbar unter https://github.com/eleurent/highway-env, zuletzt geprüft am 21.07.2023.

OpenAI (2023): The Gym toolkit. Online verfügbar unter https://github.com/openai/gym, zuletzt geprüft am 28.07.2023.

Schmidt, Lukas M.; Brosig, Johanna; Plinge, Axel; Eskofier, Bjoern M.; Mutschler, Christopher (2022): An Introduction to Multi-Agent Reinforcement Learning and Review of its Application to Autonomous Mobility. Online verfügbar unter https://arxiv.org/pdf/2203.07676.