Duale Hochschule Baden-Württemberg Mannheim

Projektarbeit

LunarLander-v2 mit Deep Q-Learning

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser(in):	Tim Kuhn
Matrikelnummer:	8284060
Firma:	RPTU
Kurs:	WWI-22DSB
Studiengangsleiter:	Prof. Dr. Bernhard Drabant
Bearbeitungszeitraum:	05.05.2024 – 31.07.2025

Inhaltsverzeichnis

1 Projektbericht: Deep Q-Learning mit LunarLander-v2 **1**
1.1 Einleitung 1
1.2 Modellarchitektur und Implementierung 1
1.3 Hyperparameter und Trainingsdetails 1
1.4 Belohnungsstruktur 2
1.5 Trainingsverlauf und Visualisierung 3
1.6 Evaluation 3
1.7 Videoerzeugung und Beispielausgabe 3
1.8 Anforderungen und Hinweise zur Ausführung 4
1.9 Fazit 4

Literaturverzeichnis **5**

1 Projektbericht: Deep Q-Learning mit LunarLander-v2

1.1 Einleitung

In diesem Projekt wurde ein Deep-Q-Learning (DQN[Mni+15]) Agent implementiert, um das OpenAI Gym-Environment `LunarLander-v2` zu erlernen[Bro+16]. Ziel war es, einen intelligenten Agenten zu trainieren, der erfolgreich auf dem Mond landet. Dazu wurden verschiedene Komponenten wie ein neuronales Netzwerk, ein Replay Buffer, die epsilon-greedy Strategie und ein Zielnetzwerk kombiniert. Das Training, die Evaluierung und die Visualisierung wurden in einem interaktiven Jupyter Notebook umgesetzt.

1.2 Modellarchitektur und Implementierung

Das zugrunde liegende neuronale Netzwerk besteht aus zwei versteckten Schichten mit jeweils 128 Neuronen und ReLU-Aktivierungen. Die finale Ausgabeschicht gibt Q-Werte für jede der vier möglichen Aktionen des Agenten aus. Der Replay Buffer speichert vergangene Erfahrungen und erlaubt es dem Agenten, zufällig ausgewählte Batch-Transitions zur Stabilisierung des Lernprozesses zu nutzen. Das Zielnetzwerk stabilisiert das Lernen zusätzlich, indem es nur periodisch aktualisiert wird.

Das Training wurde über 1000 Episoden hinweg durchgeführt, wobei sich der Agent durch Anwendung der epsilon-greedy Strategie von einer explorativen zu einer exploitierenden Politik entwickelte.

1.3 Hyperparameter und Trainingsdetails

Episoden: 1000

Discount-Faktor: $\gamma = 0{,}99$

Lernrate: 1e-3

Batch-Größe: 64

Replay Buffer: 10000

Target-Update-Intervall: 10

Epsilon (Start/Ende): 1,0 / 0,05

Epsilon-Decay: 0,995

1.4 Belohnungsstruktur

Die Belohnung im `LunarLander-v2`-Environment erfolgt automatisch durch die interne Logik von OpenAI Gym[Ope24]:

+100 bis +140: Für eine erfolgreiche Landung mit beiden Beinen.

-100**:** Bei Absturz oder Landung außerhalb der Zone.

$-0{,}3$**:** Pro Triebwerkszündung.

Weitere Belohnungen: Je nach Distanz zur Landezone, Rotation, Geschwindigkeit und Bodenkontakt.

Diese Struktur fördert sicheres, stabiles und möglichst sparsames Landen.

1.5 Trainingsverlauf und Visualisierung

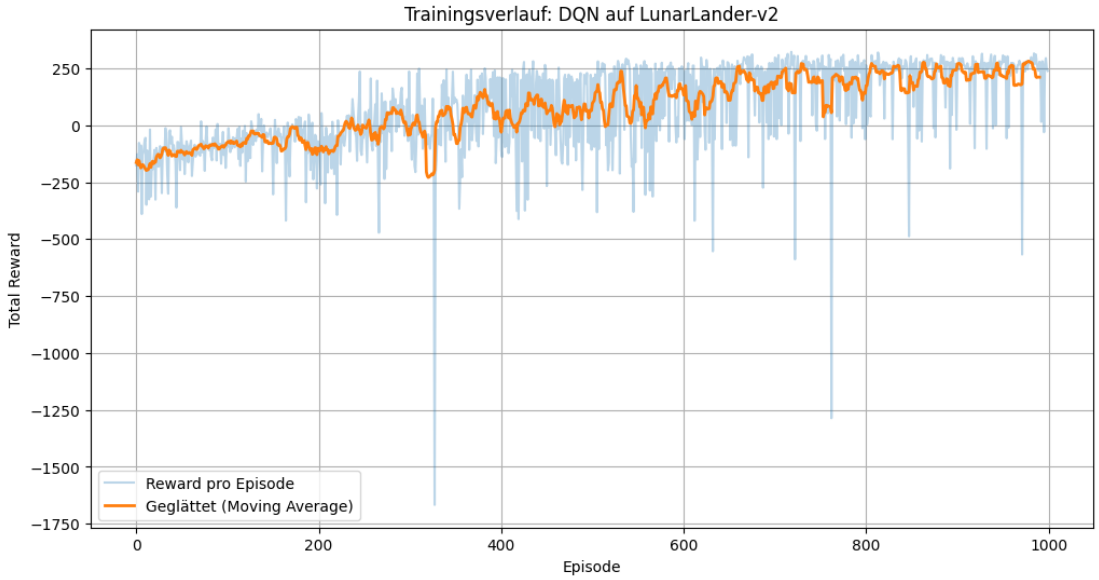

Fig. 1.1: Trainingsverlauf des DQN-Agenten in der Umgebung *LunarLander-v2*. Die blaue Linie zeigt den Reward pro Episode, die orange Linie den geglätteten Verlauf mittels gleitendem Mittelwert über 10 Episoden. Insgesamt lässt sich eine klare Lernkurve mit ansteigender Performance erkennen.

1.6 Evaluation

Nach dem Training wurde der Agent in 10 unabhängigen Testepisoden evaluiert. Dabei ergab sich ein durchschnittlicher Reward von ~220, was auf eine stabile und erfolgreiche Landefähigkeit des Agenten hinweist. In mehreren Testläufen wurden darüber hinaus Videos erstellt, um das Verhalten visuell nachvollziehen zu können.

1.7 Videoerzeugung und Beispielausgabe

Nach dem Training kann das fertige Modell mithilfe einer bereitgestellten Funktion fünfmal ausgeführt werden. Dabei wird die Umgebung jeweils visuell gerendert, und aus jedem Durchlauf wird eine GIF-Datei erzeugt. Der erzielte Gesamt-Reward jeder Episode wird automatisch im Dateinamen der entsprechenden Datei vermerkt.

Zwei Beispielvideos mit Rewards von 2 und 292 befinden sich im Ordner `videos/` auf Github[Kuh25].

1.8 Anforderungen und Hinweise zur Ausführung

Die gesamte Implementierung befindet sich in der Datei `dqn_lunarlander.ipynb`. Sie kann lokal oder in Google Colab ausgeführt werden. Die Installation der benötigten Pakete erfolgt über die bereitgestellte Datei `requirements.txt`. Wichtig ist, dass **NumPy in der Version 1.23.5** verwendet wird, da neuere Versionen zu Inkompatibilitäten mit `box2d-py` führen (z. B. wegen `np.bool`-Deprecation)[Iss24].

1.9 Fazit

Das Projekt demonstriert erfolgreich, wie ein DQN-Agent das LunarLander-v2-Environment erlernt. Die Lernkurve zeigt eine deutliche Steigerung der Leistung, und die Videoausgabe belegt die Fähigkeit des Agenten zur stabilen Landung. Durch das modulare und kommentierte Notebook ist die Implementierung leicht nachvollziehbar und kann für weitere RL-Experimente angepasst werden.

Literaturverzeichnis

[Bro+16] Greg Brockman et al. *OpenAI Gym*. https://arxiv.org/abs/1606.01540. 2016.

[Iss24] GitHub Issues. *box2d-py is incompatible with NumPy 1.24+*. 2024. URL: https://github.com/openai/gym/issues/2660.

[Kuh25] Tim Kuhn. *Deep-Q-Learning LunarLander (GitHub-Repository)*. Zugriff am 31.07.2025. 2025. URL: https://github.com/KuhnTim/deep-q-learning-lunarlander.

[Mni+15] Volodymyr Mnih et al. „Human-level control through deep reinforcement learning“. In: *Nature* 518.7540 (2015), S. 529–533.

[Ope24] OpenAI. *OpenAI Gym Docs*. Zugriff am 31.07.2025. 2024. URL: https://www.gymlibrary.dev/.