Duale Hochschule Baden-Württemberg Mannheim

Seminararbeit

Dueling DDQN Implementierung mit PER bei Atari Breakout

Studiengang Wirtschaftsinformatik

Studienrichtung Data Science

Verfasser:	Constantin Rech
Matrikelnummer:	8028907
Firma:	Boehringer Ingelheim Pharma GmbH & Co. KG
Kurs:	WWI-DSA-20
Studiengangsleiter:	Prof. Dr.-Ing. Dennis Pfisterer
Vorlesung:	Aktuelle Data Science-Entwicklungen II
Dozent:	Fr. Patzer

Inhaltsverzeichnis

Abbildungsverzeichnis

Abkürzungsverzeichnis

RL Reinforcement Learning

PER Prioritized Experience Replay

1 Wahl und Beschreibung der RL Umgebung

Die folgende Arbeit handelt von einer Implementierung eines Reinforcement Learning Ansatzes zum Lernen des Atari Spiels Breakout. Breakout ist ein Klassiker den jede Person die ein wenig mit Computer und Mobile Spielen zu tun hat kennt. Wenn der Name Breakout einer Person nichts sagt, dann hat diese es dennoch schon einmal gesehen. Auch ich habe es früher leidenschaftlich am iPad gespielt. Genau deshalb ist es nun umso faszinierender, dass mein jetziges Ich mittlerweile das Wissen hat, dieses Spiel gar nicht mehr selbst spielen zu müssen, sondern den Computer seinen Highscore schlagen lassen zu können. Da Breakout ein ursprüngliches Atari Spiel ist, wird für die Implementierung das Atari Enviroment das OpenAI Gym Interface genutzt. Dieses kann das Spiel simulieren um Reinforcement Learning (RL) möglich zu machen. Das Spiel ist simpel aufgebaut. Es gibt einen vom Spieler zu verschiebenden Balken, einen Ball und Blöcke. Der Balken ist ganz unten im Bildschirm und die Blöcke in einer zufälligen Anordnung oder einem Muster am oberen Spielfeldrand. Der Ball fliegt zwischen Balken und Blöcken hin und her. Das Spielfeld ist nach unten geöffnet, sodass der Ball ab unter dem Balken im Aus ist und somit das Spiel zu Ende ist. Über dem Balken sind sowohl Seitenwände sowie Decke für den Ball abprallend. Ziel des Spiel ist es den Balken unten immer so zu verschieben das der Ball abgefangen wird und wieder nach oben abprallt wird. Wenn der Ball einen Block trifft so zerbricht dieser. Das Endziel ist es alle Blöcke zu zerbrechen. Das Modell muss also lernen den Balken so zu verschieben, dass der Ball niemals am Balken vorbei ins Aus fliegt und somit das Spiel zu Ende ist. Der Balken muss somit immer vom Modell richtig verschoben werden, damit dieser wieder abprallen kann und Blöcke im oberen Bereich zerbrechen kann. Insgesamt hat man 5 Leben in einem Durchlauf, dass bedeutet der Ball kann 5 mal am Balken vorbeifliegen, bevor das Spiel zu Ende ist. Beispielhaft ist das Spielfeld in Abbildung 1 zu sehen.

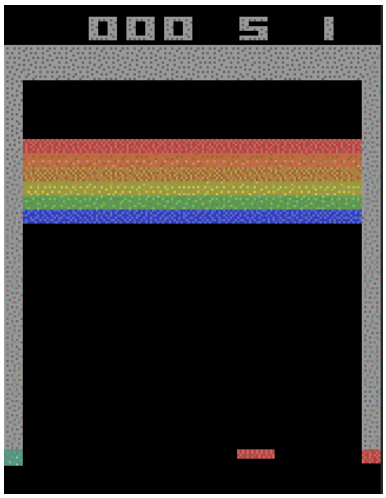

Abbildung 1: Spielfeld des Atari Breakoutspiels (OpenAI-Gym, 2023)

Bei der RL Umgebung handelt sich es um eine diskrete Umgebung. Es gibt insgesamt vier Aktionen die der Agent ausführen kann. Erstens kann dieser die Aktion Nichts (Noop) ausführen, zweitens das Spiel starten (Fire) und drittens und viertens, den Balken nach rechts oder links verschieben. Die Umgebung gibt standardmäßig ein RGB Bild des Spielfeldes zurück, dies kann nach Wahl jedoch auch in ein gray-scale Bild, wie in dieser Implementierung umgewandelt werden. Rewards erhält der Agent durch das zerstören der Blöcke. Wenn der Agent also alle Blöcke zerstört hat, hat er den höchsten Reward und gleichzeitig das Spiel gewonnen (vg. OpenAI-Gym, 2023)

2 Entwicklung des RL-Agenten

Bei der Suche nach Implementierungsmöglichkeiten für das Atari Backout Spiel wurde folgende Archtiektur bzw. auch der dazugehöhrige Code gefunden. Fokus dieser Arbeit bestand hierbei mehr darin den Code zu verstehen und in das Gelernte einzuordnen, als diesen selbst zu coden. Die Quelle ist sowohl im Jupyter-Notebook (letzte Zelle) als auch hier (Theiler, 2020) verlinkt. Die Implementierung besteht aus einem Dueling DDQN (Wang et al., 2016) mit Prioritized Experience Replay (PER) (Schaul et al., 2016). Der Aufbau des Agent besteht hierbei aus dem DQN welches ein Neuronales Netz ist, bestehend aus Convolutional Layern sowie am Ende Fully Connected Layern. Diese wird zum lernen von Gewichte eingesetzt. Das Modell bekommt als Input die Pixel beziehungsweise das grey-scale Bild des Spielfeldes und bestimmt dann den mögliche Reward der Aktionen. Hier fällt dann die Entscheidung auf die Aktion mit dem höchsten Reward. Während den Durchläufen gibt die Umgebung auch Informationen über das nächste Bild, den erreichten Reward und ob das Spiel beendet ist weiter. Diese Informationen können dann genutzt werden um besser zukünftige Entscheidungen treffen zu können. Der Speicher von diesen Informationen heißt Replay Buffer (vgl. Theiler, 2022). Grafisch kann die Architektur der Umgebung dann wie in Abbildung 2 dargestellt werden. Hierbei ist jedoch zu beachten das in dieser Abbildung nur ein DDQN dargestellt ist und kein Dueling DDQN.

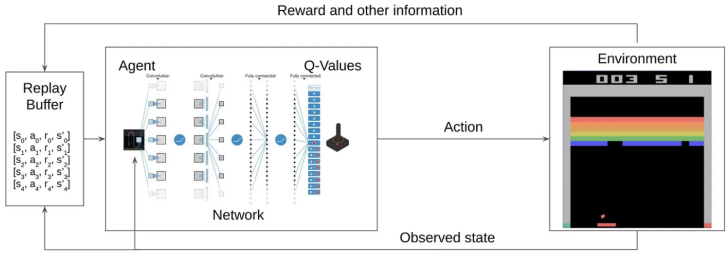

Abbildung 2: Architektur der Implementierung (Theiler, 2022)

Für die Vorhersage der richtigen Aktionen wird die Bellman-Equation genutzt. Diese muss benutzt werden, da man die richtige Aktion im Vorfeld nicht vordefiniert bestimmen kann. Durch den Dueling Ansatz gibt es zwei Netze. Eins zum vorhersagen des Q-Wertes und somit der Aktion und eins zum produzieren der Ziel Q-Werte durch die eben genannte Bellman-Equation. Alle 10000 Frames werden die beiden Netze auf die gleichen Parameter gesetzt, das zweite Netz wird hierbei mit den Parametern des Ersten gleich gesetzt. Dadurch erhält man mehr Stabilität im Training. Ein weiterer wichtiger Punkt ist das das eigentliche Atari Breakout ein 210x160 Pixel großes Spielfeld hat. Umgerechnet wären das jedoch 100800 Werte im RGB Fall (3 Dimensionen), weshalb in dieser Implementierung das Spielfeld auf 84x84 Pixel grey-scale (1 Dimension) geresized wird. Dadurch wird es nicht so rechenintensiv. Durch das Dueling der Neuronalen Netze wird intuitiv vom einem Netz gelernt, welche Zustände wertvoll sind, ohne dass wir die Auswirkungen jeder Aktion in diesem Zustand lernen müssen. Dies beschleunigt das Training erheblich. Durch die Nutzung des „BreakoutDeterministic-v4“von Gym werden 4 vergangene Frames dem Q-Network übergeben, um aus diesen Informationen zu schließen. Im Replay Buffer werden dann die verschiedenen Informationsrückflüsse bewertet. Hierfür wird eine Funktion genutzt, die die Wichtigkeit der Informationen angibt. Dies funktioniert mit Hilfe des TD errors, wie es auch schon in der Vorlesung vorgestellt wurde. Alle Ergebnisse des Trainings werden dann mit Tensorflow und dem Tensorflowboard getrackt. Im folgenden Kapitel wird auf diese getrackten Ergebnisse eingegangen.

3 Bewertung und Visualisierung

Abbildung 3 zeigt den durchschnittlichen Reward über die Dauer des Trainings an. Der Graph ist mit einem Smoothing Faktor von 0.85 versehen, um einen besseren Verlauf darstellen zu können.

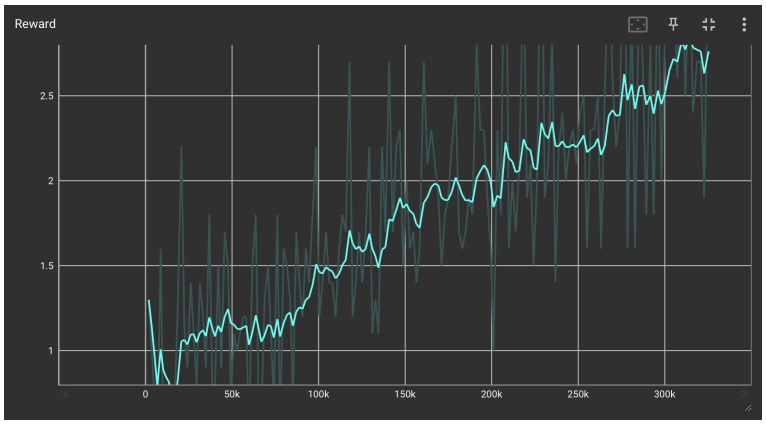

Abbildung 3: Durchschnittlicher Reward über Trainingszeit

Es ist gut zu erkennen, dass der Reward im Laufe des Trainings steigt und somit das Modell es lernt das Spiel Breakout zu spielen. Das Modell ist 533 Minuten gelaufen und hat 1660 Spiele bewältigt. Der aktuelle durchschnittliche Reward ist jedoch nicht sehr gut, da das Modell durch Hardwaregegebenheiten stark limitiert war. Es wäre empfehlenswert hier eine GPU zu nutzen, da das Training sonst sehr lange dauert. So korrelieren Zeit und Reward positiv, denn das Modell erhält mehr Reward wenn es länger im Spiel bleibt und mehr Blöcke zerstört. Es ist hier also auf jeden Fall von großem Vorteil eine GPU zu nutzen und somit weniger Rechenzeit pro Spiel zu brauchen. Wenn dies gegeben ist, können auch andere Modelle verglichen werden und geschaut werden welche effizienter sind. Hier könnte verglichen werden, ob dieses sehr optimierte Dueling DDQN mit PER wirklich erheblich besser performt als nur ein Dueling DDQN oder sogar lediglich ein DDQN.

Literatur

OpenAI-Gym. (2023). Atari Breakout. https://www.gymlibrary.dev/environments/atari/breakout/

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2016). Prioritized experience replay. https://arxiv.org/abs/1511.05952

Theiler, S. (2020). Building a powerful DQN in tensorflow 2.0. https://github.com/sebtheiler/tutorials/tree/main/dqn

Theiler, S. (2022). Building a powerful DQN in tensorflow 2.0 (explanation amp; tutorial). https://medium.com/analytics-vidhya/building-a-powerful-dqn-in-tensorflow-2-0-explanation-tutorial-d48ea8f3177a

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. https://arxiv.org/abs/1511.06581