Reinforcement Learning

für Atari Spiele

Timo Grautstück

Fachhochschule-Dortmund
FB: Informationstechnik
timo.grautstueck@fh-dortmund.de

30. August 2021

Worüber wollen wir sprechen ?

1. Einführung
   - Atari
2. Problem
   - MDP
   - Q-Funktionen
3. Model
   - Q-Netzwerk
   - Architektur
   - DQL-Algorithmus
   - Ergebnisse
4. Zusammenfassung

Atari ?

Atari, Inc.

- US-amerikanisches Unternehmen
- 1972 von Nolan Bushnell und Ted Dabney gegründet
- Herstellung von Arcade-Automaten und Computern

Durchbruch/Bekannt

- 1972: Pong *(Automat/ stationär)*
- 1977: Atari 2600 *(Atari VCS)*

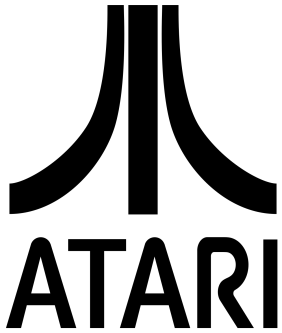


Abbildung: Atari-Logo

Fun-Facts

- Atari bez. eine Stellung in Go
- Steve Jobs/Wozniak arbeiteten vor der Gründung von Apple bei Atari

Verschiedenste Spiele

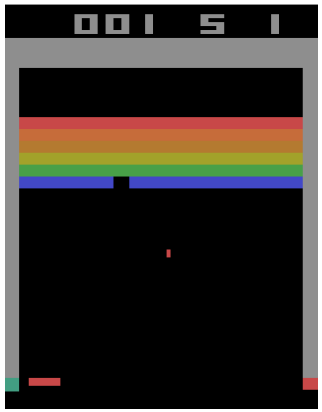


Abbildung: Breakout

Abbildung: Atari-2600

$210 \times 160 \times 3$ pixel

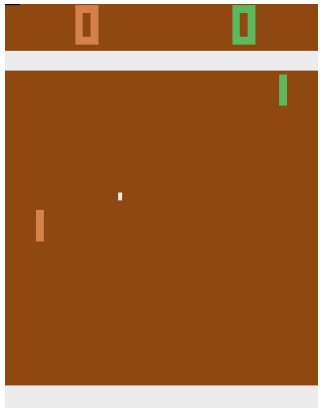


Abbildung: Pong

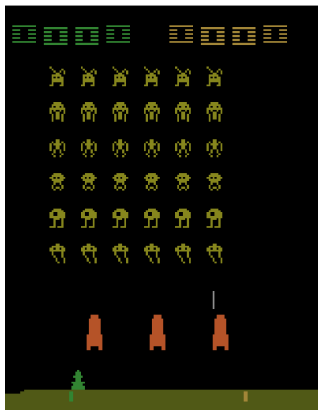


Abbildung: SpaceInvaders

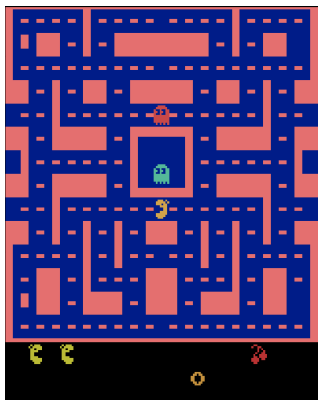


Abbildung: Ms Pacman

Software/Benchmark-Sammlungen

```
from ale_py import ALEInterface
ale = ALEInterface()
ale.loadROM('Breakout')
```

```
import gym
env = gym.make('Breakout_v4')
env.reset()
```

Arcade Learning Environment (2013)

Ein Herausforderndes Problem, eine Plattform und Testumgebung zur Bewertung und Vergleich von domänenunabhängigen KI.

- Über 55 verschiedene Atari 2600 Spiele
- Aufgebaut auf einem Open-Source Atari 2600 Emulator *(Stella)*

OpenAI Gym (2016)

Versucht die besten Elemente vorheriger Benchmark-Sammlungen *(ALE, RLLAB, . . .)* zu einem Software Paket zu vereinigen, das so bequem und zugänglich wie möglich ist.

- Verschiedene Environments *(MuJoCo, Robotics, . . .)*

Houston, we have a problem

It's a Markov decision process

1. Agent: *maximiere zukunfts Belohnungen*
2. Umgebung: $\mathcal{E}$
3. Zustand: $\mathcal{S} = \{s_1, \ldots, s_N\}$
4. Aktion: $\mathcal{A} = \{a_1, \ldots, a_N\}$
5. Belohnung: $\mathcal{R} = \{r_1, \ldots, r_N\}$

} zu jedem Zeitpunkt $t - T$

Sounds pretty simple

- Umgebung $\rightarrow$ Atari-Emulator
- Zustand $\rightarrow$ Bild aus dem Emulator $x_t \in \mathbb{R}^d$ Vektor aus Pixeln
- Aktion $\rightarrow$ legale Spielaktionen (max. 18)
- Belohnung $\rightarrow$ Änderung des Spielstands
- Agent $\rightarrow$ max. zukunfts Belohnungen durch Interaktion mit dem Emulator

Houston, we have a problem

It's a Markov decision process

1. Agent: *maximiere zukunfts Belohnungen*
2. Umgebung: $\mathcal{E}$
3. Zustand: $\mathcal{S} = \{s_1, \ldots, s_N\}$
4. Aktion: $\mathcal{A} = \{a_1, \ldots, a_N\}$
5. Belohnung: $\mathcal{R} = \{r_1, \ldots, r_N\}$

} zu jedem Zeitpunkt $t - T$

Sounds pretty simple

- Umgebung $\rightarrow$ Atari-Emulator
- Zustand $\rightarrow$ Bild aus dem Emulator $x_t \in \mathbb{R}^d$ Vektor aus Pixeln
- Aktion $\rightarrow$ legale Spielaktionen (max. 18)
- Belohnung $\rightarrow$ Änderung des Spielstands
- Agent $\rightarrow$ max. zukunfts Belohnungen durch Interaktion mit dem Emulator

Spezifischer

Zustand

Unmöglich das Spiel zu verstehen wenn wir ein Bild x_t als Zustand nutzen. Daher nutzen wir eine Sequenz von Bildern und Aktionen $s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t$. $\rightarrow$ Zustandsrepräsentation an Zeitpunkt t.

Belohnungen

Wir brauchen ein Maß für die Belohnungen, z.B die max. zu holende Belohnungen in einer Episode *(Punkt/Sterben)*.

$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \rightarrow$ return. Mit Ermäßigungsfaktor $0 \leq \gamma \leq 1$, hier $\gamma = 0.99$.

Spezifischer

Zustand

Unmöglich das Spiel zu verstehen wenn wir ein Bild x_t als Zustand nutzen. Daher nutzen wir eine Sequenz von Bildern und Aktionen $s_t = x_1, a_1, x_2, \ldots, a_{t-1}, x_t$. $\rightarrow$ Zustandsrepräsentation an Zeitpunkt t.

Belohnungen

Wir brauchen ein Maß für die Belohnungen, z.B die max. zu holende Belohnungen in einer Episode *(Punkt/Sterben)*.
$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'} \rightarrow$ return. Mit Ermäßigungsfaktor $0 \leq \gamma \leq 1$, hier $\gamma = 0.99$.

Q-Funktion

Aktions-Wert Funktion

$$Q(s,a) = \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

Was ist der erwartete Return R_t, wenn wir in einem Zustand s sind, Aktion a ausführen und der Policy π folgen. $\forall s \in \mathcal{S} \wedge \forall a \in \mathcal{A}(s)$,

Optimale Aktions-Wert Funktion

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

Was ist der erwartete Return, wenn wir in s, a ausführen und der optimalen policy π^* folgen.

Abbildung: Backup Diagramm Q^*

Q-Funktion

Aktions-Wert Funktion

$$Q(s,a) = \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

Was ist der erwartete Return R_t, wenn wir in einem Zustand s sind, Aktion a ausführen und der Policy π folgen. $\forall s \in \mathcal{S} \wedge \forall a \in \mathcal{A}(s)$,

Optimale Aktions-Wert Funktion

$$Q^*(s,a) = \max_\pi \mathbb{E}\left[R_t | s_t = s, a_t = a, \pi\right]$$

Was ist der erwartete Return, wenn wir in s, a ausführen und der optimalen policy π^* folgen.

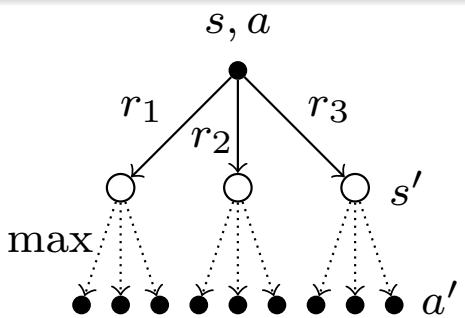


Abbildung: Backup Diagramm Q^*

Optimale Policy

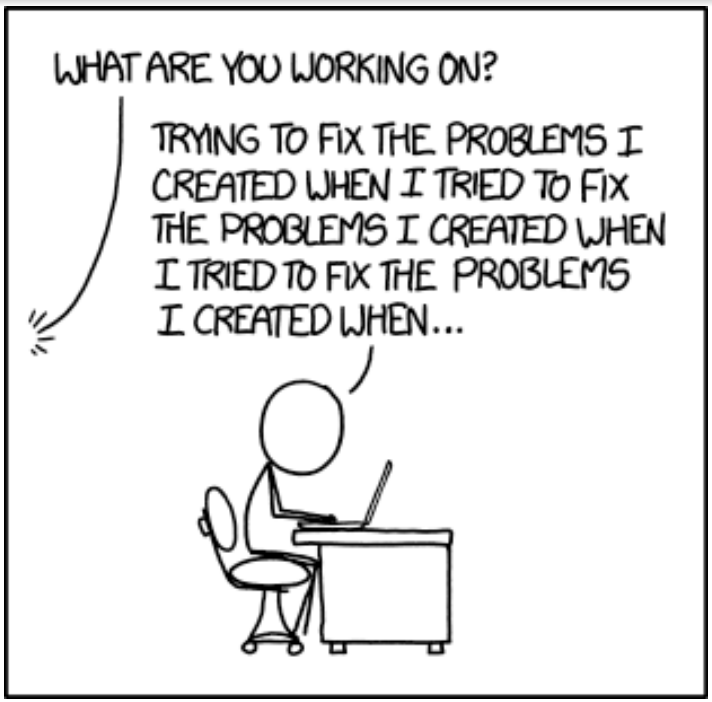
Bellmann Gleichung

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a \right]$$

Optimale Policy

Bellmann Gleichung

$$Q^*(s,a) = \mathbb{E}_{s' \sim \mathcal{E}} \left[r + \gamma \max_{a'} Q^*(s', a') \mid s, a \right]$$



Quelle: Randall Munroe, https://xkcd.com/1739/

Wie finden wir Q^* und somit eine optimale Policy π^* für unseren Agent ?

Q-Netzwerk

Wir erstellen ein Künstliches Neuronales Netz und versuchen somit Q^* zu approximieren.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

Das machen wir mit Hilfe einer Verlustfunktion.

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[(y_i - Q(s, a, \theta_i))^2\right]$$

mit dem Ziel $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a\right]$ für jede Interation, versuchen wir diese Verlustfunktion durch einen Gradientenverfahren zu minimieren und so Q^* zu approximieren.

Um welche Verlustfunktion handelt es sich in dieser Formel ?

Q-Netzwerk

Wir erstellen ein Künstliches Neuronales Netz und versuchen somit Q^* zu approximieren.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

Das machen wir mit Hilfe einer Verlustfunktion.

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[(y_i - Q(s, a, \theta_i))^2\right]$$

mit dem Ziel $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a\right]$ für jede Interation, versuchen wir diese Verlustfunktion durch einen Gradientenverfahren zu minimieren und so Q^* zu approximieren.

Um welche Verlustfunktion handelt es sich in dieser Formel ?

Q-Netzwerk

Wir erstellen ein Künstliches Neuronales Netz und versuchen somit Q^* zu approximieren.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

Das machen wir mit Hilfe einer Verlustfunktion.

$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[(y_i - Q(s, a, \theta_i))^2\right]$$

mit dem Ziel $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a\right]$ für jede Interation, versuchen wir diese Verlustfunktion durch einen Gradientenverfahren zu minimieren und so Q^* zu approximieren.

Um welche Verlustfunktion handelt es sich in dieser Formel ?

Q-Netzwerk

Wir erstellen ein Künstliches Neuronales Netz und versuchen somit Q^* zu approximieren.

$$Q(s, a; \theta) \approx Q^*(s, a)$$

Das machen wir mit Hilfe einer Verlustfunktion.

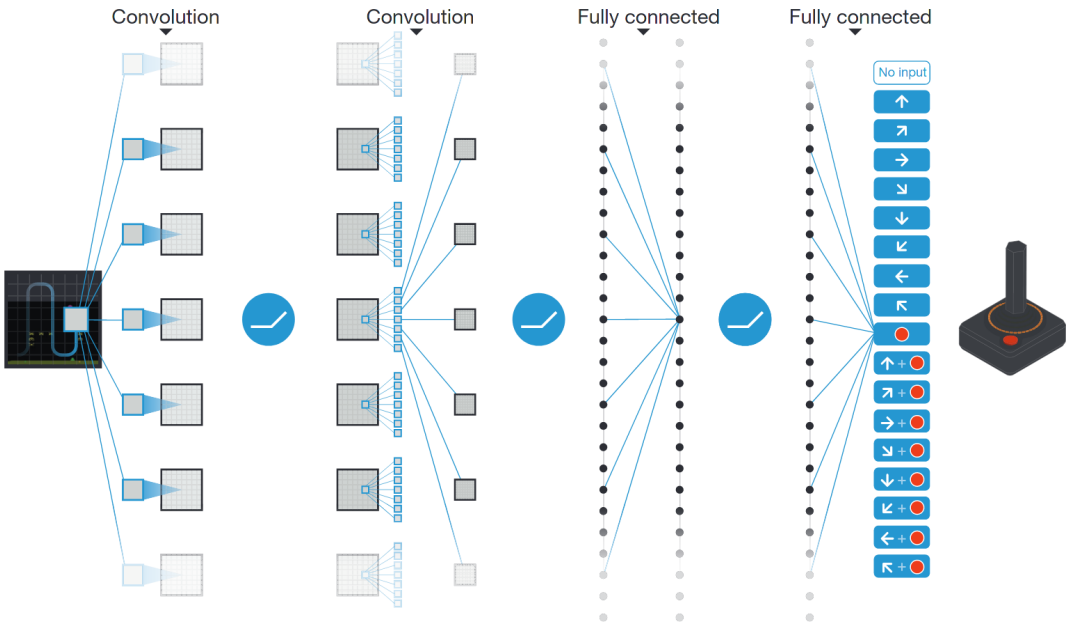
$$\mathcal{L}_i(\theta_i) = \mathbb{E}_{s,a\sim p(\cdot)}\left[(y_i - Q(s, a, \theta_i))^2\right]$$

mit dem Ziel $y_i = \mathbb{E}_{s'\sim\mathcal{E}}\left[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) \mid s, a\right]$ für jede Interation, versuchen wir diese Verlustfunktion durch einen Gradientenverfahren zu minimieren und so Q^* zu approximieren.

Um welche Verlustfunktion handelt es sich in dieser Formel ?

L2-Verlustfunktion

Q-Netzwerk/Architektur



Quelle: V. Mnih, `https://www.nature.com/articles/nature14236`

Schicht	Details	Aktivierung
Eingabe	$(84 \times 84 \times 4)$ Eingabe durch $\phi(s)$	
Faltung	16 (8×8) Filter, Schritt 4	ReLU
Faltung	32 (4×4) Filter, Schritt 2	ReLU
Dense	256 verborgene Neuronen	ReLU
Dense (Ausgabe)	Ein Neuron pro Aktion	Linear

Vorverarbeitung

Was ist nun $\phi(s)$

Ein Vorverarbeitungsschritt um Eingabedimensionen zu reduzieren.
Durch RGB zu Grauskalierung und komprimieren des Bildes.

- $210 \times 160 \times 3 \rightarrow 110 \times 84 \times 1$

Can we do better ?

Interessant ist die Spielfläche, also schneiden wir sie aus dem Bild.

- $84 \times 84 \times 1$, jedoch stapeln wir vier davon hintereinander $\times 4$.

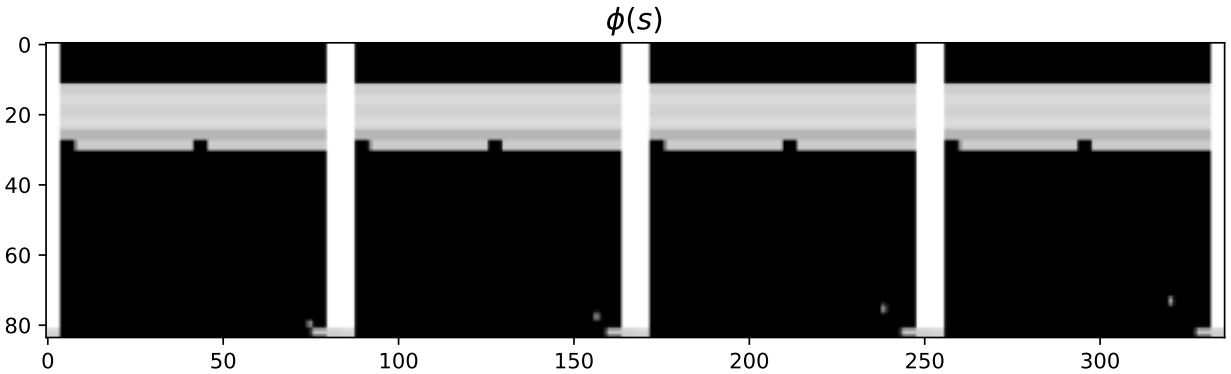


Abbildung: $\phi(s)$: Input Q-Netzwerk

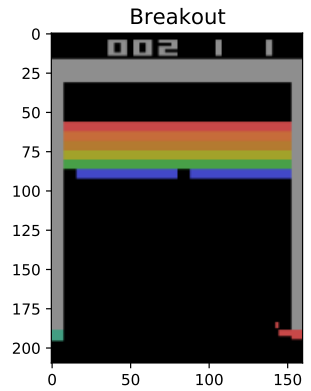
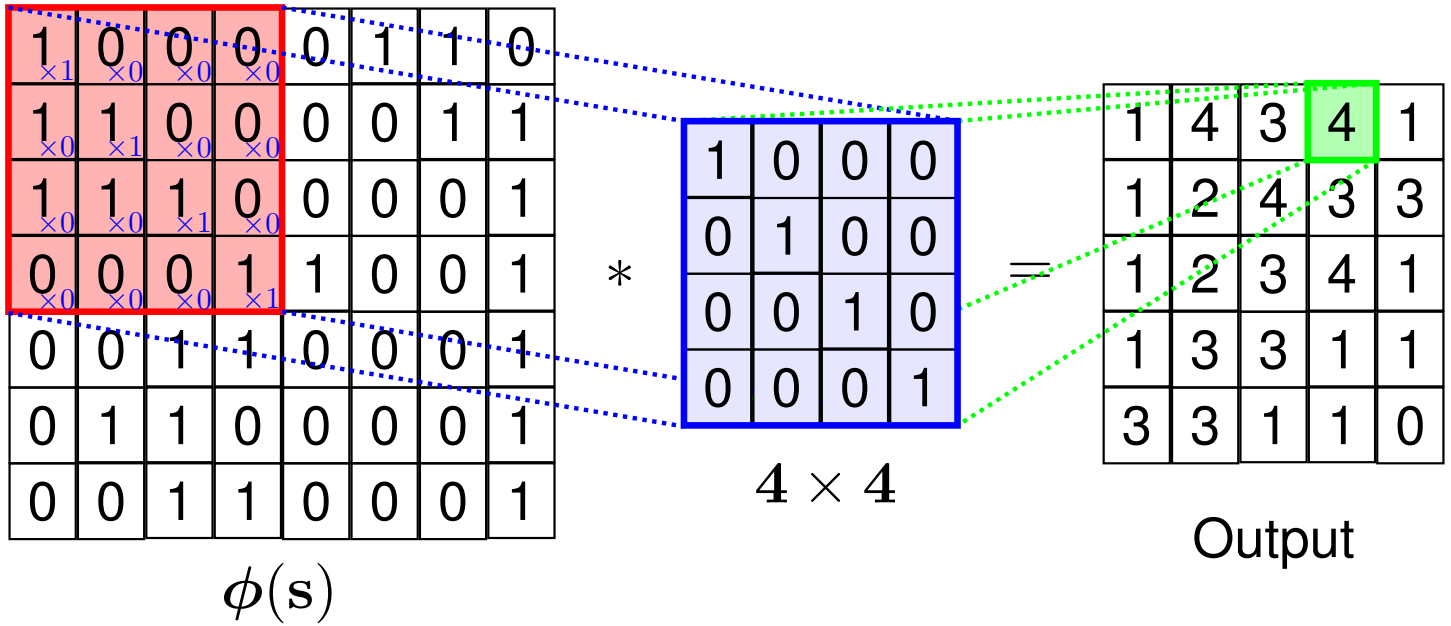


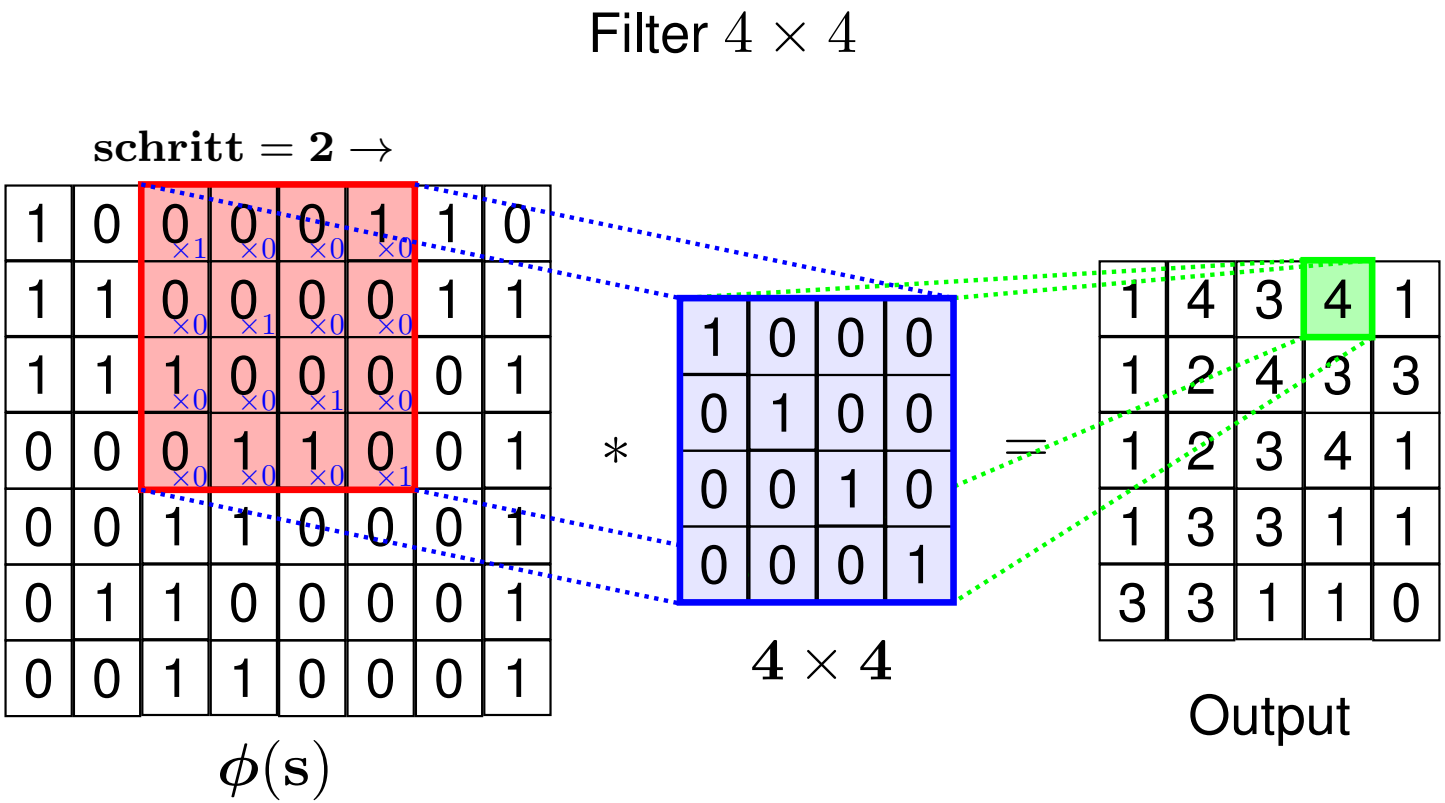
Abbildung: Breakout

Faltung/CNN

Filter 4×4

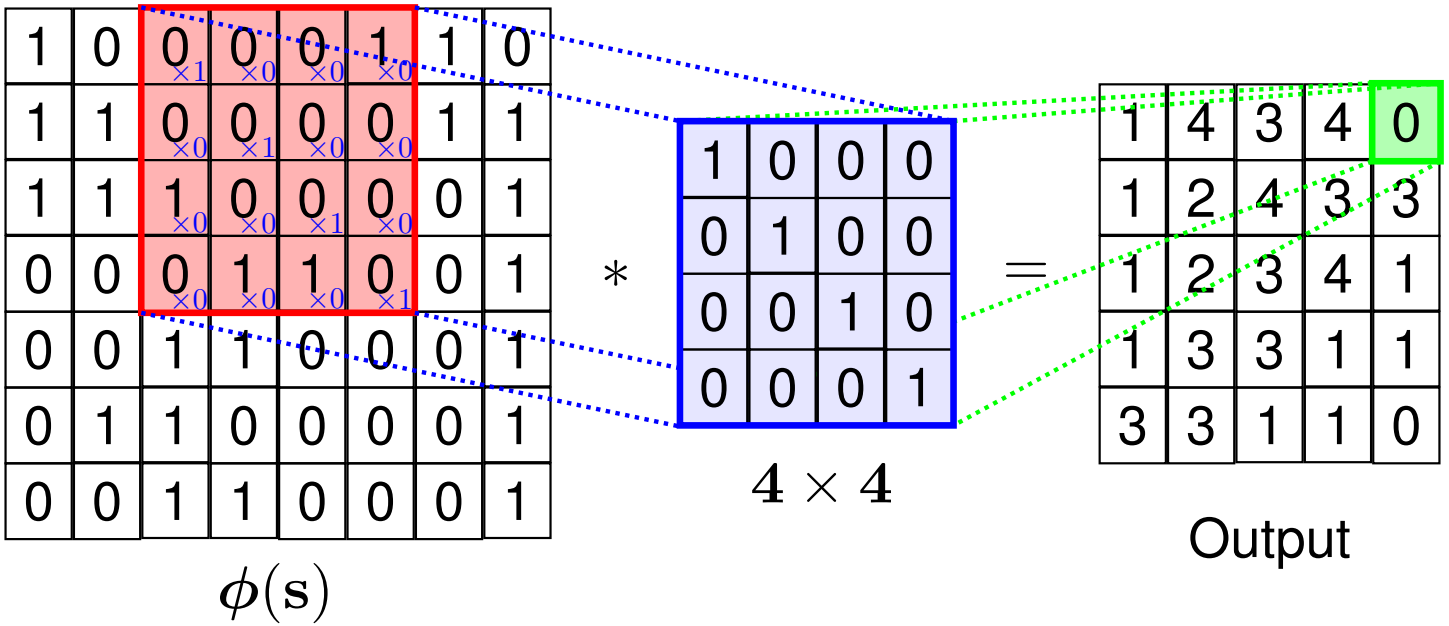


Faltung/CNN


Filter 4×4
$\mathbf{schritt} = \mathbf{2} \rightarrow$
$\phi(\mathbf{s})$
$*$
4×4
$=$
Output

Faltung/CNN

Filter 4×4

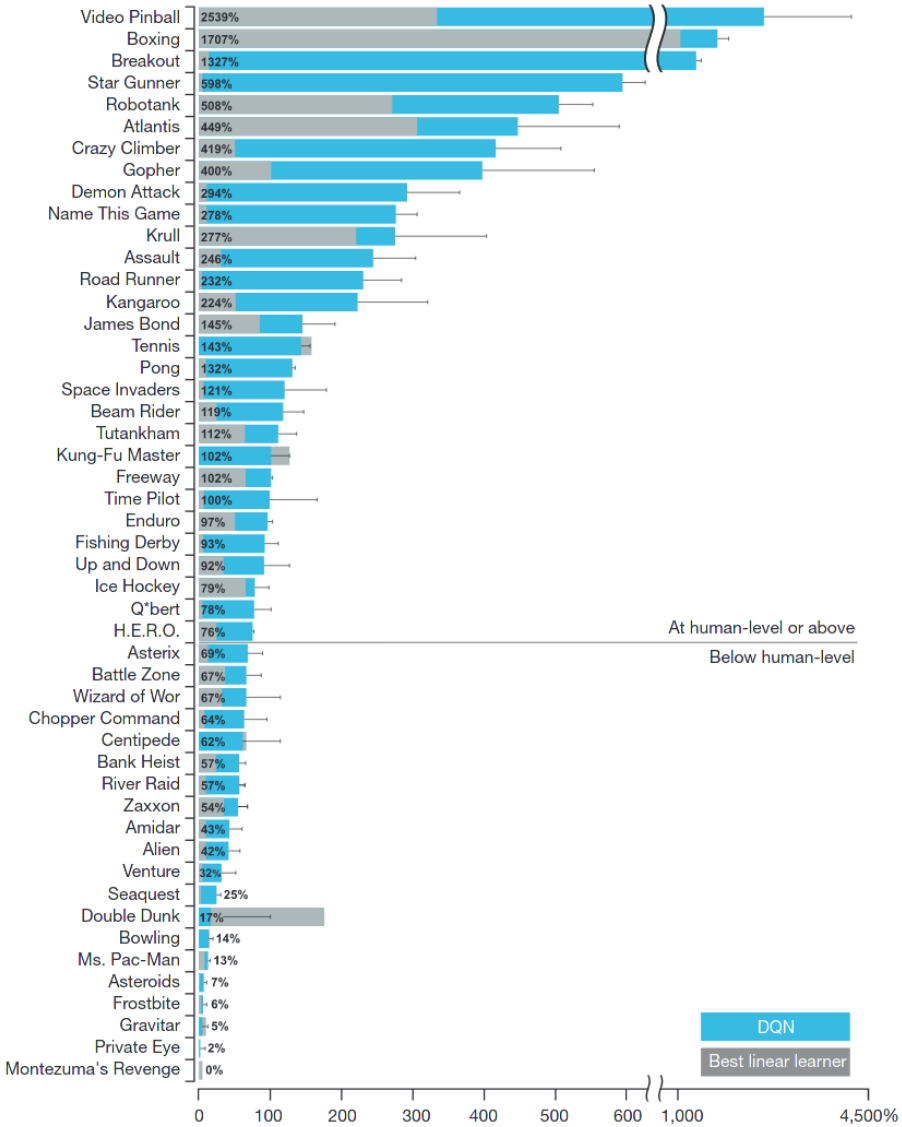


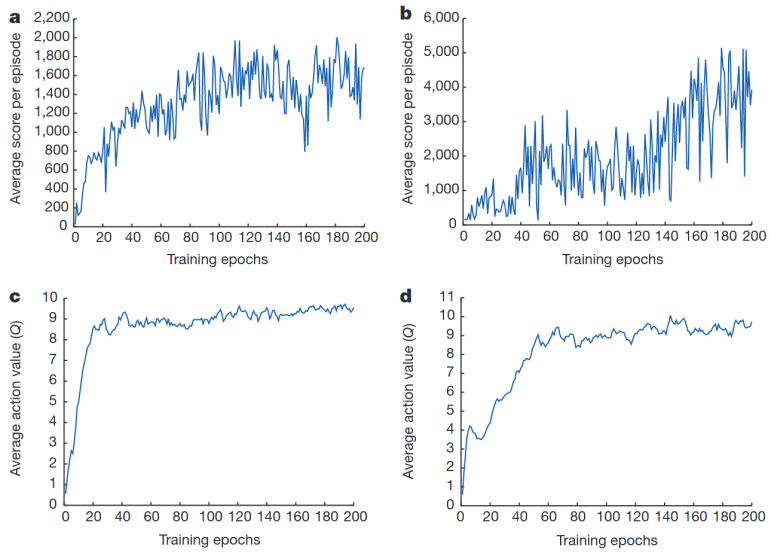
Algoritmus

Deep Q-Learning mit Erlebnis-Wiedergabe

Initialisiere Wiedergabespeicher $\mathcal{D}$ mit Kapazität N
Initialisiere aktions-werte Funktion Q mit zufälligen Gewichten
for Episode = 1,M **do**
    Initialisiere Sequenz $s_1 = \{x_1\}$ und vorverarbeite $\phi_1(s) = \phi(s_1)$
    for $t = 1, T$ **do**
        Mit Wahrscheinlichkeit ϵ wähle eine zufällige Aktion a_t
        andernfalls wähle $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
        Führe Aktion a_t im Emulator aus, erhalte Belohnung r_t und Bild x_{t+1}
        Setze $s_{t+1} = s_t, a_t, x_{t+1}$ und vorverarbeite $\phi_{t+1} = \phi(s_{t+1})$
        Speicher Übergang $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
        Wähle einen zufälligen minibatch von Übergängen $(\phi_j, a_j, r_j, \phi_{j+1})$ aus $\mathcal{D}$
        Setze $y_i = \begin{cases} r_j & \text{für beendendes } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{für nicht beendendes } \phi_{j+1} \end{cases}$
        Mache einen Gradientenabstieg schritt auf $(y_j - Q(\phi_j, a_j; \theta))^2$
    end for
end for

Ergebnisse aus 2015





Quelle: V. Mnih, `https://www.nature.com/articles/nature14236`

Agent spielt Breakout:
`https://www.youtube.com/watch?v=TmPfTpjtdgg`

Worüber haben wir gesprochen ?

1. MDP's am Beispiel Atari
2. Q-Funktionen
3. Architektur/NN
4. DQL-Algorithmus

Quellen


R.S. Sutton, A.G. Barto
Reinforcement Learning: An Introduction - Chap. 3
MIT Press, Cambridge, MA, 2018
http://incompleteideas.net/book/the-book-2nd.html


G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba
OpenAI Gym
arXiv:1606.01540, 2016
https://arxiv.org/abs/1606.01540


M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling
The Arcade Learning Environment: An Evaluation Platform for General Agents
arXiv:1207.4708, 2013
https://arxiv.org/abs/1207.4708


V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves,I. Antonoglou, D. Wierstra, M. Riedmiller
Playing Atari with Deep Reinforcement Learning
arXiv:1312.5602, 2013
https://arxiv.org/abs/1312.5602


V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves,I. Antonoglou, D. Wierstra, M. Riedmiller, ...
Human-level control through deep reinforcement learning
Nature 518, 529-533 (2015)
https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf

Quellen

D. Seita
Frame Skipping and Pre-Processing for Deep Q-Networks on Atari 2600 Games
https://danieltakeshi.github.io/2016/11/25/frame-skipping-and-preprocessing-for-deep-q-networks-on-atari-2600-games/

Abbildungen:
https://commons.wikimedia.org/wiki/File:Atari_logo_alt.svg
https://de.wikipedia.org/wiki/Datei:Atari-2600-Wood-4Sw-Set.png