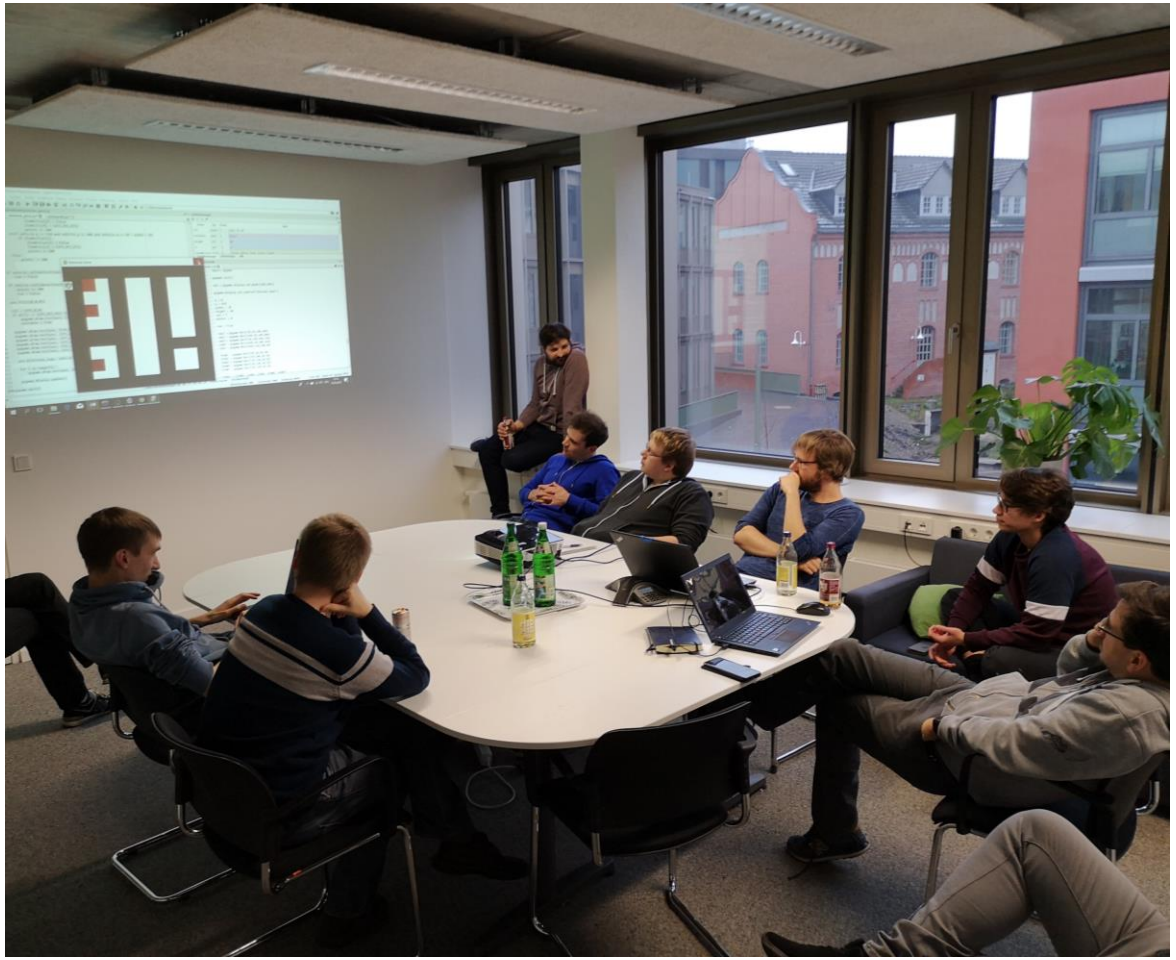
Meetup Vortrag 19.02.19 – Martin & Florian

Wie man einer KI das Spielen beibringt

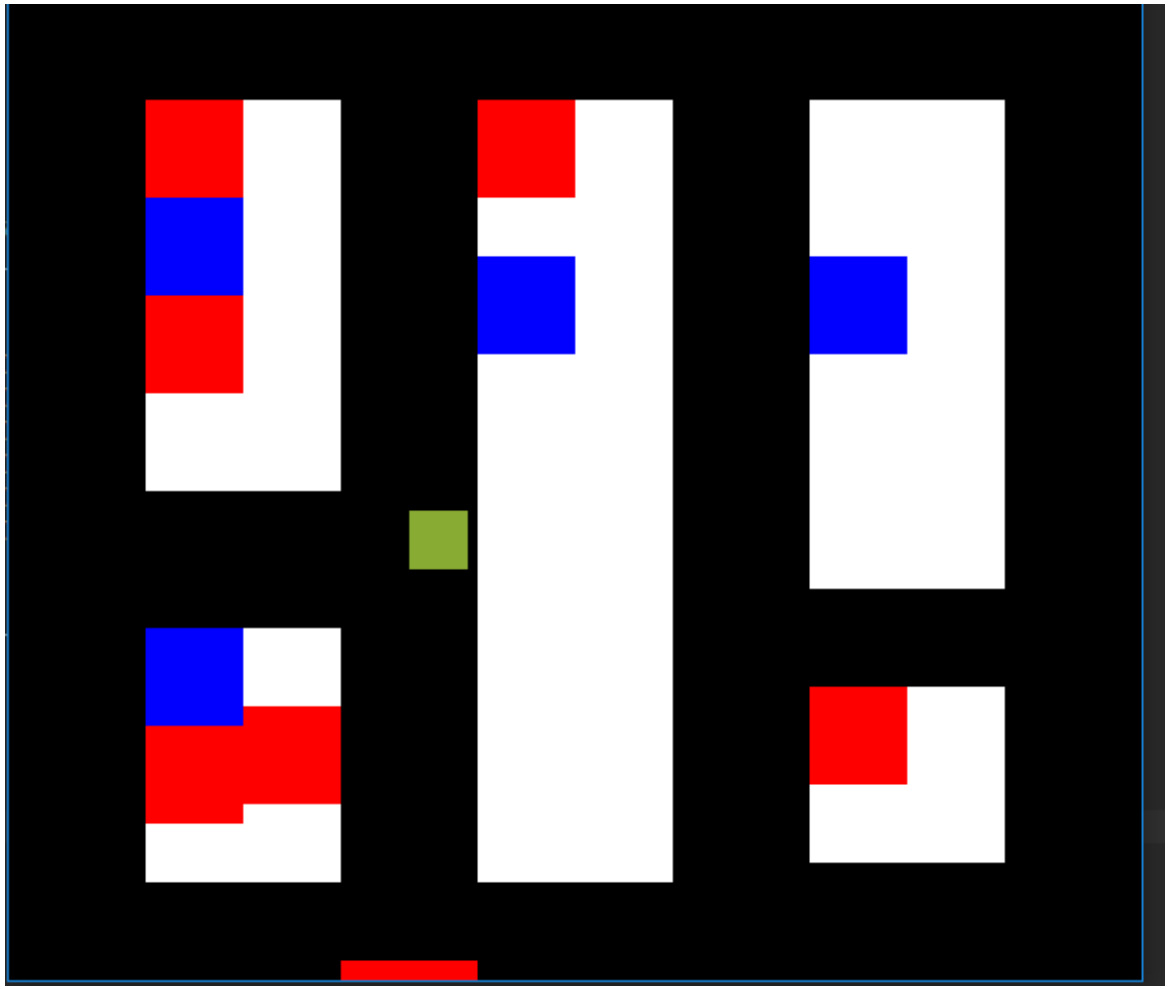
Hintergrund: Hackathon

1. Reinforcement Learning verstehen
2. Entscheidung für ein Spiel
3. Spiel entwickeln
4. Algorithmus entwerfen
5. Modell trainieren



Das Spiel

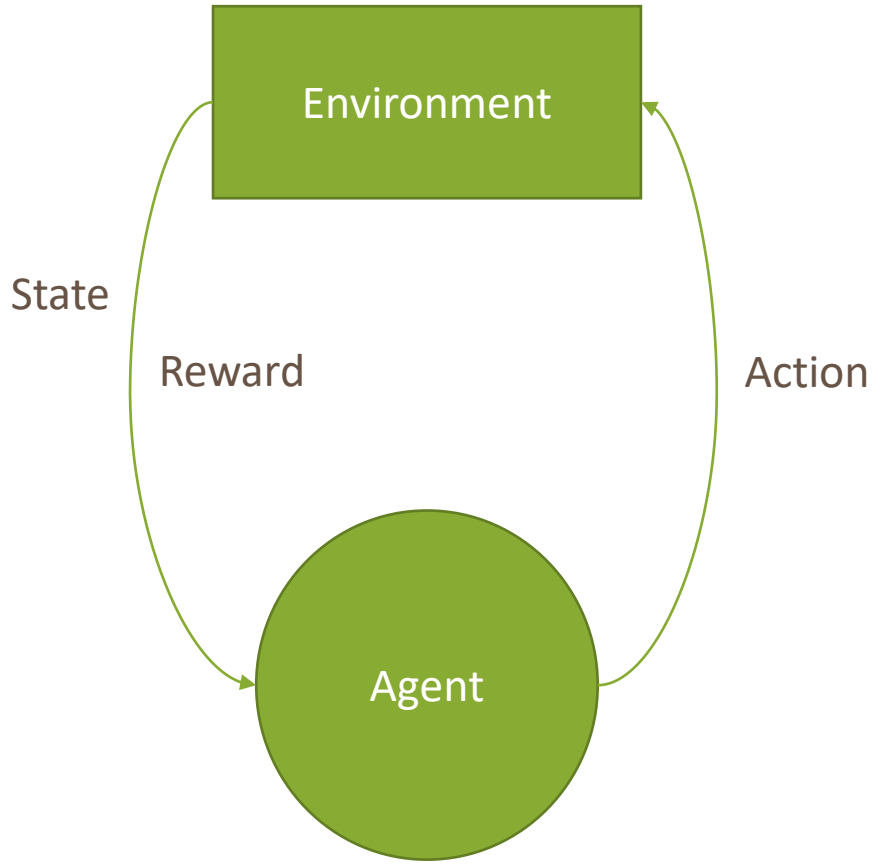
Wir sind ein Lagerarbeiter mit der Aufgabe bestimmte Pakete so schnell wie möglich einzusammeln und zum Lagerausgang zu bringen.



Reinforcement Learning

Reinforcement Learning

- Environment liefert State (Zustand) und Reward (Belohnung) an den Agent
- Agent entscheidet sich für eine Action und liefert diese an das Environment
- Interaktion passiert in diskreten Zeitschritten
- Reward r_{t+1} wírd vom Agent in Verbindung gesetzt mit dem Übergang (s_t, a_t, s_{t+1}) wobei s_t den State und a_t die Action zum Zeitpunkt t darstellt
- Herausforderung: Langfristige vs kurzfristige Auswirkungen



Q-Learning

- $Q^{new}(s_t, a_t) = (1 - \alpha) \times Q(s_t, a_t) + \alpha \times \underbrace{(r_t + \gamma \times \max_a Q(s_{t+1}, a))}_{\text{„Gelernter“ Wert}}$

- γ – Discount, je größer desto stärker werden zukünftige Belohnungen gewichtet
- α – Lernrate, „explore“ ($\alpha = 1$) vs „exploit“ ($\alpha = 0$)
- r_t – Belohnung
- s_t – Zustand
- a_t - Aktion

Anwendungen – DeepMind

- In 2010 gegründetes Unternehmen
- In 2014 von Google übernommen
- Entwicklung von AI im Bereich Medizin
- Forschung im Bereich Reinforcement Learning bei Spielen (Arcade-Games, Strategie Spiele)

Anwendung – DeepMinds AlphaZero

- Erste Entwicklung für atari Arcade Spiele

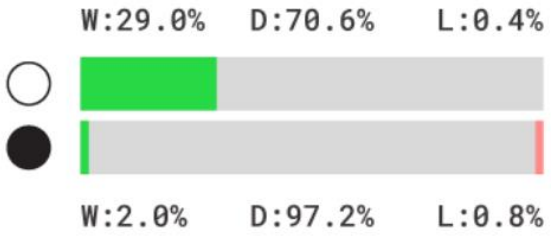
- AlphaGo: Programm für das Strategiespiel Go – Sieg gegen den stärksten Go-Spieler in 2017

- Weiterentwicklung zu AlphaZero: Erfolge in Go, Shogi und Schach
  - Lernt durch Spielen gegen sich selbst
  - Generalisierter Algorithmus, der vorherige Versionen (AlphaGo) und andere KIs (Stockfish, Elmo,…) besiegt (unter teilweise kritischenBedingungen)
  - AlphaZero rechnet ca. 80.000 Positionen pro Sekunde (Schach), Stockfish rechnet 70 Millionen
- Während herkommliche Engines auf Regeln und Heuristiken beruhen, basiert DeepMind auf einem neuronalen Netz

Ergebnisse gegen State-of-the-Art-Engines

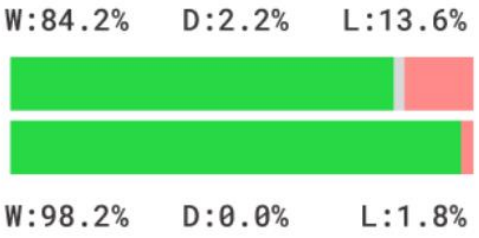
Chess

AlphaZero vs. Stockfish

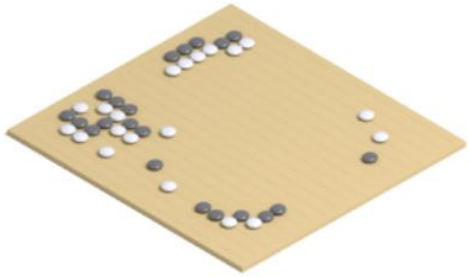


Shogi

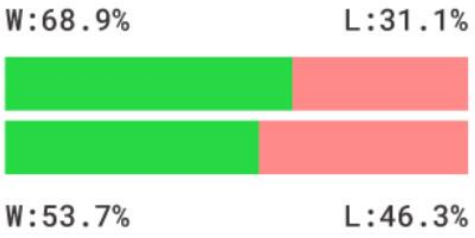
AlphaZero vs. Elmo



Go



AlphaZero vs. AG0



Anwendung – AlphaStar

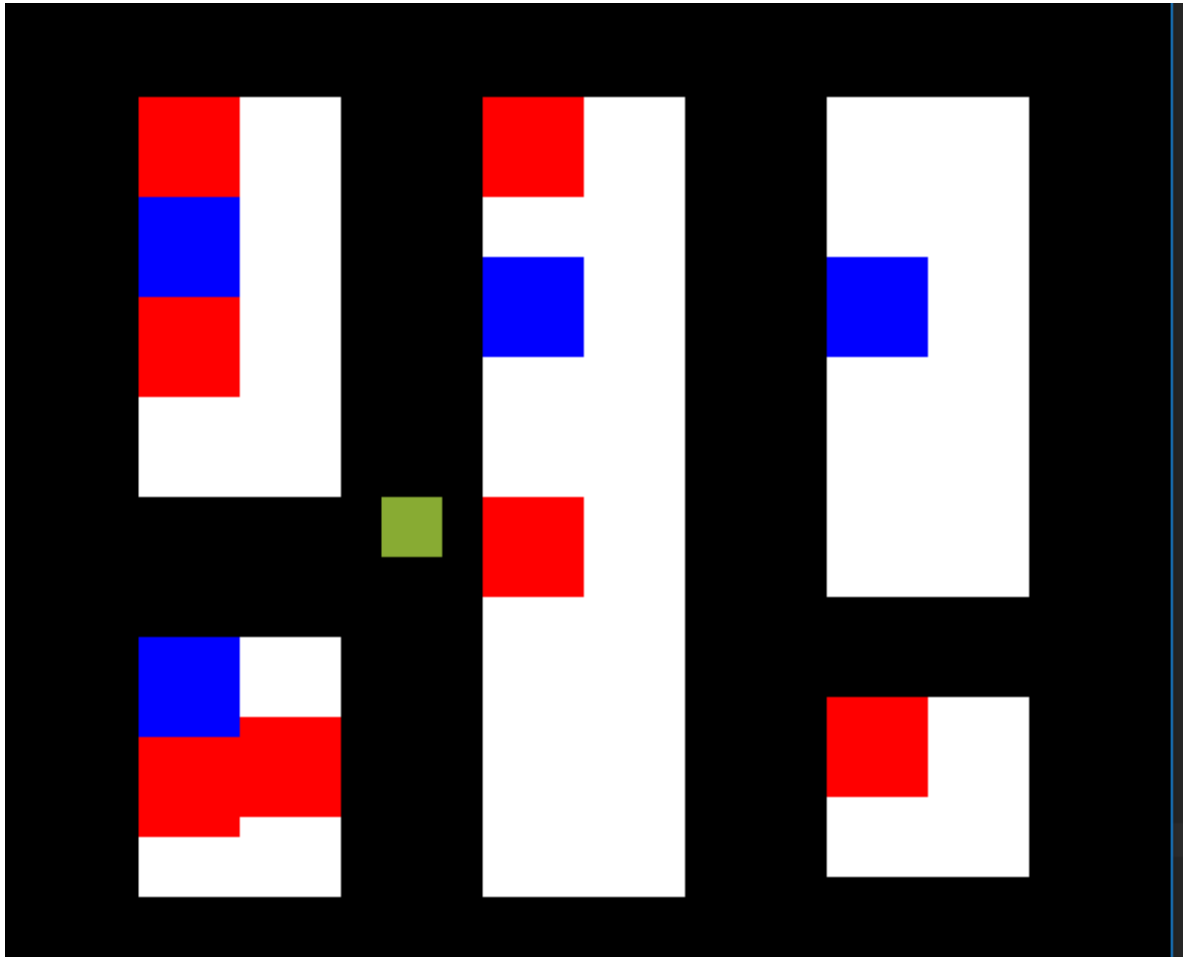
- KI zum Spielen des Echtzeitstrategiespiels Starcraft II
- Basis: Aufgezeichnete Spiele von professionellen SC II Spielern und Spielen gegen sich selbst
- Über 200 Jahre Spielerfahrung und Siege gegen professionelle Spieler

Weitere Implementierungen

- Super Mario. Bros
- Autofahren GTA V
- Flappy Bird
- ...

Entwicklung des Spiels

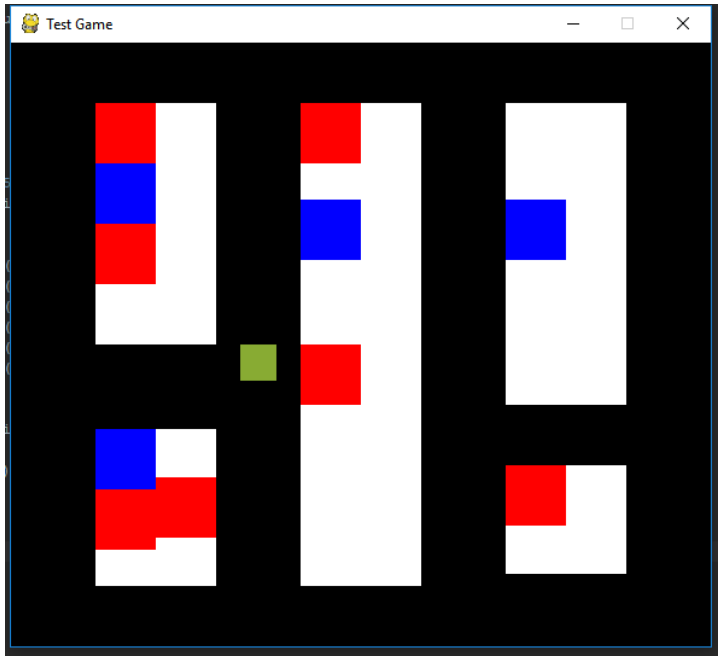
- Mit pygame (python Paket)
- Rechtecke, die alle 10 ms neu gezeichnet werden
- Input: Up, Down, Right, Left, Space
- 400 Züge
- Ziel: Einsammeln mind. Eines Items und finden des Ausgangs
- Sekundäres Ziel: Nicht gegen Regal (Game Over) oder Wand fahren (Punktabzug)

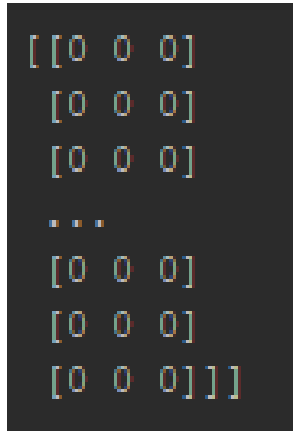


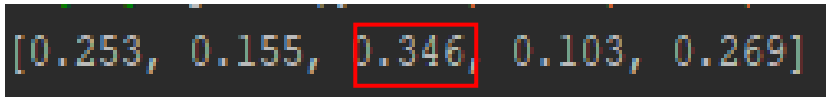
Punktesystem

- +6 pts - Einsammeln eines Items
- -4 pts - Fahren gegen die Wand
- -1 pts – Drücken der Leertaste (ohne Einsammeln eines Items)
- +(3 + Anzahl der verbliebenen Züge) pts – Finden des Ausgangs
- -10 pts – Fahren gegen Regal

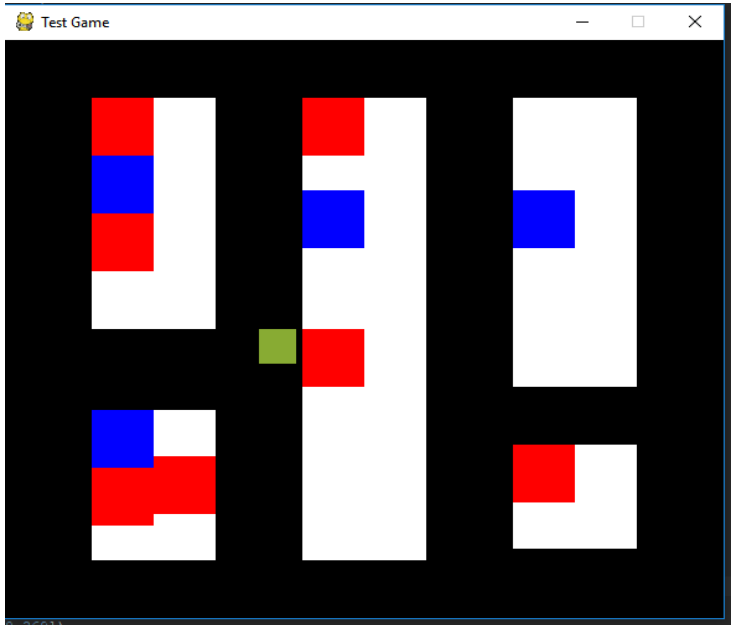
Wie spielt die KI?

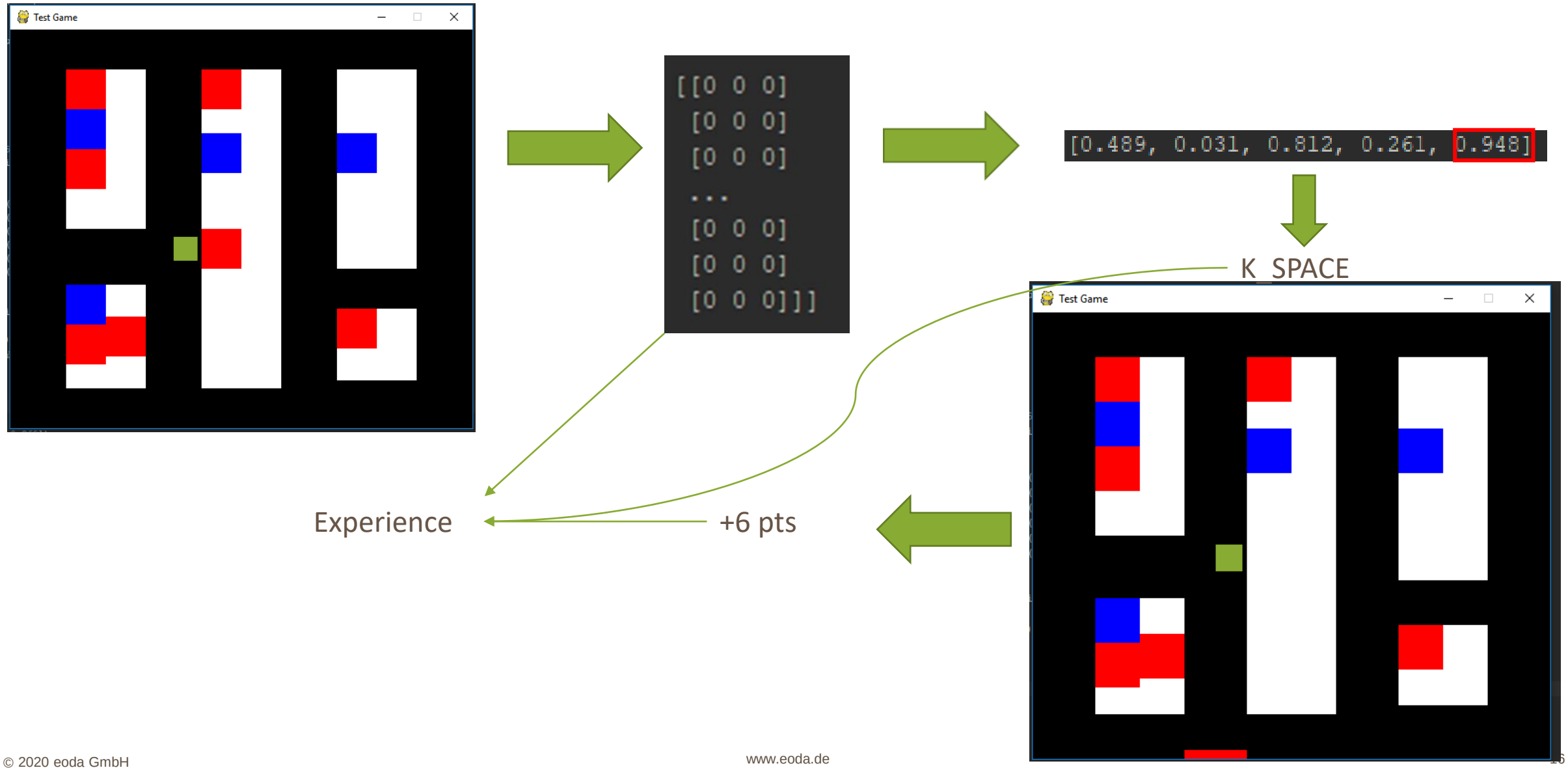






K_RIGHT

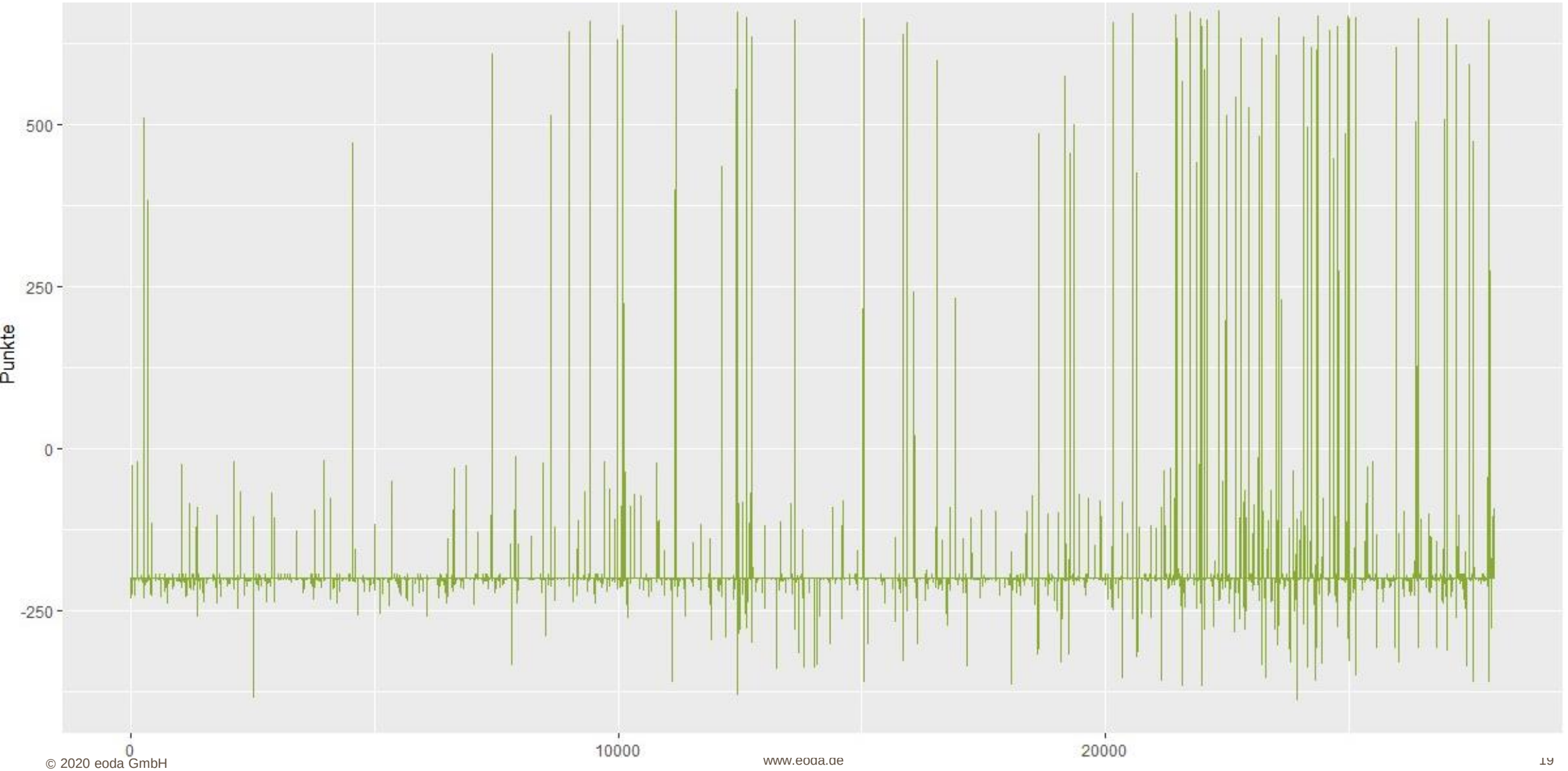


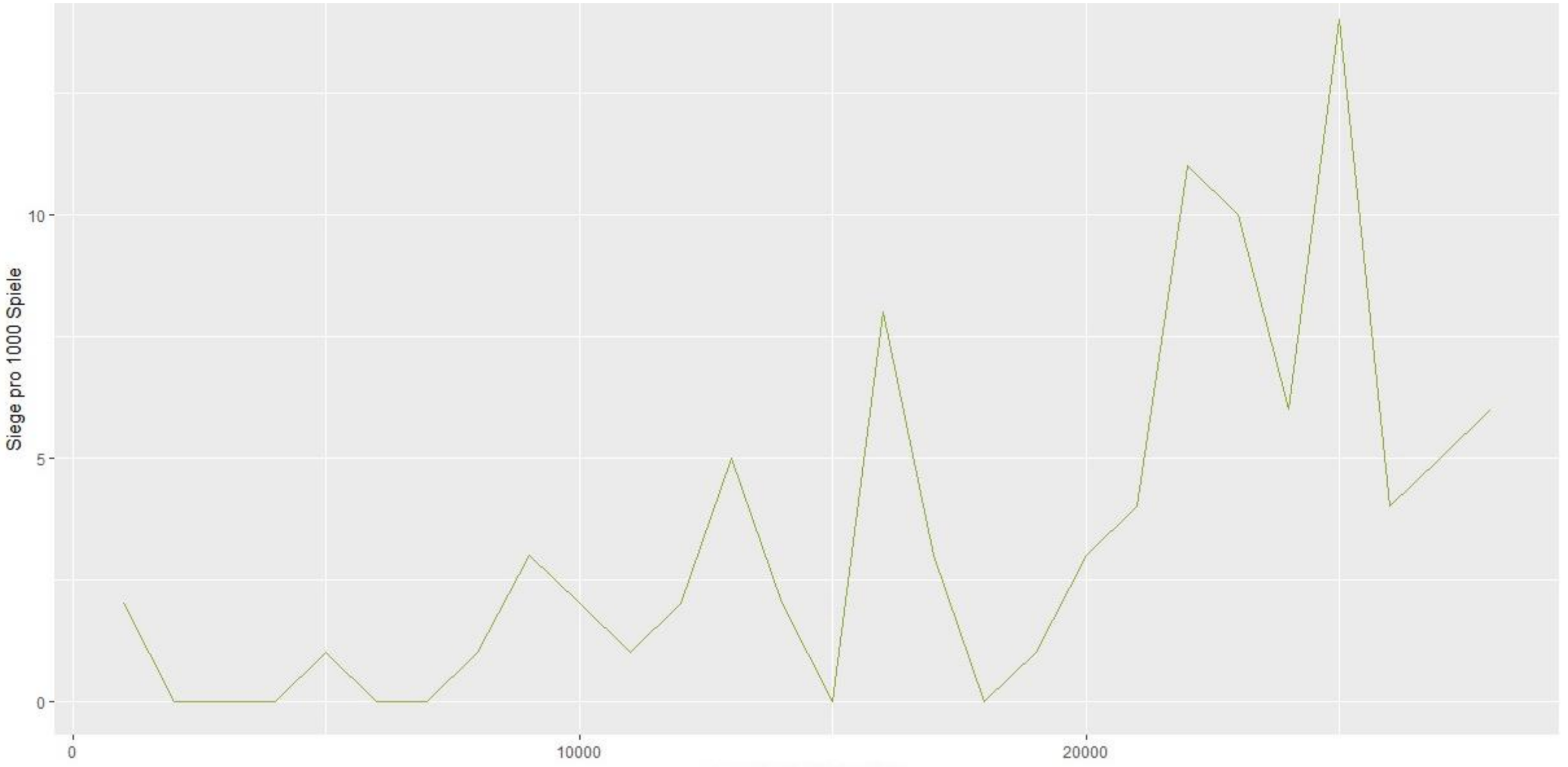

Test Game
[[0 0 0]
[0 0 0]
[0 0 0]
...
[0 0 0]
[0 0 0]
[0 0 0]]
[0.489, 0.031, 0.812, 0.261, 0.948]
K_SPACE
Test Game
Experience
+6 pts

Algorithmus: Designentscheidungen

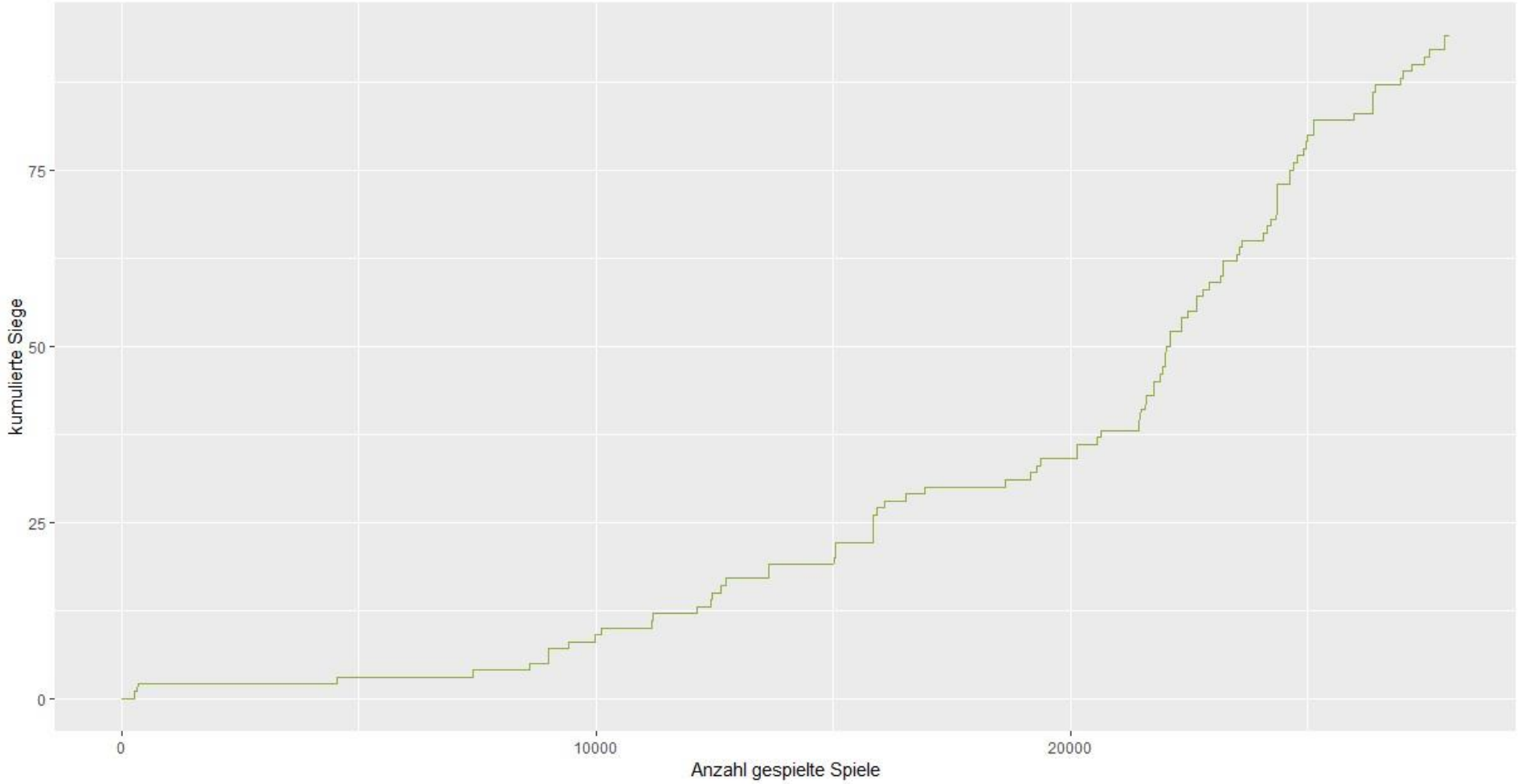
```
1
2  possibleInput = [K_UP, K_DOWN, K_RIGHT, K_LEFT, K_SPACE]
3  epsilon = 0.1
4  num_actions = 5  # left, right, up, down, pickup
5
6  batch_size = 15
7  discount = 0.7
8  grid_size = 580*500
9
10 hidden_size = 100
11
12 # function for model definition
13 def baseline_model(grid_size, num_actions, hidden_size):
14     #seting up the model with keras
15     model = Sequential()
16     model.add(Dense(hidden_size, input_shape=(grid_size,), activation='tanh'))
17     model.add(Dense(hidden_size, activation='tanh'))
18     model.add(Dense(5, activation = "linear"))
19     model.compile(sgd(lr=.01), "mse")
20     return model
```

Ergebnisse





Ergebnisse – kumulierte Siege



Verbesserungsmöglichkeiten

- Design des neuronalen Netz (Wahl der Aktivierungsfunktionen, Anzahl der Layer,…)
- Form des Inputs (1D/2D-Array, …)
- Punktevergabe
- Wahl der Parameter (Zufallszüge, batch-Size, …)
- Mehr Lernen: Bis jetzt „nur“ ca. 25000 Durchläufe


Jobs mit Perspektive in der Branche der Zukunft
Wir suchen Unterstützung in den Bereichen:
Data Science
Softwareentwicklung
IT-System Engineering
Sales
Mehr Informationen auf unserer Karriereseite.

Die Data Science Spezialisten.

eoda GmbH
Universitätsplatz 12
34127 Kassel
www.eoda.de
info@eoda.de
+49 561 202724-40

@eodaGmbH

blog.eoda.de

@eodaGmbH

eodaGmbH