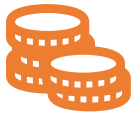
Chessmaster

By A. Bernrieder, N. Wichter, J. Brebeck, T. Hilbradt

Inhaltsverzeichnis

Einführung/Ziele

Algorithmen



Wirtschaftliche Aspekte

Auswertungen

Live-Demo

Ausblick

Einführung/Ziele

Einführung

- Warum eine Schach AI?
  - Schnell erlernte Grundregeln
  - Komplexe Prognose von Spielzügen
  - Nach 2 Zügen bereits 72.084 verschiedene Feldzustände

Ziele

- gegen menschliche Spieler spielen können
- ermöglichen seine Schachtaktiken zu üben
- eigenständig trainieren und verbessern

Algorithmen

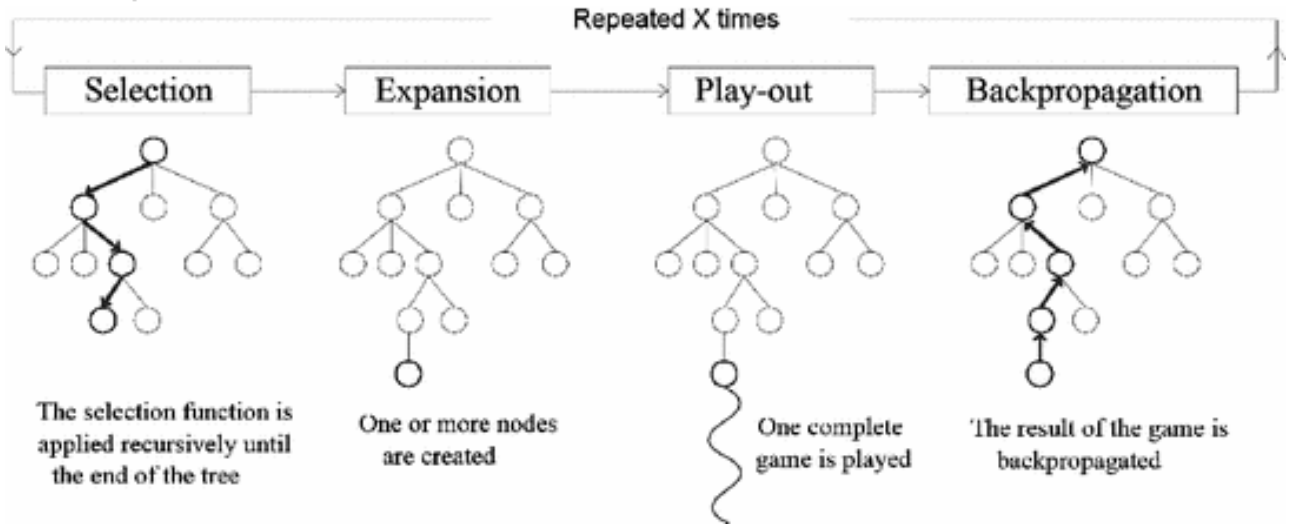
Algorithmen

- 3 aufeinanderaufbauende Algorithmen
  - Monte Carlo Tree Search
  - Temporal Difference Learning (SARSA)
  - Neuronales Netzwerk (Convolutional Neural Network)
- Spiele eine Anzahl Spiele und lerne dann von ihnen
  - Reinforcement → Verstärkend
- Gegner während Training?
  - Greedy Agent
    - wählt immer maximalen Schaden (--> Move Chess)

Monte Carlo Tree Search

- Effiziente Suche r
- Nutzen während
- Probleme:
  - Hoher Rechenaufwand
    - Simulationen
    - Iterationen
- Problemlösung
  - Simulation → Zufallsspiel
  - Iterationen → Agent
- (Herkunft:
  - Eigene Implementierung)

Wähle Node mit höchstem Wert (UCB1-Algorithm)	$UCB1(S_i) = \bar{v} + c * \sqrt{\frac{\ln(N)}{n_i}}$	$\bar{v}$ = Durchschnittswert der node $\left(\frac{v}{ni}\right)$ $c = Exploration\ visiteparameter\ (statisch)$ $N = Anzahl\ Besuche\ parent\ node$ $n_i = Anzahl\ Besuche\ i-th\ node$



SARSA Lernalgortihmus

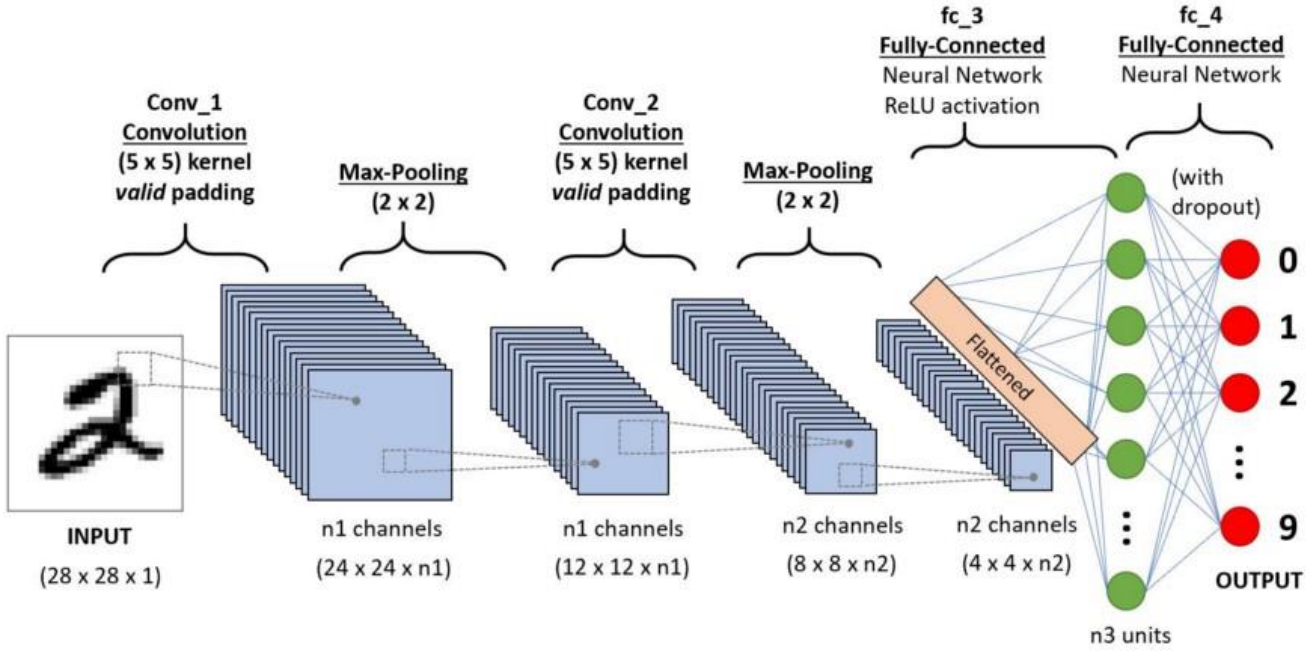
- State-Action-Reward-State-Action
- Input
  - Game Board (von Node)
- Training:
  - Aktueller Agent: Wert vor Zug predicten
  - MCTS errechnet Wert und führt Zug aus
  - Aktueller Agent: Wert nach Zug predicten
  - Mini-Batch (n Spiele)
  - Trainiere neuen Agent

$$V(S_t) = V(S_t) + \alpha * (G_t - V(S_t))$$

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha * (r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

Convolutional Neural Network

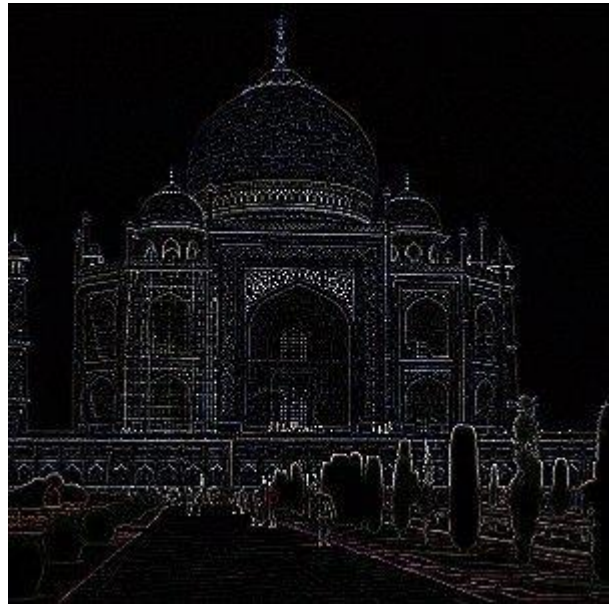
- Convolutional layers
- Pooling layers
- Flattening
- Fully connected layers



´Quelle: https://miro.medium.com/max/875/1*uAeANQIOQPqWZnnuH-VEyw.jpeg

Convolutional Neural Network

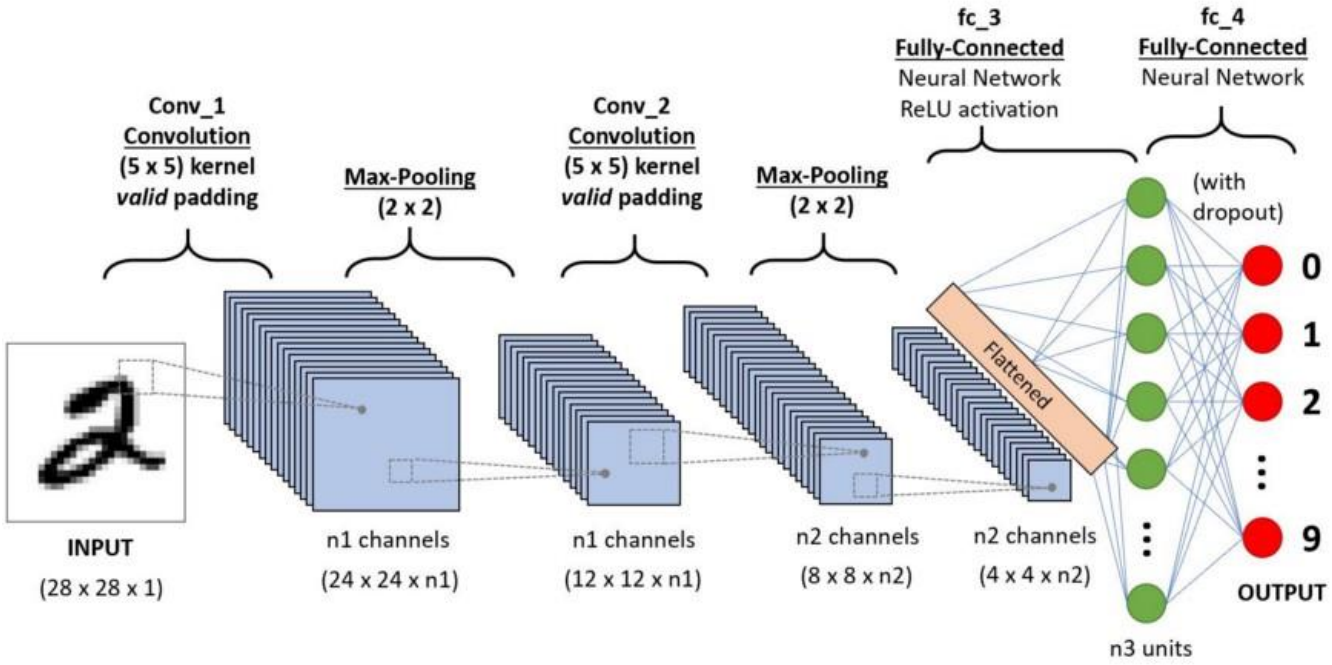
	0	1	0	
	1	-4	1	
	0	1	0	



https://docs.gimp.org/2.8/en/plug-in-convmatrix.html

Convolutional Neural Network

- Input:
  Schachbrettzustand eines Nodes
- Architektur:
  - 7 Conv2D Layer
  - 7 Flatten Layer
  - 6 Dense / fully connected Layer
- Output: Wert des Boards



https://miro.medium.com/max/875/1*uAeANQIOQPqWZnnuH-VEyw.jpeg

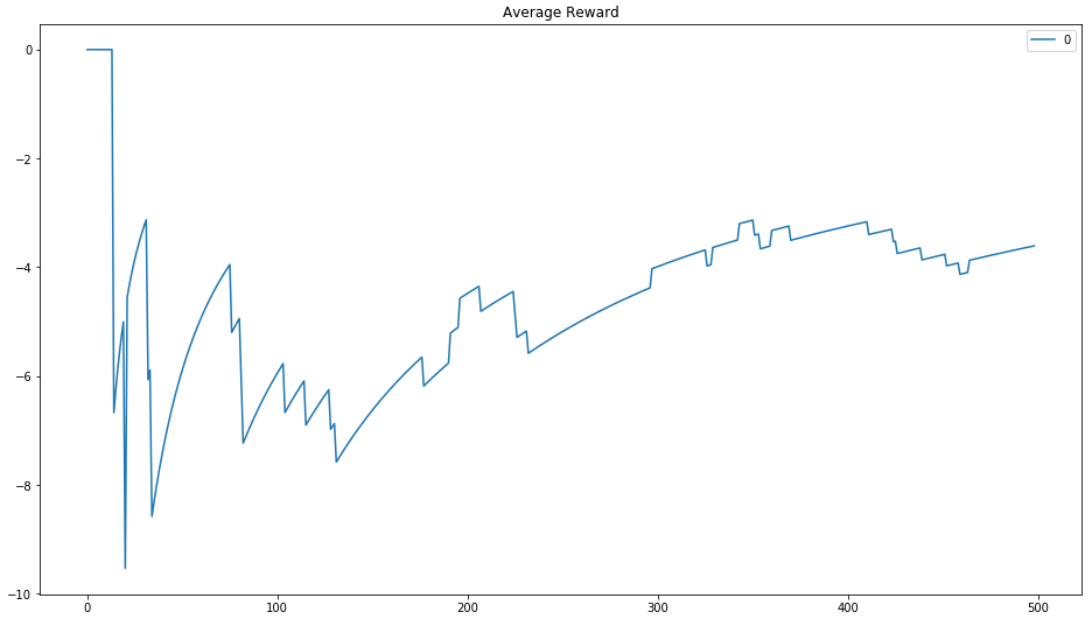
Wirtschaftliche Aspekte

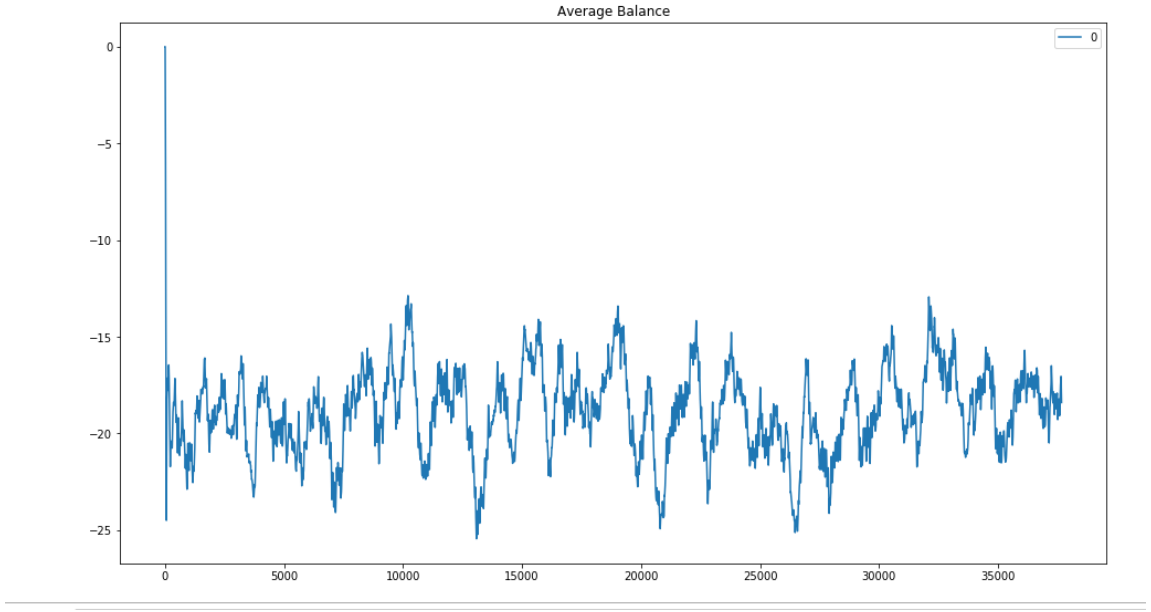
Business-Modell

- Trainingsprogramm für Schachspieler
- Evaluierungstool für AI‘s
  - Elo-Zahl als Maß für Spielerstärke
- Grundlage für andere Ais

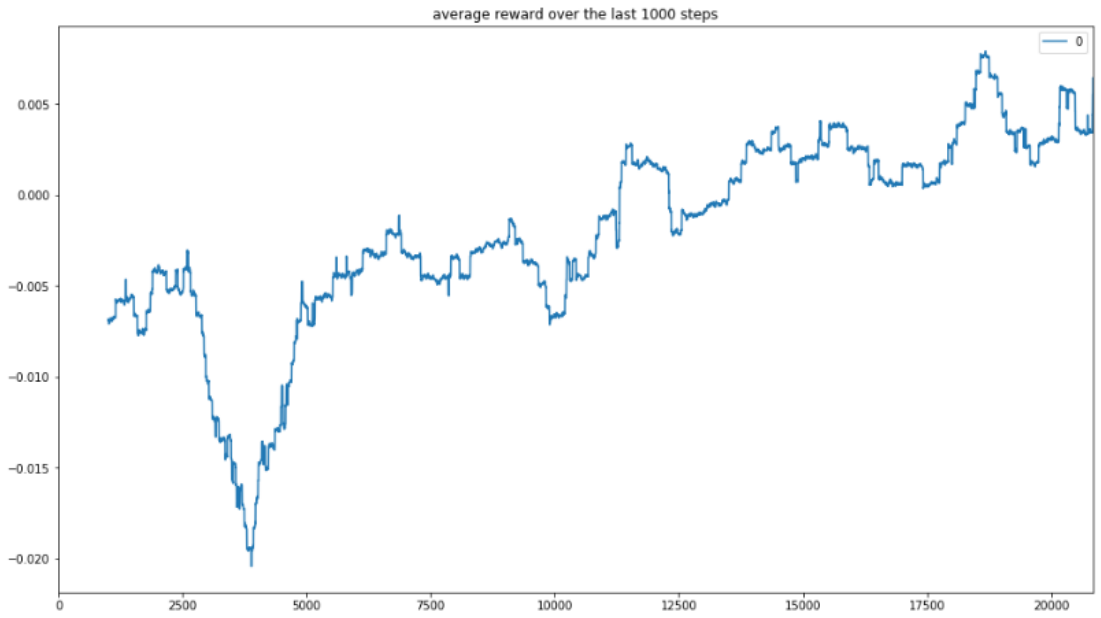
Auswertung

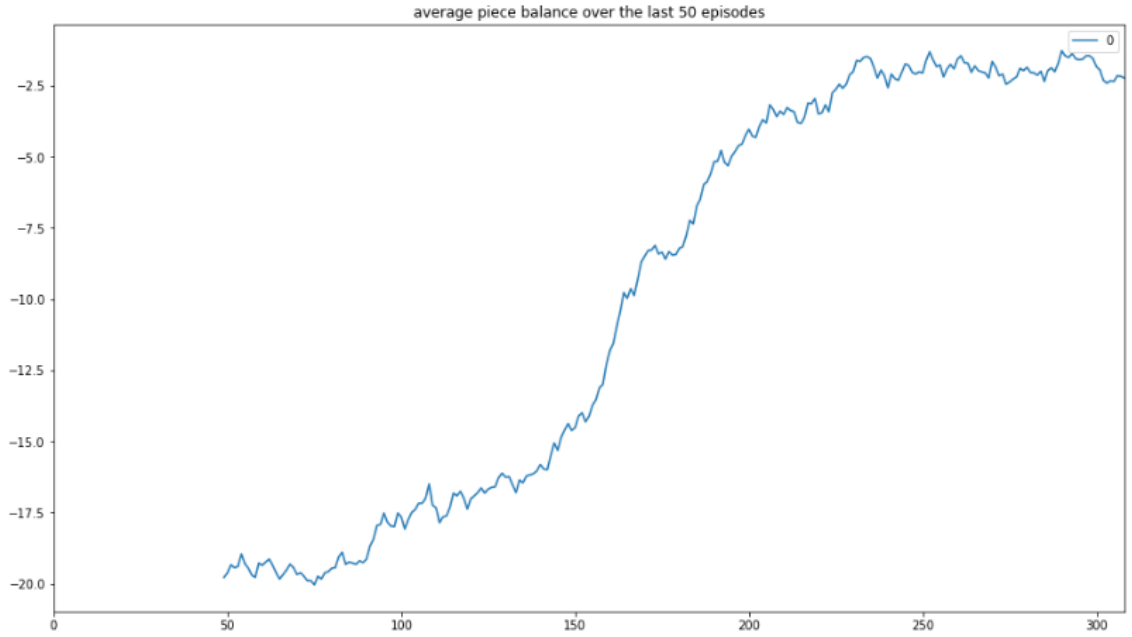
Auswertung des Modells





Auswertung des Modells





Demonstration

Ausblick

Ausblick

- Erheblich höhere Trainingszeit verbessert Modell
  - ca. 800.000 Iterationen bei Mu Zero
- Anpassung an ähnliche Spiele
  - Shogi, Go, etc.
- Verwendung in E-Sport

Quellen

- https://hci.iwr.uni-heidelberg.de/system/files/private/downloads/297868474/report_robert-klassert.pdf
- https://towardsdatascience.com/reinforcement-learning-temporal-difference-sarsa-q-learning-expected-sarsa-on-python-9fecfda7467e
- http://www.informatik.uni-ulm.de/ni/Lehre/SS07/RL/vorlesung/rl06.pdf
- http://elo-kompetenzteam.de/2018/04/07/die-elo-zahl/
- http://www.sfbux.de/wp-content/uploads/artikel/berechenbarkeit.pdf
- https://www.kaggle.com/arjanso/reinforcement-learning-chess-5-tree-search

Danke für die Aufmerksamkeit