OthelloZero

Vom leeren Brett zur Meisterstrategie

Leo Pracht

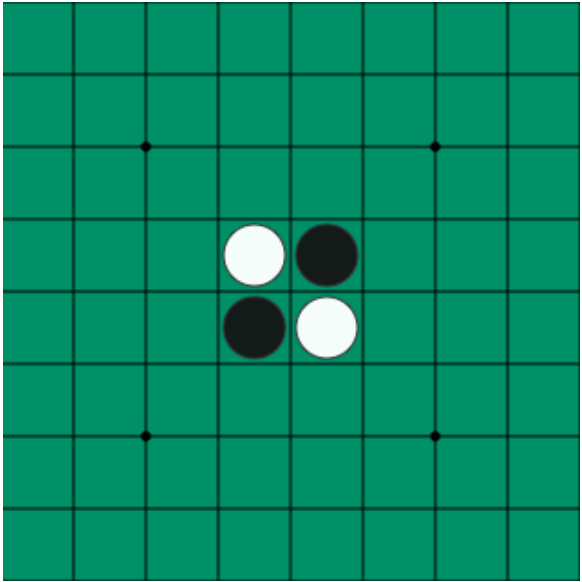
Kantonsschule Kreuzlingen

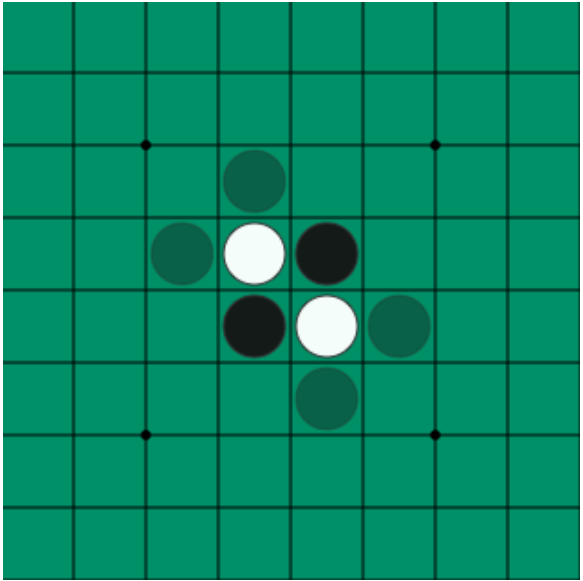
Maturitätsarbeit 2025

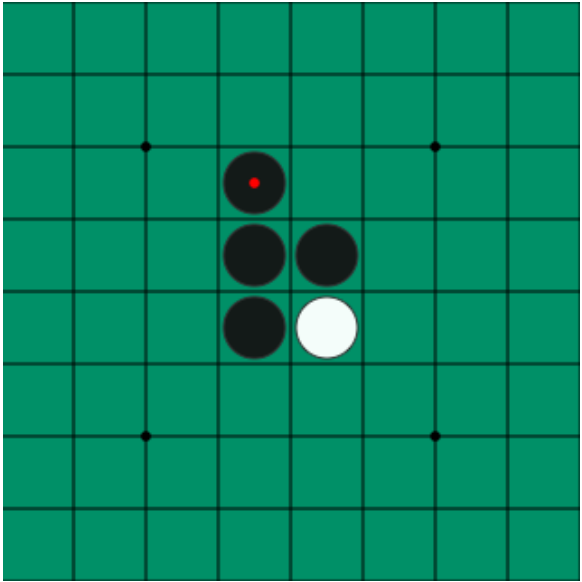
Motivation

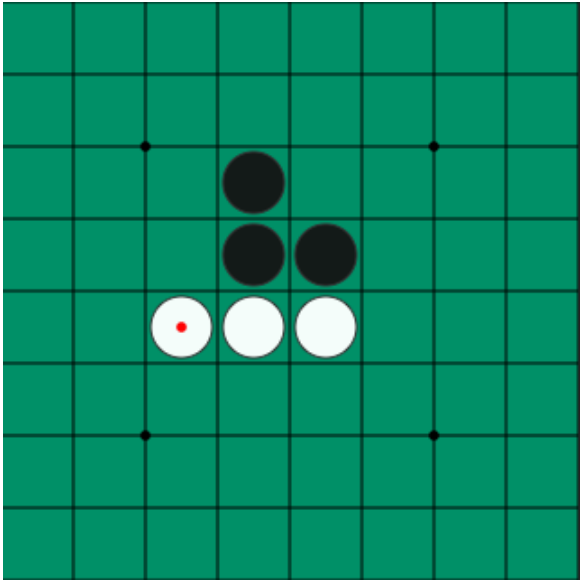
„Wie kann ich meinem Computer etwas beibringen, das ich selbst nicht kann?“

— *Meine Motivation für OthelloZero*









Funktionstheorie

Was ist eine Funktion?

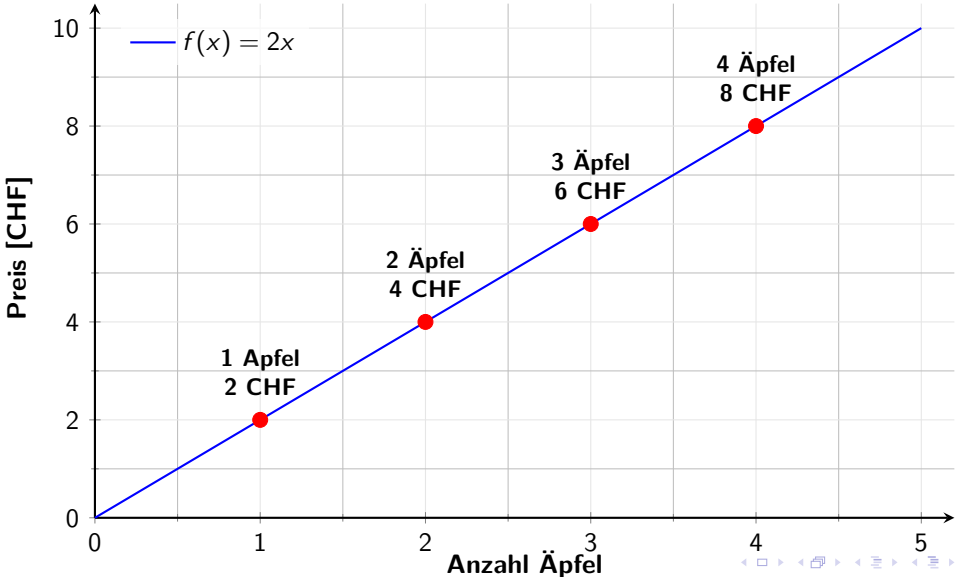
Eine **Funktion** ist ein System mit *Eingaben* und *Ausgaben*:

$$x \xrightarrow{\text{Eingabe}} f(x) \xrightarrow{\text{Ausgabe}} y$$

Funktionen beschreiben die Welt!

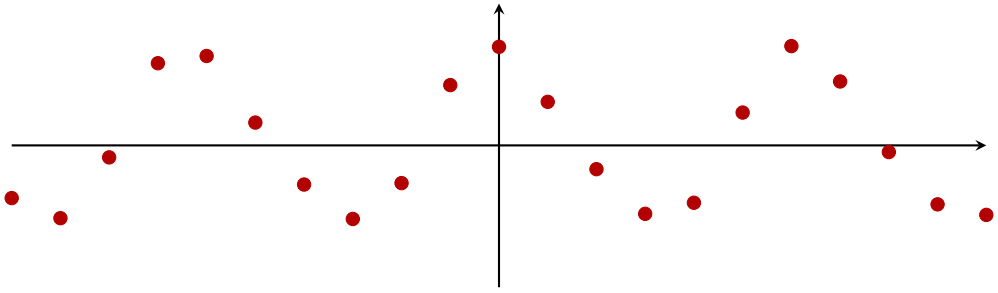
Das Licht, das wir sehen, die Töne, die wir hören – sogar Ihr momentaner Puls – alles lässt sich durch Funktionen beschreiben.

Funktionen grafisch darstellen



Was, wenn wir die Funktion nicht kennen?

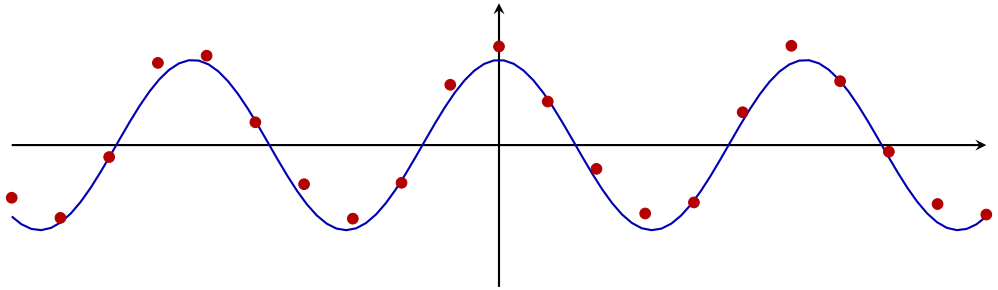
- Wir kennen nur einzelne Punkte – also die **x- und y-Werte**.
- Können wir daraus die zugrunde liegende Funktion rekonstruieren?



⇒ **Was wir brauchen, ist einen Funktionsapproximator – das ist ein neuronales Netz (KI).**

Von Datenpunkten zur generalisierten Funktion

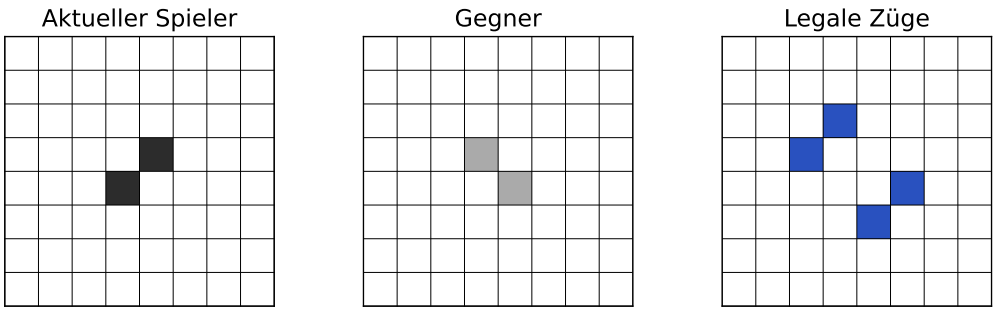
- Die KI hat das Muster gelernt – sie kann nun auch neue Werte vorhersagen.
- Selbst wenn die Daten verrauscht sind, bleibt das Grundmuster erhalten.



⇒ **Aus der approximierten Funktion lassen sich neue Werte erzeugen – auch solche, die im Datensatz nie vorkamen.**

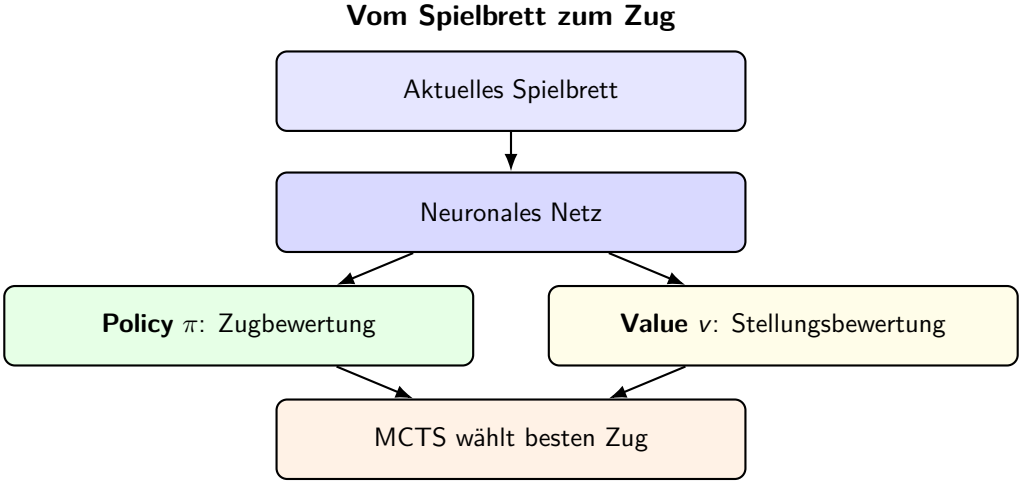
Eingabe in das neuronale Netz

Netzwerkinput (3×8×8)



Im Fall von **OthelloZero** erhält das neuronale Netz nicht nur eine einzelne Zahl,
sondern eine **komplette Spielfeldkonfiguration**:
Positionen der eigenen und gegnerischen Steine sowie die gültigen Züge.

Neuronale Netze in OthelloZero

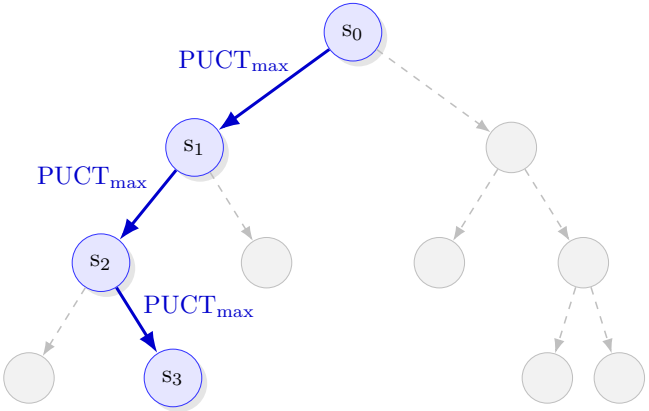


⇒ Das Netz liefert **Policy (π)** und **Value (v)**, die die MCTS zur Zugwahl kombiniert.

MCTS – Überblick

- **Ziel:** Durch viele kurze „Gedankenspiele“ herausfinden, welcher Zug am besten ist.
- **Drei Schritte:**
  1. **Selektion** – zuerst dort weiterspielen, wo es spannend oder vielversprechend aussieht.
  2. **Erweiterung** – einen neuen Zug ausprobieren, um Neues zu entdecken.
  3. **Rücktragen** – das Ergebnis merken und daraus lernen.

MCTS mit PUCT-Selektion



- Hauptselektionspfad ($s_0 \to s_1 \to s_2 \to s_3$)
- Alternative Pfade
- Leere Knoten: Unbesucht

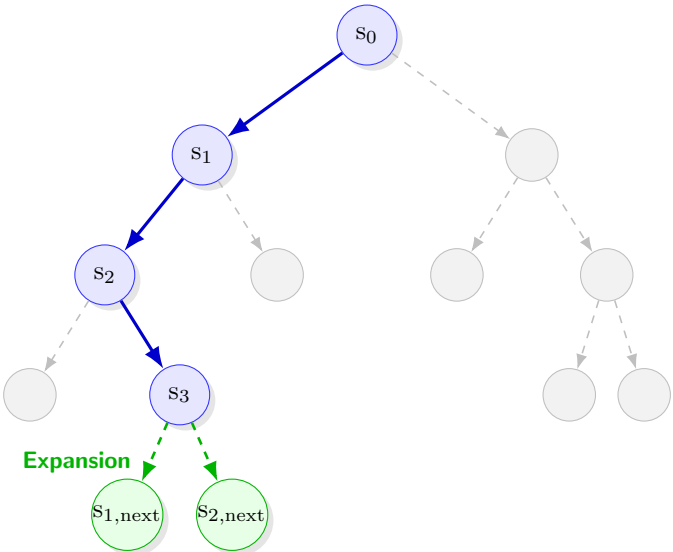
PUCT, ganz kurz

$$a^* = \arg\max_a \left[Q(s,a) + U(s,a) \right]$$

$$U(s,a) = c_{\text{puct}} \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

- Q = **Erfahrung:** Wie gut lief dieser Zug bisher in unseren Probepartien?
- U = **Neugier:** Netz-Prior P pusht vielversprechende Züge; selten getestete (N klein) werden bevorzugt.
- $\mathbf{c}_{\text{puct}}$ = Neugier-Regler: steuert Balance zwischen Q (bewährt) und U (entdecken).
- **Ergebnis:** Wir wählen, was *bewährt* + *vielversprechend* ist.

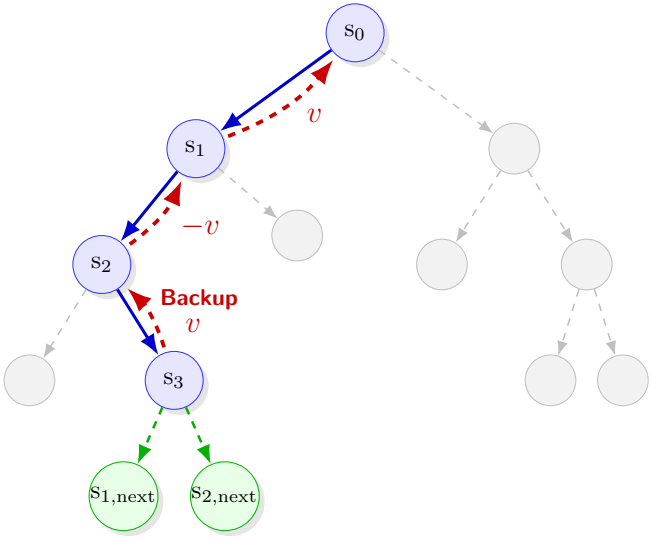
MCTS mit Selektion und Expansion



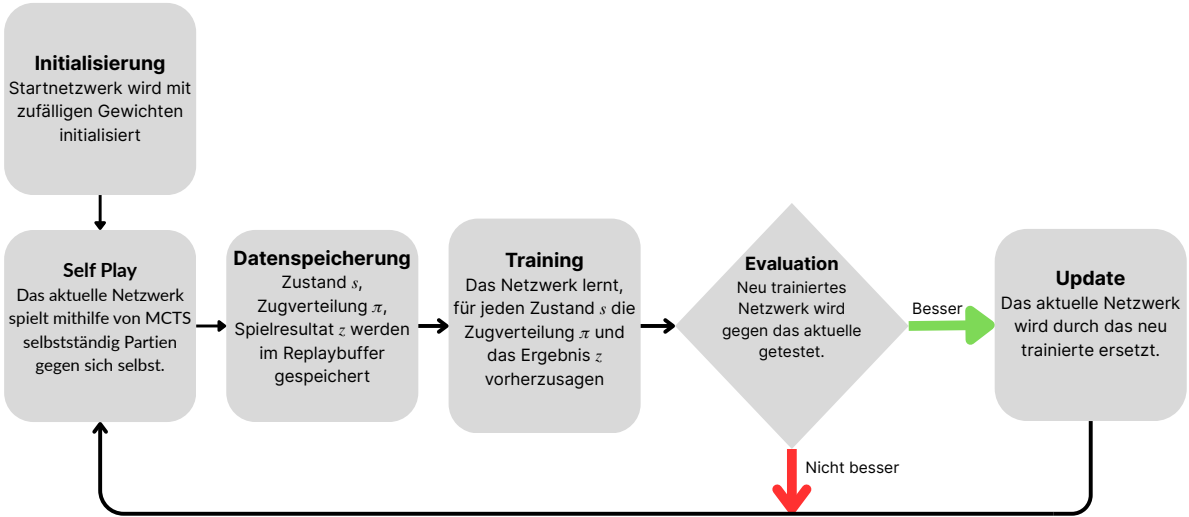
- Selektion
- Expansion $s_{1,\text{next}}$, $s_{2,\text{next}}$
- Unbesuchte Knoten

MCTS Backpropagation

Schritt 3: Wert-Update entlang des Pfades

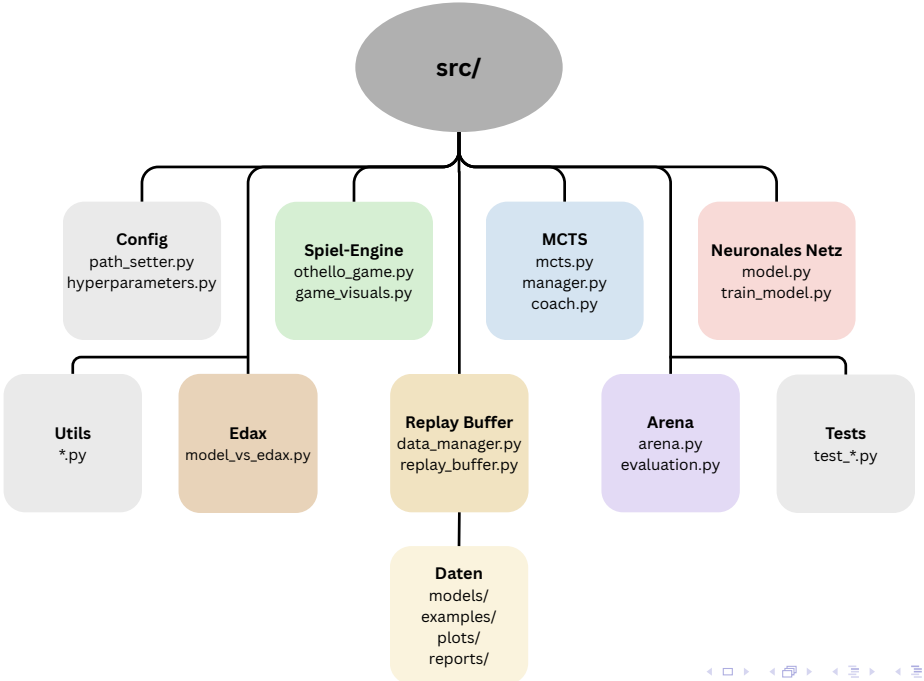


- Selektion
- Expansion
- Backpropagation s_1, s_2, s_3
- Unbesuchte Knoten


Initialisierung
Startnetzwerk wird mit zufälligen Gewichten initialisiert
Self Play
Das aktuelle Netzwerk spielt mithilfe von MCTS selbstständig Partien gegen sich selbst.
Datenspeicherung
Zustand s, Zugverteilung π, Spielresultat z werden im Replaybuffer gespeichert
Training
Das Netzwerk lernt, für jeden Zustand s die Zugverteilung π und das Ergebnis z vorherzusagen
Evaluation
Neu trainiertes Netzwerk wird gegen das aktuelle getestet.
Besser
Update
Das aktuelle Netzwerk wird durch das neu trainierte ersetzt.
Nicht besser

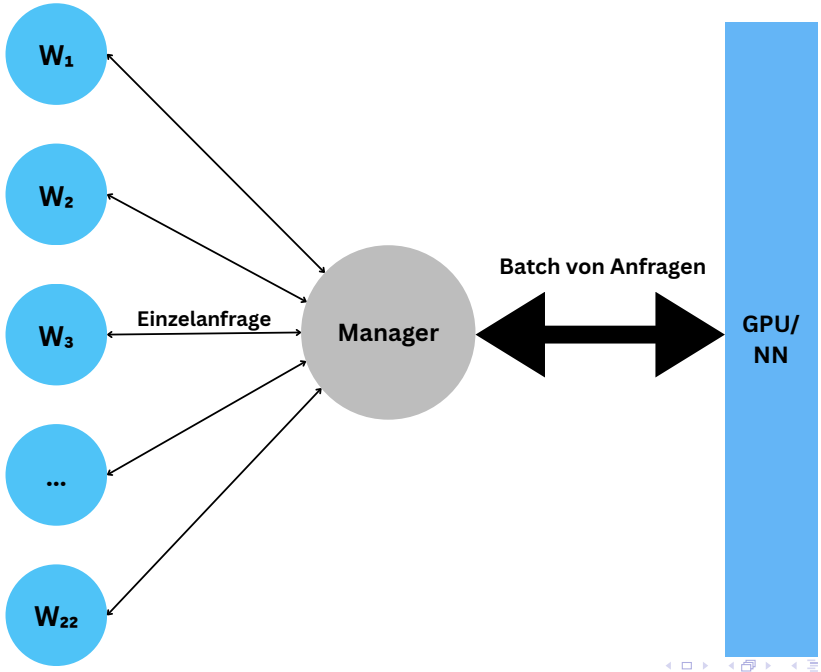
Umsetzung

- Implementierung in Python
- Training auf **RTX 4060** mit **22 parallelen Prozessen**
- Trainingsdauer: rund **48 Stunden** für **55 Generationen**
- Pro Iteration: **264 Partien** mit etwa **126 000 Datenpunkten**
- Insgesamt sah das Netz rund **7 Millionen Spielpositionen**


src/
Config
path_setter.py
hyperparameters.py
Spiel-Engine
othello_game.py
game_visuals.py
MCTS
mcts.py
manager.py
coach.py
Neuronales Netz
model.py
train_model.py
Utils
*.py
Edax
model_vs_edax.py
Replay Buffer
data_manager.py
replay_buffer.py
Arena
arena.py
evaluation.py
Tests
test_*.py
Daten
models/
examples/
plots/
reports/

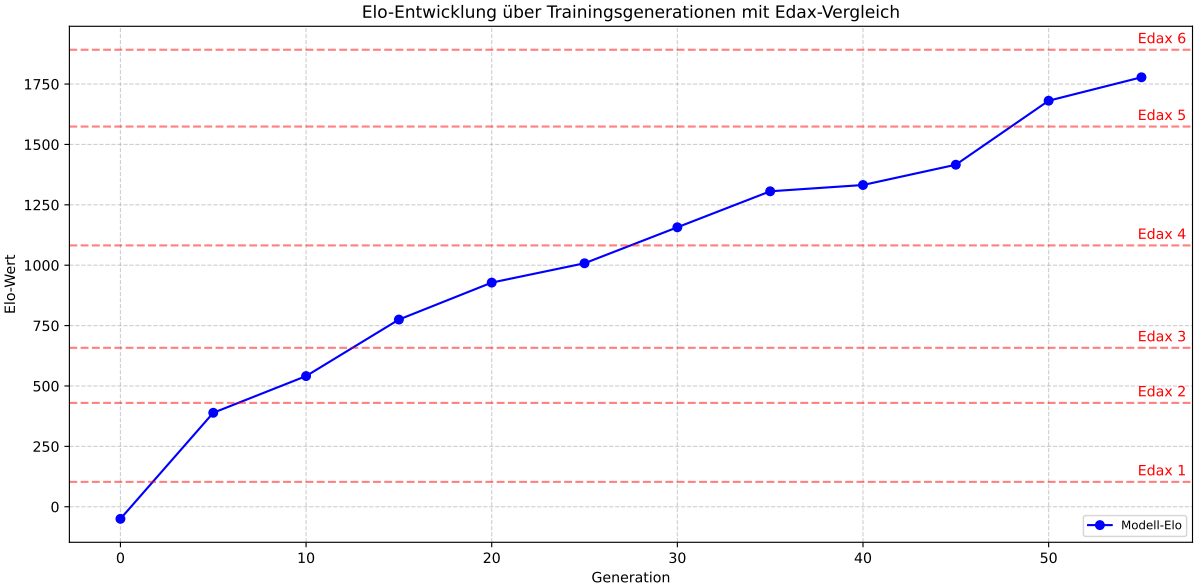
Umsetzung

- Implementierung in Python
- Training auf **RTX 4060** mit **22 parallelen Prozessen**
- Trainingsdauer: rund **48 Stunden** für **55 Generationen**
- Pro Iteration: **264 Partien** mit etwa **126 000 Datenpunkten**
- Insgesamt sah das Netz rund **7 Millionen Spielpositionen**

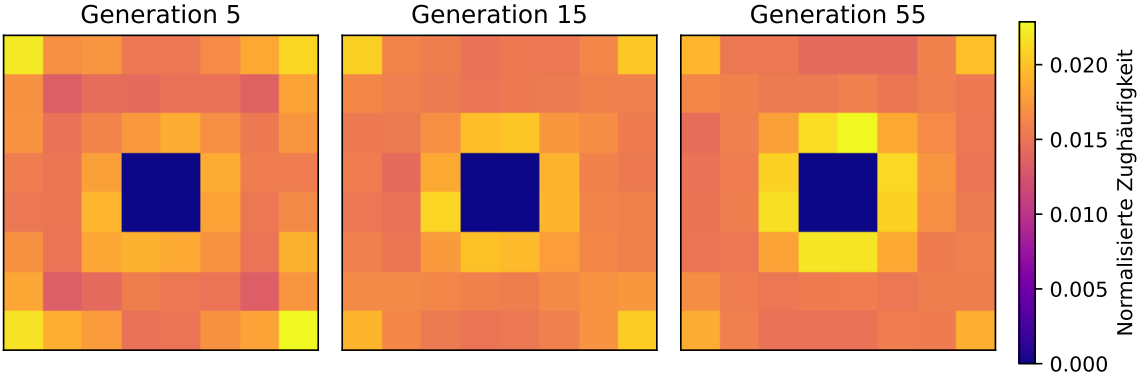

W1
W2
W3
...
W22
Einzelanfrage
Manager
Batch von Anfragen
GPU/
NN

Umsetzung

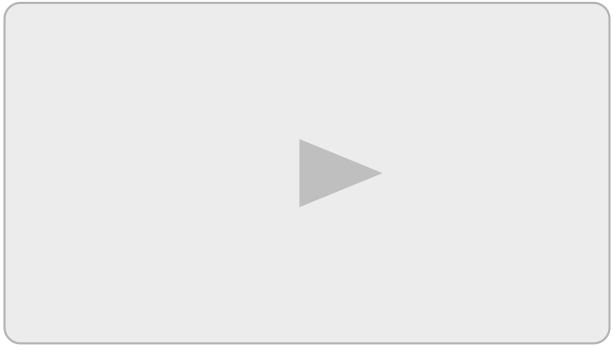
- Implementierung in Python
- Training auf **RTX 4060** mit **22 parallelen Prozessen**
- Trainingsdauer: rund **48 Stunden** für **55 Generationen**
- Pro Iteration: **264 Partien** mit etwa **126 000 Datenpunkten**
- Insgesamt sah das Netz rund **7 Millionen Spielpositionen**


Elo-Entwicklung über Trainingsgenerationen mit Edax-Vergleich
Edax 6
Edax 5
Edax 4
Edax 3
Edax 2
Edax 1
Modell-Elo
Elo-Wert
Generation
1750
1500
1250
1000
750
500
250
0
0
10
20
30
40
50

Policy-Heatmap: Aufmerksamkeit des Modells



OthelloZero Demo



„OthelloZero“ im Mittel- bis Endspiel

Modell (weiß) vs. Leo (schwarz) · Clipdauer: ca. 40 s

Methodenkritik

- **Rechenleistung:** Begrenzte Trainingszeit, weniger Self-Play-Partien
- **Implementierung:** Python statt C++/Rust – Fokus auf Verständlichkeit
- **Zufallseinflüsse:** Ergebnisse leicht variabel (stochastisches Training)
- **Evaluation:** Nur Vergleich mit Edax, keine menschliche Referenz

$\rightarrow$ *Ziel war Nachvollziehbarkeit statt maximaler Leistung.*

Fazit & Danksagung

„Intelligenz entsteht nicht aus Kraft, sondern aus Einsicht.“

— frei nach David Silver / Demis Hassabis

Und genau das ist für mich das Faszinierende an AlphaZero – und an OthelloZero:

Intelligenz entsteht nicht durch Rechenleistung,
sondern durch Struktur, Strategie und Lernen.

Danksagung

Mein Dank gilt meiner Betreuerin für die wertvolle Unterstützung,
allen, die Interesse an meiner Arbeit gezeigt haben,
und ganz besonders meiner Mutter, die mir die KSK überhaupt ermöglicht hat.