

# Investigation of possible improvements to increase the efficiency of the AlphaZero algorithm.

Colin Clausen

X.X.2020

1. Einleitung
2. Grundlagen
3. Untersuchte neue Ideen
4. Ende

1. Einleitung
2. Grundlagen
3. Untersuchte neue Ideen
4. Ende



- ▶ Spiele als Lernumgebung für AI

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt
  - ▶ 1997: Schach



- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt
  - ▶ 1997: Schach
  - ▶ 2016: Go

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt
  - ▶ 1997: Schach
  - ▶ 2016: Go
- ▶ AlphaGo

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt
  - ▶ 1997: Schach
  - ▶ 2016: Go
- ▶ AlphaGo
- ▶ AlphaZero

- ▶ Spiele als Lernumgebung für AI
  - ▶ 1951: Nimrod
  - ▶ 1995: Vier gewinnt
  - ▶ 1997: Schach
  - ▶ 2016: Go
- ▶ AlphaGo
- ▶ AlphaZero
- ▶ Großer Rechenaufwand nötig: 5000+ TPUs



- ▶ Untersuche mögliche Effizienzsteigerungen

- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline

- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline
- ▶ Experimentiere mit Vier gewinnt



- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline
- ▶ Experimentiere mit Vier gewinnt
- ▶ Evaluiere verschiedene neue Ideen

- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline
- ▶ Experimentiere mit Vier gewinnt
- ▶ Evaluiere verschiedene neue Ideen
  - ▶ Evolution von Hyperparametern

- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline
- ▶ Experimentiere mit Vier gewinnt
- ▶ Evaluiere verschiedene neue Ideen
  - ▶ Evolution von Hyperparametern
  - ▶ Self-play im Baumformat

- ▶ Untersuche mögliche Effizienzsteigerungen
- ▶ Solide Baseline
- ▶ Experimentiere mit Vier gewinnt
- ▶ Evaluiere verschiedene neue Ideen
  - ▶ Evolution von Hyperparametern
  - ▶ Self-play im Baumformat
  - ▶ Auxiliary Features

1. Einleitung

2. Grundlagen

Algorithmus

Baselines

3. Untersuchte neue Ideen

4. Ende

# Monte Carlo tree search (MCTS)

Grundlagen ▷ Algorithmus

# Monte Carlo tree search (MCTS)

Grundlagen ▷ Algorithmus

- ▶ Seit 2006 sehr verbreitet in Computer Go.

# Monte Carlo tree search (MCTS)

Grundlagen ▷ Algorithmus

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero



# Monte Carlo tree search (MCTS)

Grundlagen ▷ Algorithmus

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:
  - ▶ Eine Policy die Züge einschätzen kann

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:
  - ▶ Eine Policy die Züge einschätzen kann
  - ▶ Eine sehr schnelle Rolloutpolicy

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:
  - ▶ Eine Policy die Züge einschätzen kann
  - ▶ Eine sehr schnelle Rolloutpolicy
    - ▶ Alternativ: Policy zur Positionseinschätzung

# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:
  - ▶ Eine Policy die Züge einschätzen kann
  - ▶ Eine sehr schnelle Rolloutpolicy
    - ▶ Alternativ: Policy zur Positionseinschätzung
- ▶ Output: Eine bessere Policy über die Züge in der analysierten Position



# Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go.
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
  - ▶ Wandere den Baum von der Wurzel hinab
  - ▶ Wäge ab zwischen bekannten guten Zügen und wenig untersuchten neuen Zügen
  - ▶ Schließlich propagiere Spielergebnis zurück durch den Baum
- ▶ Benötigt:
  - ▶ Eine Policy die Züge einschätzen kann
  - ▶ Eine sehr schnelle Rolloutpolicy
    - ▶ Alternativ: Policy zur Positionseinschätzung
- ▶ Output: Eine bessere Policy über die Züge in der analysierten Position
- ▶ MCTS ist praktisch ein Verbesserungsoperator



- ▶ Kernidee: Kombiniere MCTS mit Deep Learning

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - ▶ Verbessere das langsame Netz durch RL

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - ▶ Verbessere das langsame Netz durch RL
  - ▶ Erzeuge Datensatz für Netzwerk zur Positionsevaluierung



- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - ▶ Verbessere das langsame Netz durch RL
  - ▶ Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - ▶ Verwende erzeugte Netzwerke um mit MCTS zu spielen

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - ▶ Verbessere das langsame Netz durch RL
  - ▶ Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - ▶ Verwende erzeugte Netzwerke um mit MCTS zu spielen
    - ▶ Das langsame RL-Netzwerk macht die Ersteinschätzung der Züge

- ▶ Kernidee: Kombiniere MCTS mit Deep Learning
- ▶ Trainingsprozess involviert aber kein MCTS
- ▶ Trainiere mehrere Netzwerke
  - ▶ Supervised auf Datensatz von besten Menschen: Schnell und Langsam
  - ▶ Verbessere das langsame Netz durch RL
  - ▶ Erzeuge Datensatz für Netzwerk zur Positionsevaluierung
  - ▶ Verwende erzeugte Netzwerke um mit MCTS zu spielen
    - ▶ Das langsame RL-Netzwerk macht die Ersteinschätzung der Züge
    - ▶ Einschätzung der Spielposition: Netzwerk + Rollouts



- ▶ Drastische Vereinfachung von AlphaGo

- ▶ Drastische Vereinfachung von AlphaGo
- ▶ Kernidee: Verwende MCTS bereits zur Trainingsphase

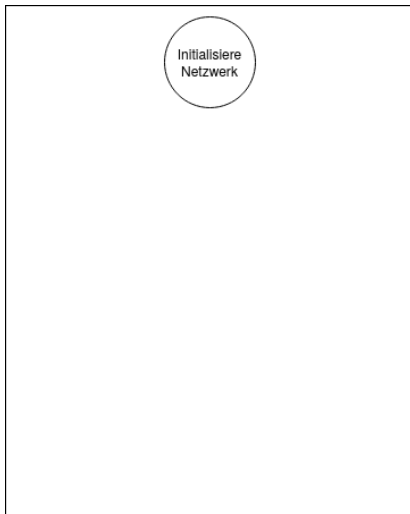
- ▶ Drastische Vereinfachung von AlphaGo
- ▶ Kernidee: Verwende MCTS bereits zur Trainingsphase
- ▶ Verwendet nur ein Netzwerk: Positionsbewertung und Zugpolicy

- ▶ Drastische Vereinfachung von AlphaGo
- ▶ Kernidee: Verwende MCTS bereits zur Trainingsphase
- ▶ Verwendet nur ein Netzwerk: Positionsbewertung und Zugpolicy
- ▶ Kein Bedarf für Datensatz von besten Menschen



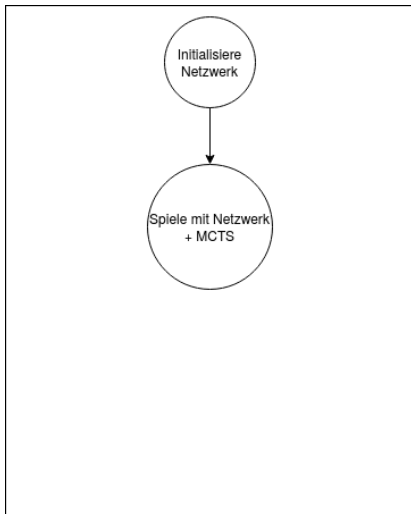
# AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



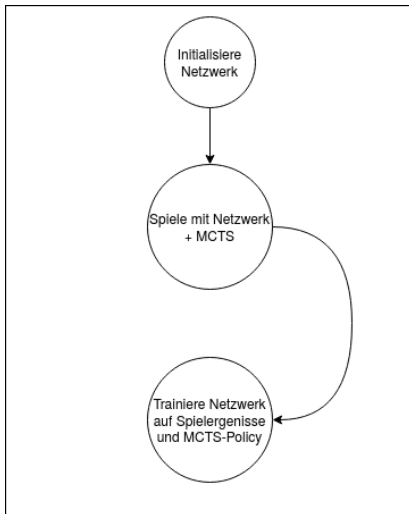
# AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



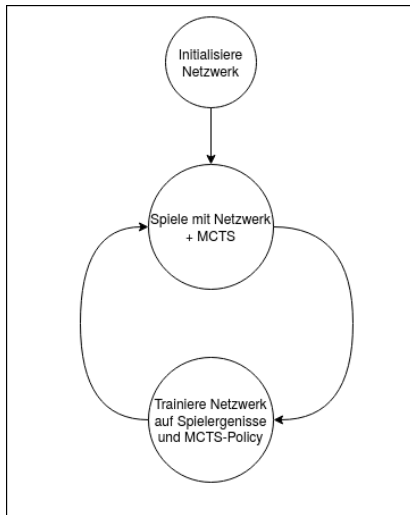
# AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



# AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus

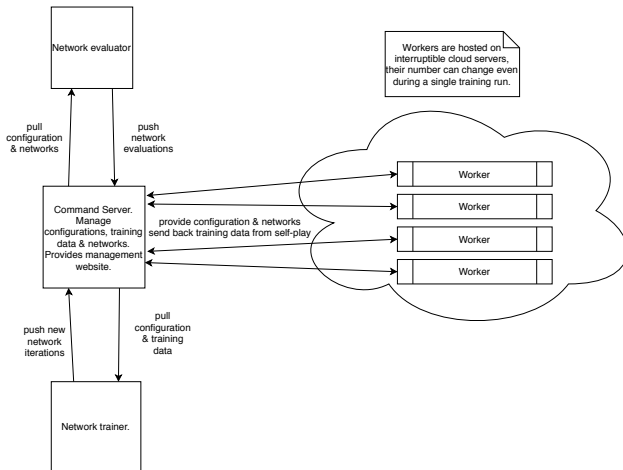


# Aufbau der Experimente

Grundlagen ▷ Baselines

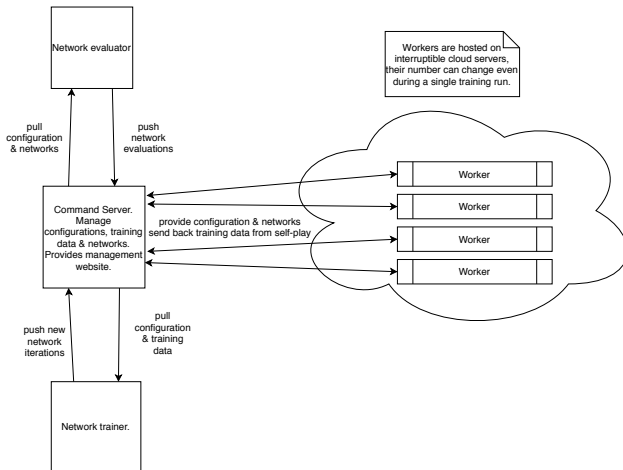
# Aufbau der Experimente

## Grundlagen ▸ Baselines



# Aufbau der Experimente

## Grundlagen ► Baselines



## ► Laufzeit auf Referenzmaschine

# Hyperparametersuche

Grundlagen ▷ Baselines



- ▶ Optimierte wichtige Hyperparameter für Vier gewinnt

# Hyperparametersuche

Grundlagen ▷ Baselines

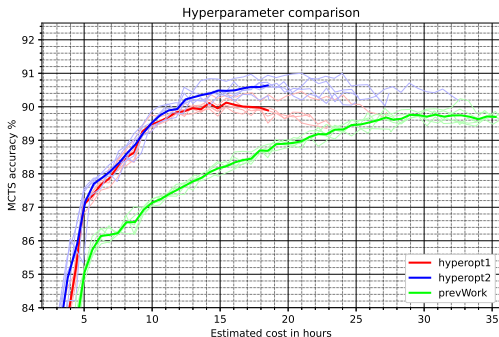
- ▶ Optimierte wichtige Hyperparameter für Vier gewinnt
- ▶ Bayesian Optimization Package verwendet

- ▶ Optimierte wichtige Hyperparameter für Vier gewinnt
- ▶ Bayesian Optimization Package verwendet
- ▶ Fitnessfunktion: Trainiere über 2 Stunden, messe Accuracy

- ▶ Optimierte wichtige Hyperparameter für Vier gewinnt
- ▶ Bayesian Optimization Package verwendet
- ▶ Fitnessfunktion: Trainiere über 2 Stunden, messe Accuracy
- ▶ 65 Steps, Laufzeit ca. eine Woche

# Hyperparametersuche

- ▶ Optimierte wichtige Hyperparameter für Vier gewinnt
- ▶ Bayesian Optimization Package verwendet
- ▶ Fitnessfunktion: Trainiere über 2 Stunden, messe Accuracy
- ▶ 65 Steps, Laufzeit ca. eine Woche



# Extended baseline

Grundlagen ▸ Baselines

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation



- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow
  - ▶ Playout Caps

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow
  - ▶ Playout Caps
  - ▶ Vorhersage des nächsten Zuges des Gegners

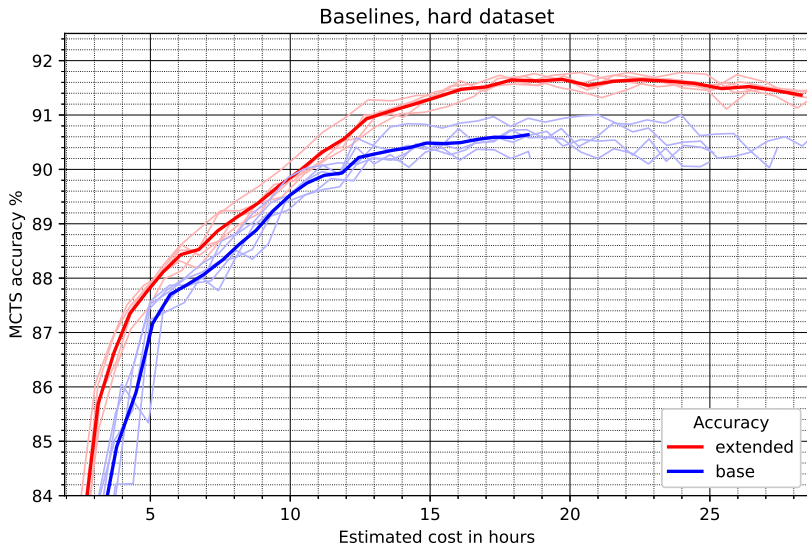
- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow
  - ▶ Playout Caps
  - ▶ Vorhersage des nächsten Zuges des Gegners
  - ▶ Verbesserung des Netzwerks

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow
  - ▶ Playout Caps
  - ▶ Vorhersage des nächsten Zuges des Gegners
  - ▶ Verbesserung des Netzwerks
- ▶ Alle außer Playout Caps zeigten eine tendenzielle Verbesserung

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - ▶ Deduplikation
  - ▶ Cyclic learning rate
  - ▶ Verbessertes Trainingswindow
  - ▶ Playout Caps
  - ▶ Vorhersage des nächsten Zuges des Gegners
  - ▶ Verbesserung des Netzwerks
- ▶ Alle außer Playout Caps zeigten eine tendenzielle Verbesserung
- ▶ Kombiniert ist die Verbesserung sehr erheblich.

# Extended baseline Ergebnisse

Grundlagen ▸ Baselines





### 1. Einleitung

### 2. Grundlagen

### 3. Untersuchte neue Ideen

- Evolutionary Self-play

- Games as trees

- Auxiliary features

- Fazit

### 4. Ende

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▸ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo
- ▶ Verwende Gaussian Mutation um die besten Spieler zu mutieren



# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo
- ▶ Verwende Gaussian Mutation um die besten Spieler zu mutieren
- ▶ Zwei Arten von Hyperparametern untersucht

# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo
- ▶ Verwende Gaussian Mutation um die besten Spieler zu mutieren
- ▶ Zwei Arten von Hyperparametern untersucht
  - ▶ Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.

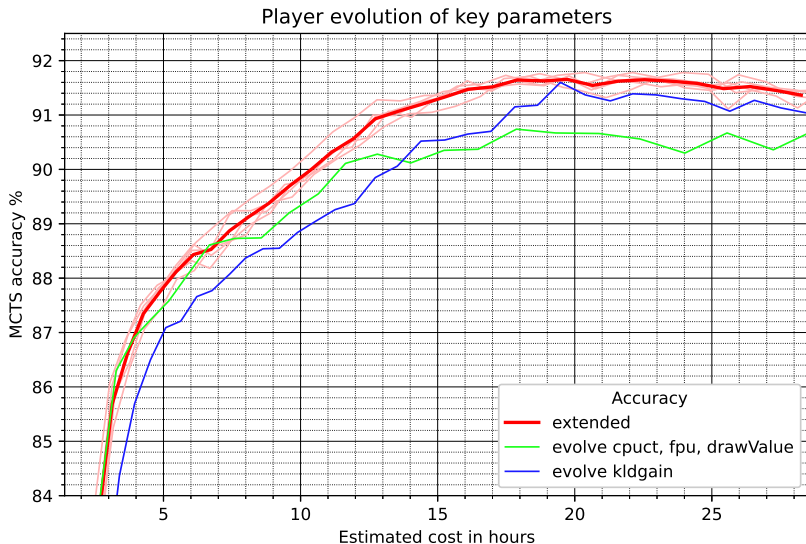
# Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Self-play-phase zur Evolution von Hyperparametern
- ▶ Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- ▶ Implementiert als eine Liga als Spielern.
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo
- ▶ Verwende Gaussian Mutation um die besten Spieler zu mutieren
- ▶ Zwei Arten von Hyperparametern untersucht
  - ▶ Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.
  - ▶ MCTS Parameter: cpuct, fpu, drawValue

# Erste Ergebnisse

Untersuchte neue Ideen ▷ Evolutionary Self-play





- ▶ Bedingungen für erfolgreiche Evolution:

- Bedingungen für erfolgreiche Evolution:
  - Die Liga muss gute Parameter erkennen

- ▶ Bedingungen für erfolgreiche Evolution:
  - ▶ Die Liga muss gute Parameter erkennen
  - ▶ Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt



# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.
  - ▶ Bei 0 hat der Parameter keinen Effekt

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.
  - ▶ Bei 0 hat der Parameter keinen Effekt
  - ▶ Bei 1 werden gute Züge selten gespielt, schlechte am häufigsten.

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.
  - ▶ Bei 0 hat der Parameter keinen Effekt
  - ▶ Bei 1 werden gute Züge selten gespielt, schlechte am häufigsten.
- ▶ Der Parameter wird innerhalb von 10 Generationen deutlich reduziert.



# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.
  - ▶ Bei 0 hat der Parameter keinen Effekt
  - ▶ Bei 1 werden gute Züge selten gespielt, schlechte am häufigsten.
- ▶ Der Parameter wird innerhalb von 10 Generationen deutlich reduziert.
- ▶ Ein voller Trainingslauf hat eher 30 bis 50 Generationen

# Erkennt die Liga gute Parameter?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ "Gut" im Sinne der Evolution: Gewinne viele Spiele
- ▶ Erfinde einen neuen Parameter für den der optimale Wert klar ist
- ▶ Stelle die Bewertung der Spielzüge auf den Kopf.
- ▶ Wertebereich von 0 bis 1.
  - ▶ Bei 0 hat der Parameter keinen Effekt
  - ▶ Bei 1 werden gute Züge selten gespielt, schlechte am häufigsten.
- ▶ Der Parameter wird innerhalb von 10 Generationen deutlich reduziert.
- ▶ Ein voller Trainingslauf hat eher 30 bis 50 Generationen
- ▶ Die Liga mit Evolution funktioniert ganz klar

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter
  - ▶ Bayesian Optimization



# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter
  - ▶ Bayesian Optimization
- ▶ Direkt optimierte Parameter sind am besten

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter
  - ▶ Bayesian Optimization
- ▶ Direkt optimierte Parameter sind am besten
- ▶ Die Evolution findet Parameter von ähnlicher Qualität wie die baseline

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter
  - ▶ Bayesian Optimization
- ▶ Direkt optimierte Parameter sind am besten
- ▶ Die Evolution findet Parameter von ähnlicher Qualität wie die baseline
- ▶ Aber erst am Ende des Trainings

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- ▶ Vergleiche 3 Hyperparametersets
  - ▶ Bester Spieler aus der Evolution
  - ▶ Baseline Parameter
  - ▶ Bayesian Optimization
- ▶ Direkt optimierte Parameter sind am besten
- ▶ Die Evolution findet Parameter von ähnlicher Qualität wie die baseline
- ▶ Aber erst am Ende des Trainings
  - ▶ Potentielles Problem: Das Training startet langsamer

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▷ Evolutionary Self-play

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt



# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- ▶ Nach hoher Siegesrate zu optimieren ist also nicht zielführend

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- ▶ Nach hoher Siegesrate zu optimieren ist also nicht zielführend
- ▶ Eine weitere untersuchte Alternative: Novelty search

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- ▶ Nach hoher Siegesrate zu optimieren ist also nicht zielführend
- ▶ Eine weitere untersuchte Alternative: Novelty search
  - ▶ Keine bedeutend höhere Diversität.

# Folgt Lernfortschritt aus mehr Siegen?

Untersuchte neue Ideen ▶ Evolutionary Self-play

- ▶ Vergleiche die 3 Optionen in 1000 Spiele Matches
- ▶ Klarer Sieger: Die Evolution
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- ▶ Nach hoher Siegesrate zu optimieren ist also nicht zielführend
- ▶ Eine weitere untersuchte Alternative: Novelty search
  - ▶ Keine bedeutend höhere Diversität.
  - ▶ Die Hyperparametersuche hat darauf schon implizit geachtet

# Implementierung: Games as trees

Untersuchte neue Ideen ▷ Games as trees

# Implementierung: Games as trees

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration durch Zurücksetzen an kritische Positionen

# Implementierung: Games as trees

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum

# Implementierung: Games as trees

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service



# Implementierung: Games as trees

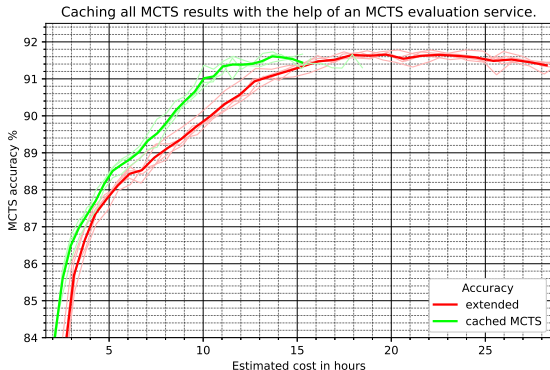
Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service
- ▶ Nebeneffekt: Keine doppelte Auswertung von Positionen

# Implementierung: Games as trees

Untersuchte neue Ideen ▶ Games as trees

- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service
- ▶ Nebeneffekt: Keine doppelte Auswertung von Positionen



# Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▷ Games as trees

# Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▶ Games as trees

- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen

# Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▷ Games as trees

- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation

# Zurücksetzen auf kritische Position

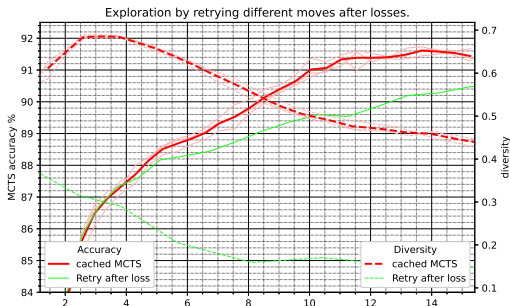
Untersuchte neue Ideen ▷ Games as trees

- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation
- ▶ Beginne neues Spiel in dieser Position

# Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▷ Games as trees

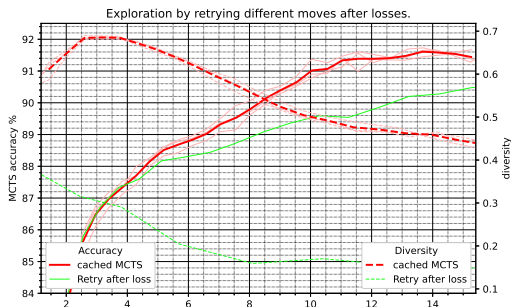
- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation
- ▶ Beginne neues Spiel in dieser Position



# Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▷ Games as trees

- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation
- ▶ Beginne neues Spiel in dieser Position



- ▶ Die Spieldiversität nimmt deutlich ab



# Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das

# Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum

# Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
  - ▶ 150k+ Knoten

# Spiele als MCTS-Baum

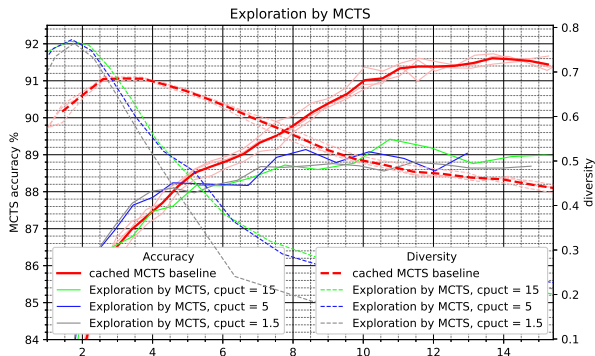
Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
  - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen

# Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
  - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen



# Implementierung: Auxiliary features

Untersuchte neue Ideen ▷ Auxiliary features

# Implementierung: Auxiliary features

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Trainiere ein kleines Netzwerk, ca. 70k Parameter



# Implementierung: Auxiliary features

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Trainiere ein kleines Netzwerk, ca. 70k Parameter
- ▶ Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes

# Implementierung: Auxiliary features

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Trainiere ein kleines Netzwerk, ca. 70k Parameter
- ▶ Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes
- ▶ Verschiedene Optionen wurden im Supervised Setting vorselektiert



- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke
- ▶ Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke
- ▶ Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger
- ▶ Kosten des Trainings des kleinen Netzwerkes sind zu hoch.



- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke
- ▶ Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger
- ▶ Kosten des Trainings des kleinen Netzwerkes sind zu hoch.
- ▶ Zwei Probleme mit dem Ansatz:

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke
- ▶ Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger
- ▶ Kosten des Trainings des kleinen Netzwerkes sind zu hoch.
- ▶ Zwei Probleme mit dem Ansatz:
  - ▶ Trainingskosten des kleinen Netzwerkes

- ▶ Kleine Gewinne im Supervised Setting, keine Gewinne im AlphaZero Setting
- ▶ Positiver Effekt auf initialen Anstieg der Spielstärke
- ▶ Negativer Effekt auf finale Spielstärke
- ▶ Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger
- ▶ Kosten des Trainings des kleinen Netzwerkes sind zu hoch.
- ▶ Zwei Probleme mit dem Ansatz:
  - ▶ Trainingskosten des kleinen Netzwerkes
  - ▶ Finale Spielstärke wird gestört

# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk

# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk
- ▶ Tausche des Netzwerk alle paar Iterationen gegen größeres aus

# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk
- ▶ Tausche des Netzwerk alle paar Iterationen gegen größeres aus
- ▶ Dieses Vorgehen ist generell effektiver

# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk
- ▶ Tausche des Netzwerk alle paar Iterationen gegen größeres aus
- ▶ Dieses Vorgehen ist generell effektiver
- ▶ Aber keine neue Idee



# Das Netzwerk wachsen lassen

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk
- ▶ Tausche des Netzwerk alle paar Iterationen gegen größeres aus
- ▶ Dieses Vorgehen ist generell effektiver
- ▶ Aber keine neue Idee
- ▶ Die Trainingskosten des kleinen Netzwerkes verschwinden

# Finale Spielstärke darf nicht geschadet werden

Untersuchte neue Ideen ▷ Auxiliary features

# Finale Spielstärke darf nicht geschadet werden

Untersuchte neue Ideen ▶ Auxiliary features

- ▶ Auxiliary features dürfen nicht der finalen Spielstärke schaden

# Finale Spielstärke darf nicht geschadet werden

Untersuchte neue Ideen ▶ Auxiliary features

- ▶ Auxiliary features dürfen nicht der finalen Spielstärke schaden
- ▶ Es wurden verschiedene Optionen erprobt

# Finale Spielstärke darf nicht geschadet werden

Untersuchte neue Ideen ▶ Auxiliary features

- ▶ Auxiliary features dürfen nicht der finalen Spielstärke schaden
- ▶ Es wurden verschiedene Optionen erprobt
- ▶ Keine verbesserte die Situation



- AlphaZero Experimentalframework entwickelt

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden



- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert
    - ▶ Nur eine gute Fitnessfunktion fehlt

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert
    - ▶ Nur eine gute Fitnessfunktion fehlt
    - ▶ Unterschied zwischen Lernfortschritt und Spielstärke

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert
    - ▶ Nur eine gute Fitnessfunktion fehlt
    - ▶ Unterschied zwischen Lernfortschritt und Spielstärke
  - ▶ Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert
    - ▶ Nur eine gute Fitnessfunktion fehlt
    - ▶ Unterschied zwischen Lernfortschritt und Spielstärke
  - ▶ Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - ▶ Auxiliary features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar

- ▶ AlphaZero Experimentalframework entwickelt
- ▶ Keine großen Verbesserungen gefunden
- ▶ Trotzdem einiges Interessante Erkenntnisse
  - ▶ Nicht alle Ideen vorheriger Arbeiten funktionieren mit Vier gewinnt
  - ▶ Evolution für Hyperparameter funktioniert
    - ▶ Nur eine gute Fitnessfunktion fehlt
    - ▶ Unterschied zwischen Lernfortschritt und Spielstärke
  - ▶ Alternative Explorationsmethoden zeigen vor allem wie gut die einfache Standardversion funktioniert
  - ▶ Auxiliary features aus dem inneren eines kleineren Netzwerks sind nur schwer nutzbar
    - ▶ Aber das Konzept des Netzwerkwachstums sollte weiter erforscht werden



1. Einleitung
2. Grundlagen
3. Untersuchte neue Ideen
4. Ende

Danke für Ihre Aufmerksamkeit.

- ▶ Maybe do some references here...