

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

Colin Clausen

3.9.2020

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

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
- ▶ Benötigt:

Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- ▶ 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
- ▶ Benötigt:
 - ▶ Eine Policy die Züge einschätzen kann
 - ▶ Lösung um die Suchtiefe zu begrenzen

Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- ▶ Benötigt:
 - ▶ Eine Policy die Züge einschätzen kann
 - ▶ Lösung um die Suchtiefe zu begrenzen
 - ▶ Eine sehr schnelle Rolloutpolicy

Monte Carlo tree search (MCTS)

- ▶ Seit 2006 sehr verbreitet in Computer Go
- ▶ Iterativer Baumsuchalgorithmus verwendet in AlphaGo/AlphaGoZero/AlphaZero
- ▶ Benötigt:
 - ▶ Eine Policy die Züge einschätzen kann
 - ▶ Lösung um die Suchtiefe zu begrenzen
 - ▶ 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
- ▶ Benötigt:
 - ▶ Eine Policy die Züge einschätzen kann
 - ▶ Lösung um die Suchtiefe zu begrenzen
 - ▶ 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
- ▶ Benötigt:
 - ▶ Eine Policy die Züge einschätzen kann
 - ▶ Lösung um die Suchtiefe zu begrenzen
 - ▶ Eine sehr schnelle Rolloutpolicy
 - ▶ Alternativ: Policy zur Positionseinschätzung
- ▶ Output: Eine bessere Policy über die Züge in der analysierten Position
- ▶ MCTS ist praktisch ein Verbesserungsoperator

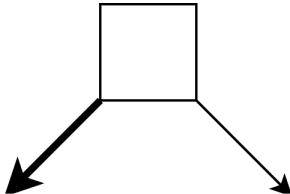
MCTS Beispiel

Grundlagen ▷ Algorithmus



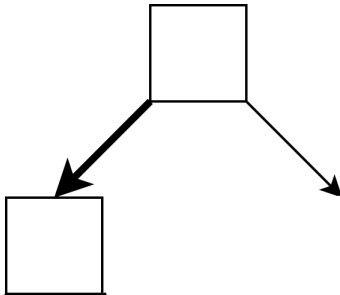
MCTS Beispiel

Grundlagen ▷ Algorithmus



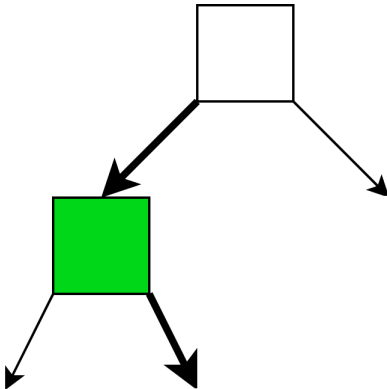
MCTS Beispiel

Grundlagen ▷ Algorithmus



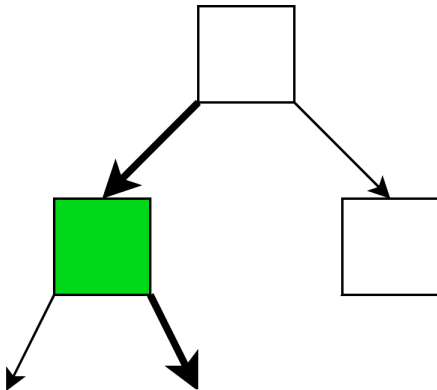
MCTS Beispiel

Grundlagen ▷ Algorithmus



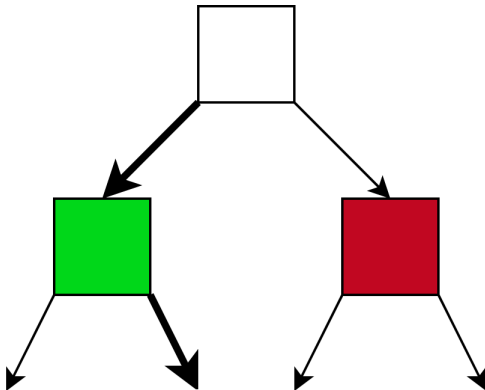
MCTS Beispiel

Grundlagen ▷ Algorithmus



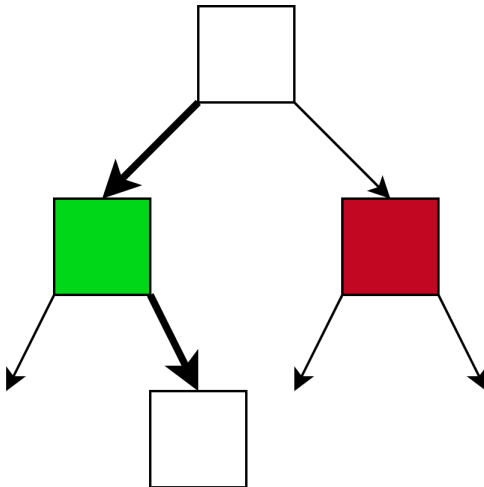
MCTS Beispiel

Grundlagen ▸ Algorithmus



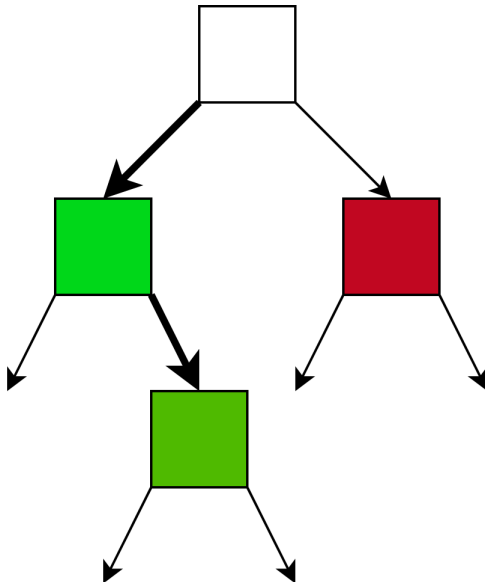
MCTS Beispiel

Grundlagen ▷ Algorithmus



MCTS Beispiel

Grundlagen ▷ Algorithmus



- ▶ 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
- ▶ Trainiere 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

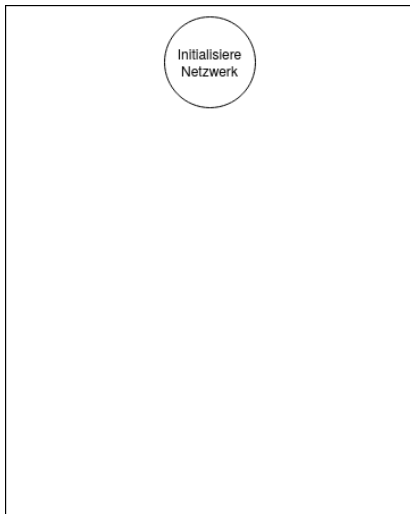
- ▶ Drastische Vereinfachung von AlphaGo
- ▶ Kernidee: Verwende MCTS bereits zur Trainingsphase

- ▶ 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

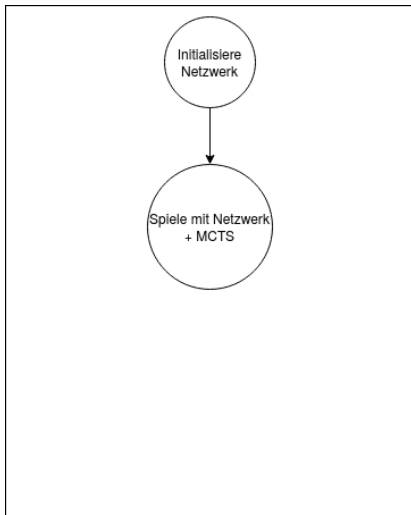
AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



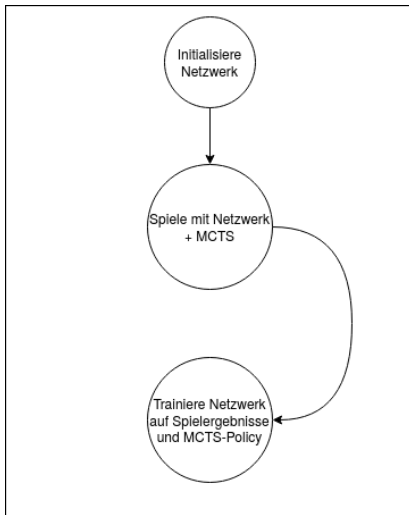
AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



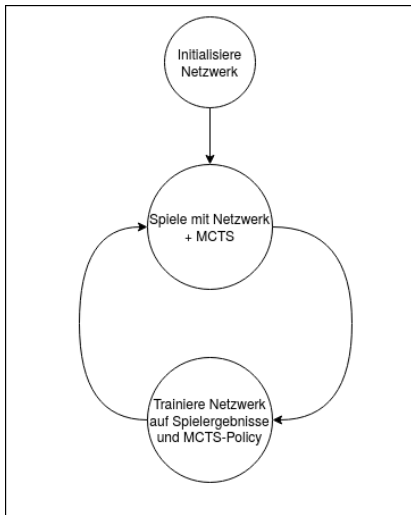
AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



AlphaZero: Trainingsablauf

Grundlagen ▷ Algorithmus



Extended baseline

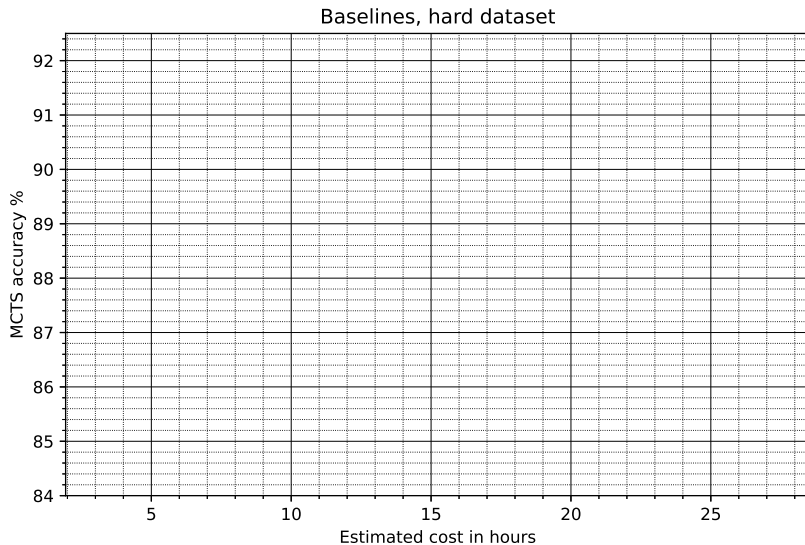
Grundlagen ▸ Baselines

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten

- ▶ Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
- ▶ Kombiniert ist die Verbesserung sehr erheblich

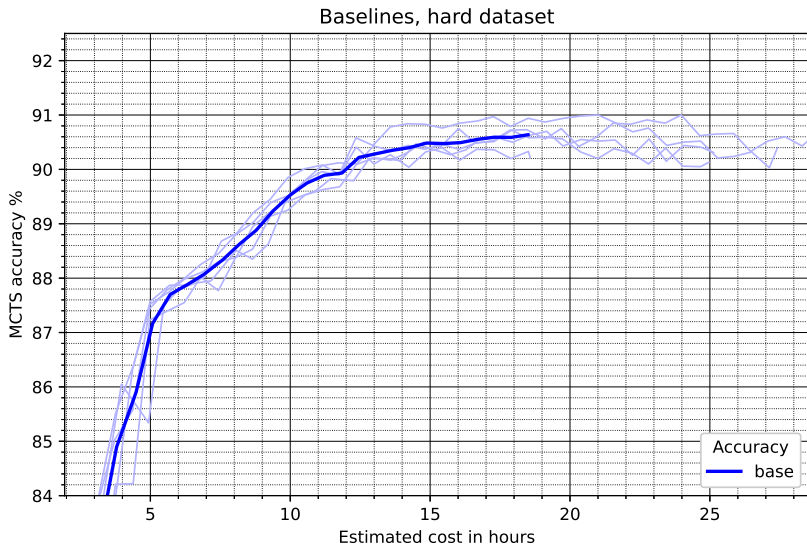
Extended baseline Ergebnisse

Grundlagen ▷ Baselines



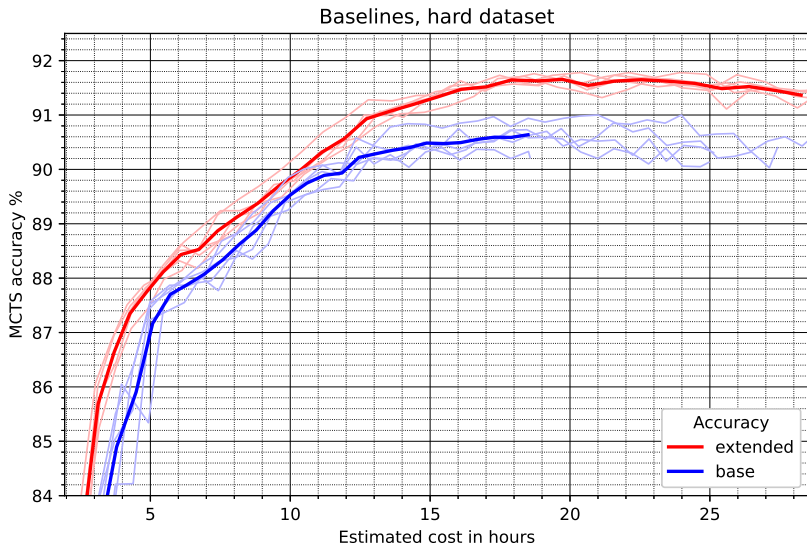
Extended baseline Ergebnisse

Grundlagen ▸ Baselines



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 Selbstspielphase zur Evolution von Hyperparametern

Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von Spielern

Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von Spielern
- ▶ Ein Spieler ist ein Hyperparameterset

Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von Spielern
- ▶ Ein Spieler ist ein Hyperparameterset
- ▶ Bewerte Spieler mit Elo

Implementierung: Evolutionary Self-play

Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von 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 Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von 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 Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von 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

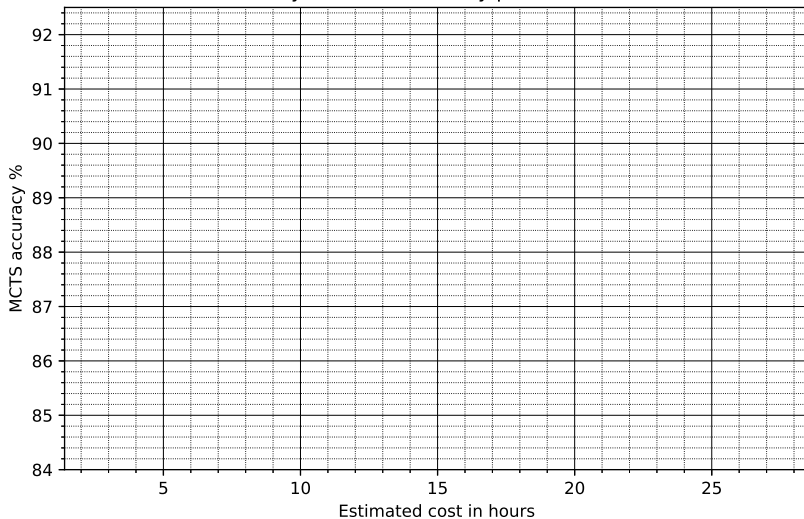
Untersuchte neue Ideen ▷ Evolutionary Self-play

- ▶ Verwende die Selbstspielphase zur Evolution von Hyperparametern
- ▶ Implementiert als eine Liga von 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

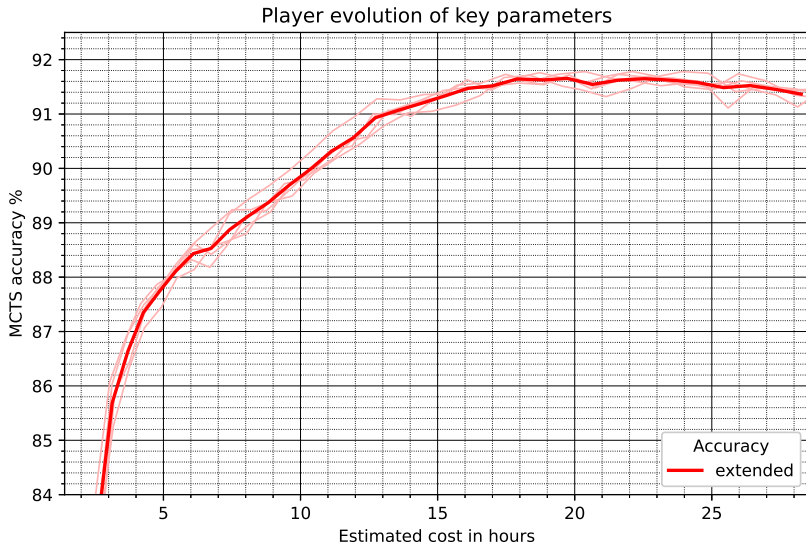
Untersuchte neue Ideen ▷ Evolutionary Self-play

Player evolution of key parameters



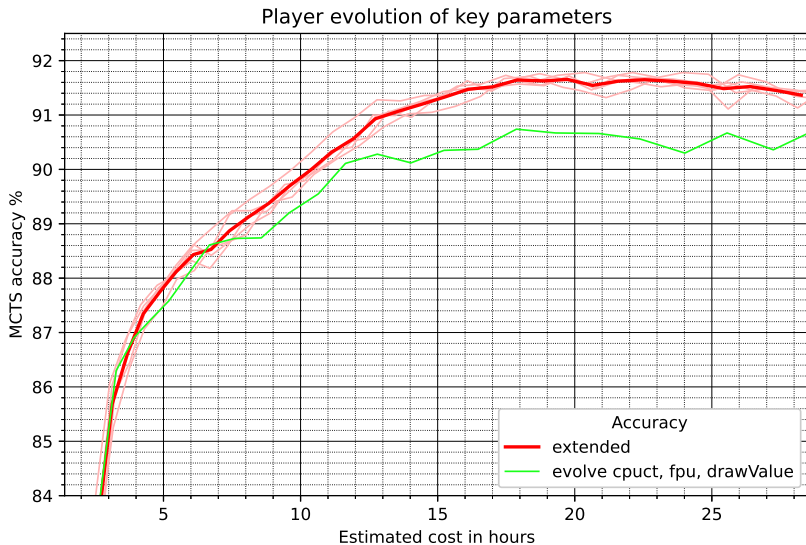
Erste Ergebnisse

Untersuchte neue Ideen ▷ Evolutionary Self-play



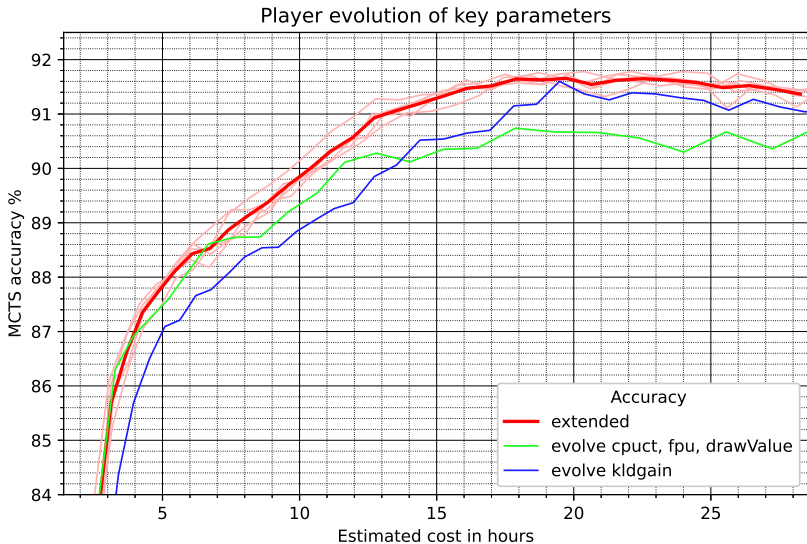
Erste Ergebnisse

Untersuchte neue Ideen ▷ Evolutionary Self-play



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
 - ▶ Sie funktioniert

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

- ▶ Bedingungen für erfolgreiche Evolution:
 - ▶ Die Liga muss gute Parameter erkennen
 - ▶ Sie funktioniert
 - ▶ Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
 - ▶ Dies ist das Problem

- ▶ Bedingungen für erfolgreiche Evolution:
 - ▶ Die Liga muss gute Parameter erkennen
 - ▶ Sie funktioniert
 - ▶ Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
 - ▶ Dies ist das Problem
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt

- ▶ Bedingungen für erfolgreiche Evolution:
 - ▶ Die Liga muss gute Parameter erkennen
 - ▶ Sie funktioniert
 - ▶ Viele Siege müssen sich übertragen auf schnelleren Lernfortschritt
 - ▶ Dies ist das Problem
- ▶ Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt
- ▶ Nach hoher Siegesrate zu optimieren ist also nicht zielführend

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

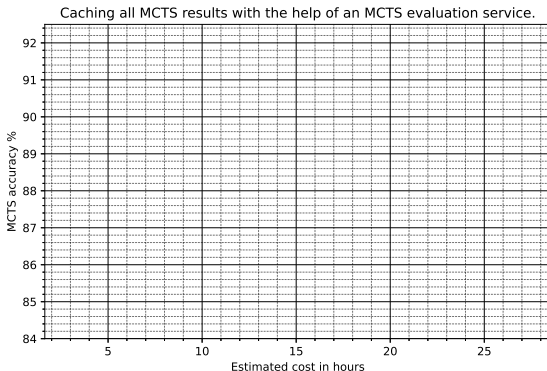
- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service

- ▶ 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

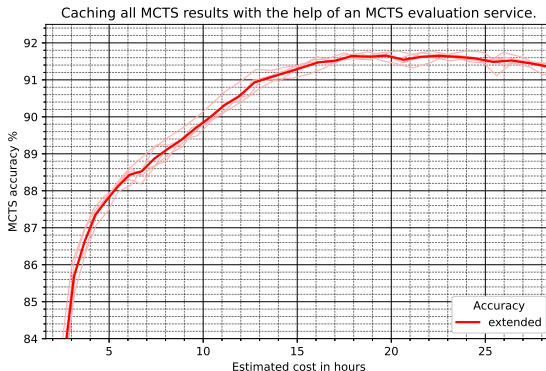
- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service
- ▶ Nebeneffekt: Keine doppelte Auswertung von Positionen



Implementierung: Games as trees

Untersuchte neue Ideen ▶ Games as trees

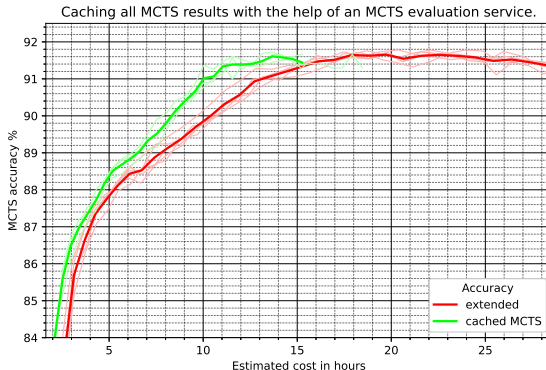
- ▶ Exploration durch Zurücksetzen an kritische Positionen
- ▶ Spiele als MCTS-Baum
- ▶ Notwendigkeit für MCTS-Evaluation Service
- ▶ Nebeneffekt: Keine doppelte Auswertung von Positionen



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

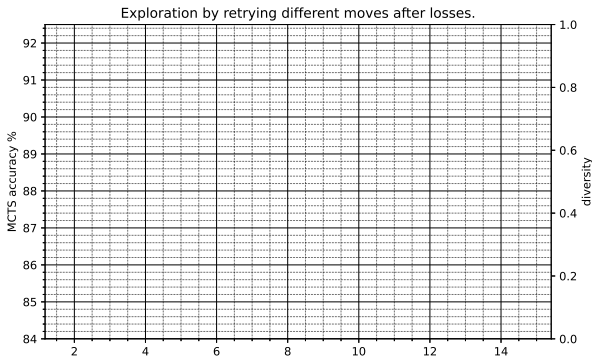
Untersuchte neue Ideen ▶ Games as trees

- ▶ 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

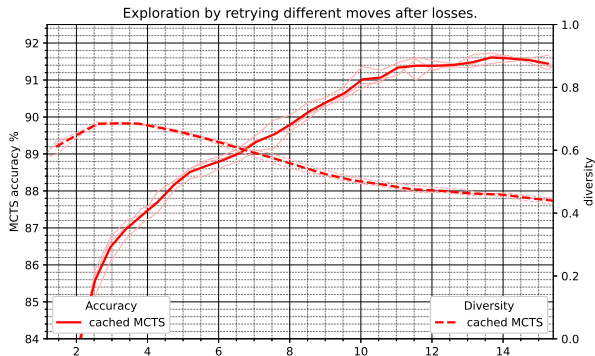
- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation
- ▶ Beginne neues Spiel in dieser Position



Zurücksetzen auf kritische Position

Untersuchte neue Ideen ▷ Games as trees

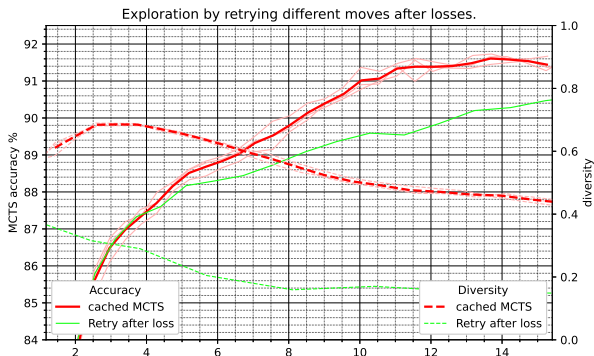
- ▶ Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- ▶ Wähle Position anhand der Entwicklung der Positionsevaluation
- ▶ Beginne neues Spiel in dieser Position



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



Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

- ▶ Exploration-Exploitation: MCTS macht das

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

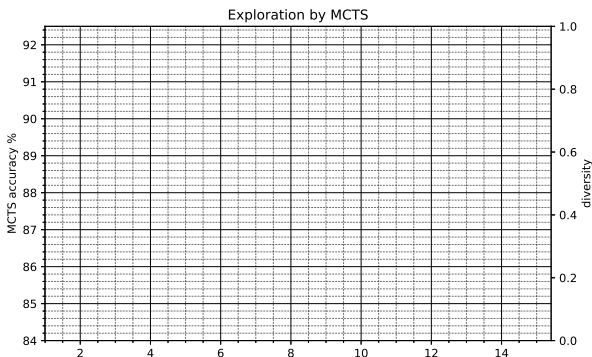
- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
 - ▶ 150k+ Knoten

- ▶ 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

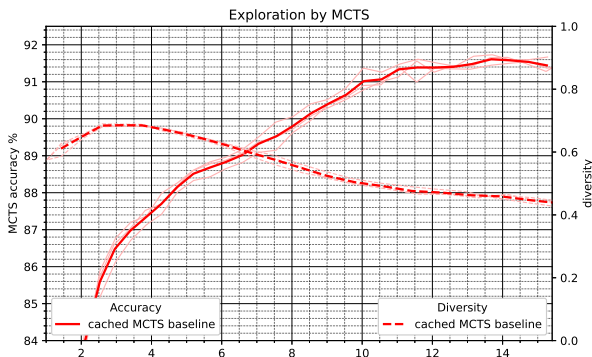
- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
 - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen



Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

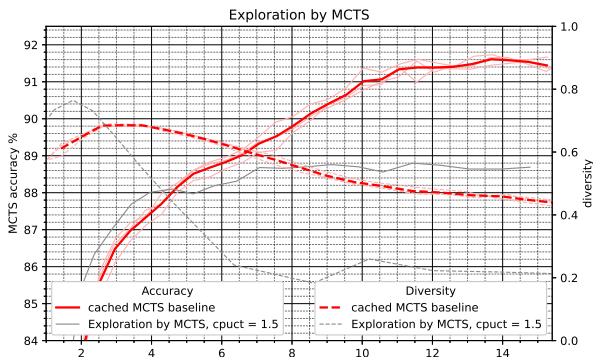
- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
 - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen



Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

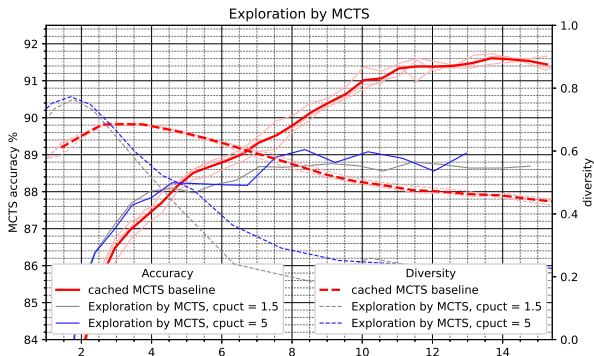
- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
 - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen



Spiele als MCTS-Baum

Untersuchte neue Ideen ▷ Games as trees

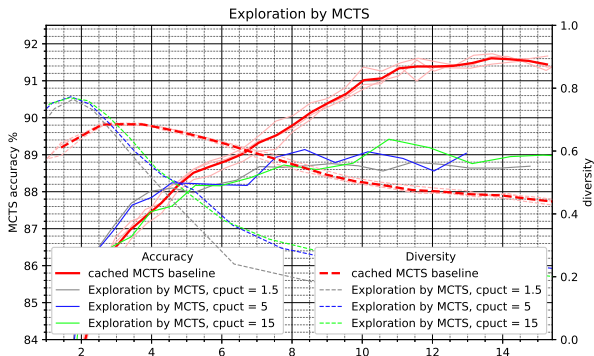
- ▶ Exploration-Exploitation: MCTS macht das
- ▶ Baue einen einzigen MCTS Baum
 - ▶ 150k+ Knoten
- ▶ Reporte die Positionen in den Knoten als Trainingspositionen



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

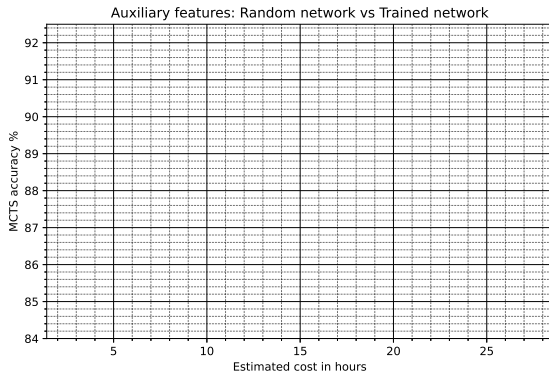
Untersuchte neue Ideen ▶ Auxiliary features

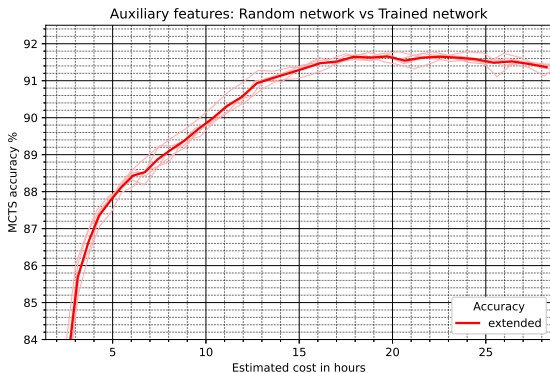
- ▶ Trainiere ein kleines Netzwerk, ca. 70k Parameter
- ▶ Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes
- ▶ Verschiedene Optionen wurden im Supervised Setting vorselektiert

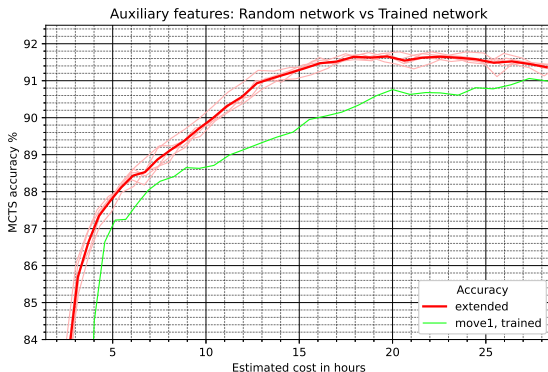
Implementierung: Auxiliary features

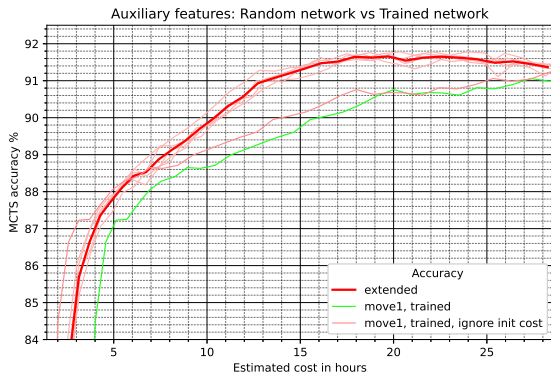
Untersuchte neue Ideen ▷ Auxiliary features

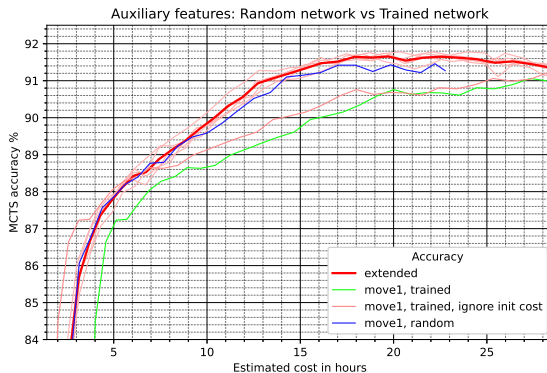
- ▶ Trainiere ein kleines Netzwerk, ca. 70k Parameter
- ▶ Verwende interne Features aus diesem Netzwerk zur Regularisierung des großen Netzwerkes
- ▶ Verschiedene Optionen wurden im Supervised Setting vorselektiert
- ▶ Kleine Gewinne im Supervised Setting











Probleme

Untersuchte neue Ideen ▷ Auxiliary features

- ▶ Zwei Probleme mit dem Ansatz:

- ▶ Zwei Probleme mit dem Ansatz:
 - ▶ Trainingskosten des kleinen Netzwerks

- ▶ Zwei Probleme mit dem Ansatz:
 - ▶ Trainingskosten des kleinen Netzwerks
 - ▶ Das Netzwerk mit dem Trainingslauf wachsen lassen hilft

- ▶ Zwei Probleme mit dem Ansatz:
 - ▶ Trainingskosten des kleinen Netzwerks
 - ▶ Das Netzwerk mit dem Trainingslauf wachsen lassen hilft
 - ▶ Finale Spielstärke wird gestört

- ▶ Zwei Probleme mit dem Ansatz:
 - ▶ Trainingskosten des kleinen Netzwerks
 - ▶ Das Netzwerk mit dem Trainingslauf wachsen lassen hilft
 - ▶ Finale Spielstärke wird gestört
 - ▶ Keine Lösung gefunden

- 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
- ▶ Vorschläge für weitere Forschung

- ▶ 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
- ▶ Vorschläge für weitere Forschung
 - ▶ Suche nach Fitnessfunktion für Evolution

- ▶ 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
- ▶ Vorschläge für weitere Forschung
 - ▶ Suche nach Fitnessfunktion für Evolution
 - ▶ Das Konzept des Netzwerkwachstums sollte weiter erforscht werden

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

- ▶ Levente Kocsis et al.: Bandit based monte-carlo planning, 2006.
- ▶ David Silver et al.: Mastering the game of go with deep neural networks and tree search, 2016
- ▶ David Silver et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play, 2018

Danke für Ihre Aufmerksamkeit.