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

Colin Clausen

3.9.2020

Inhalt

Inhalt

Einleitung

Einleitung

- Spiele als Lernumgebung für AI

Einleitung

- Spiele als Lernumgebung für AI
  - 1951: Nimrod

Einleitung

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

Einleitung

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

Einleitung

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

Einleitung

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

Einleitung

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

Einleitung

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

Zielsetzung

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen

Zielsetzung

- Untersuche mögliche Effizienzsteigerungen
- Solide Baseline

Zielsetzung

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

Zielsetzung

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

Zielsetzung

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

Zielsetzung

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

Zielsetzung

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

Inhalt

Monte Carlo tree search (MCTS)

Monte Carlo tree search (MCTS)

- Seit 2006 sehr verbreitet in Computer Go.

Monte Carlo tree search (MCTS)

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

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

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

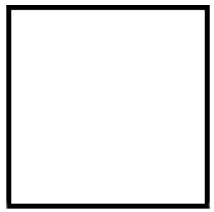
Monte Carlo tree search (MCTS)

- 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

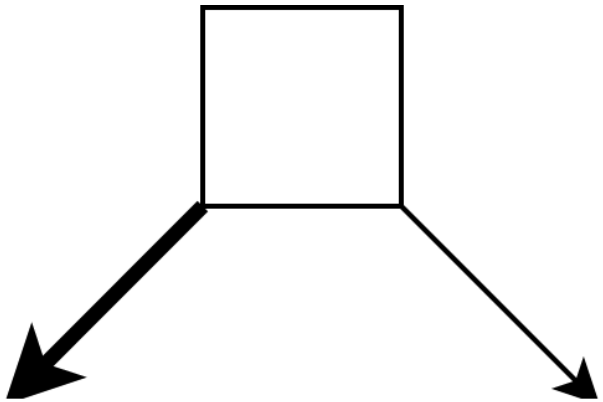
Monte Carlo tree search (MCTS)

- 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

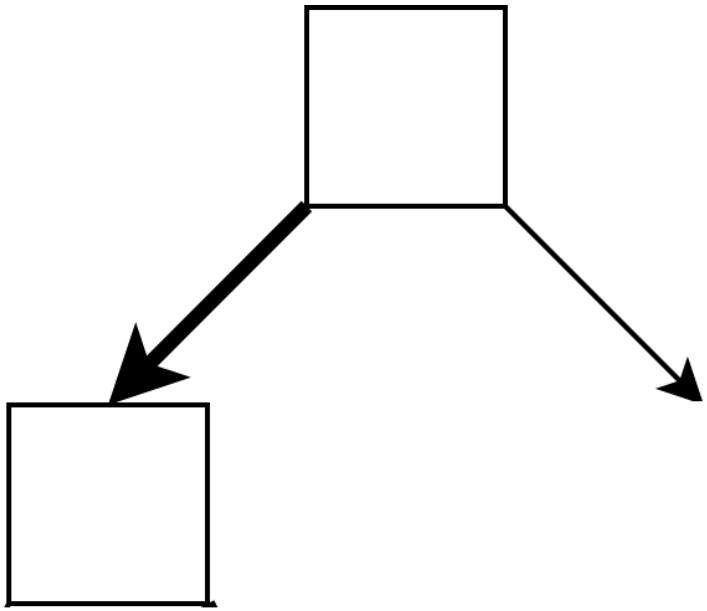
MCTS Beispiel



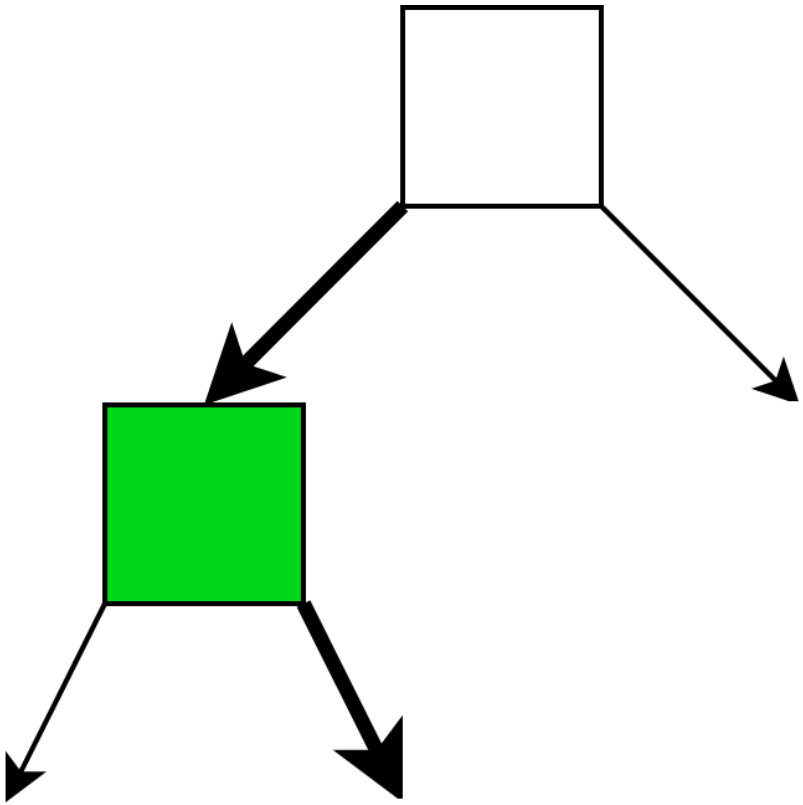
MCTS Beispiel



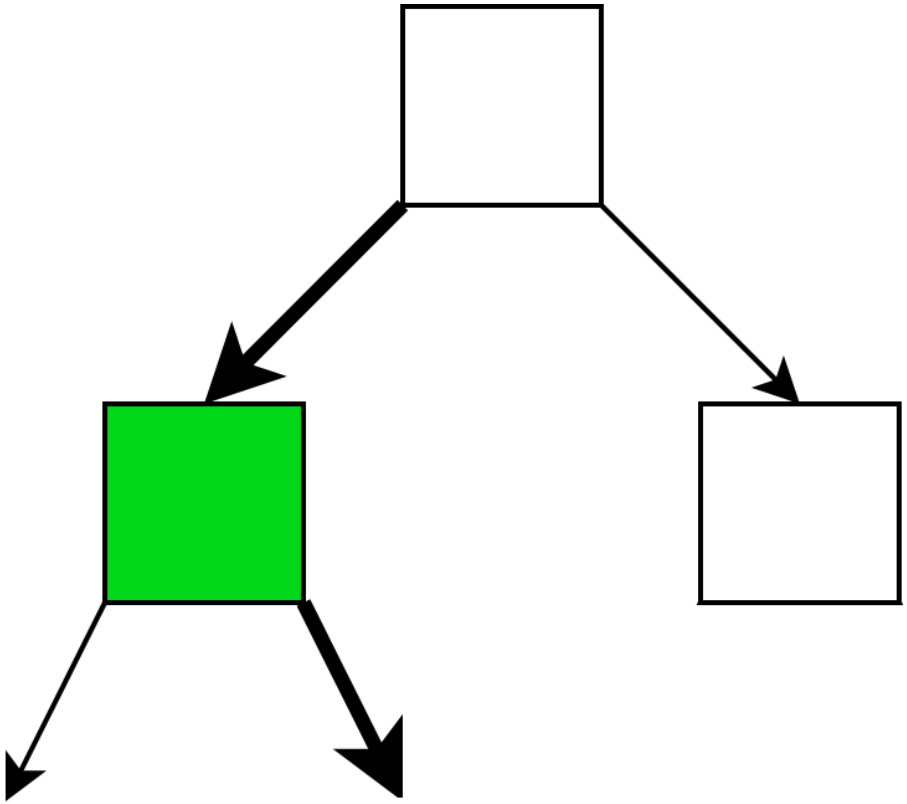
MCTS Beispiel



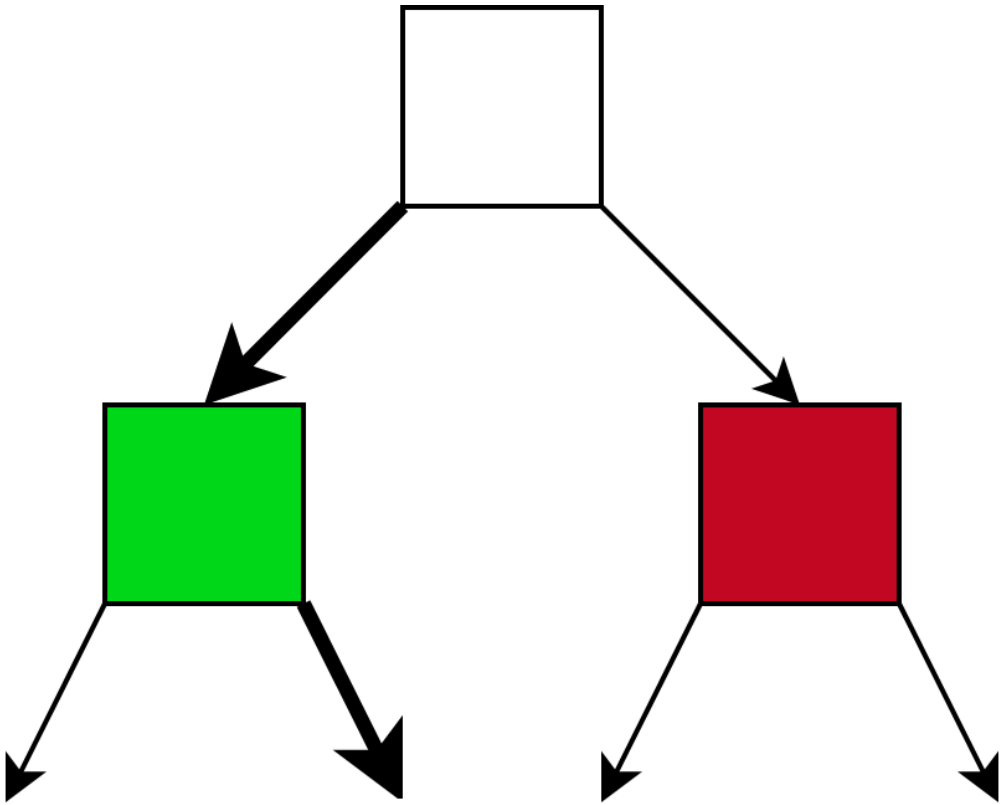
MCTS Beispiel



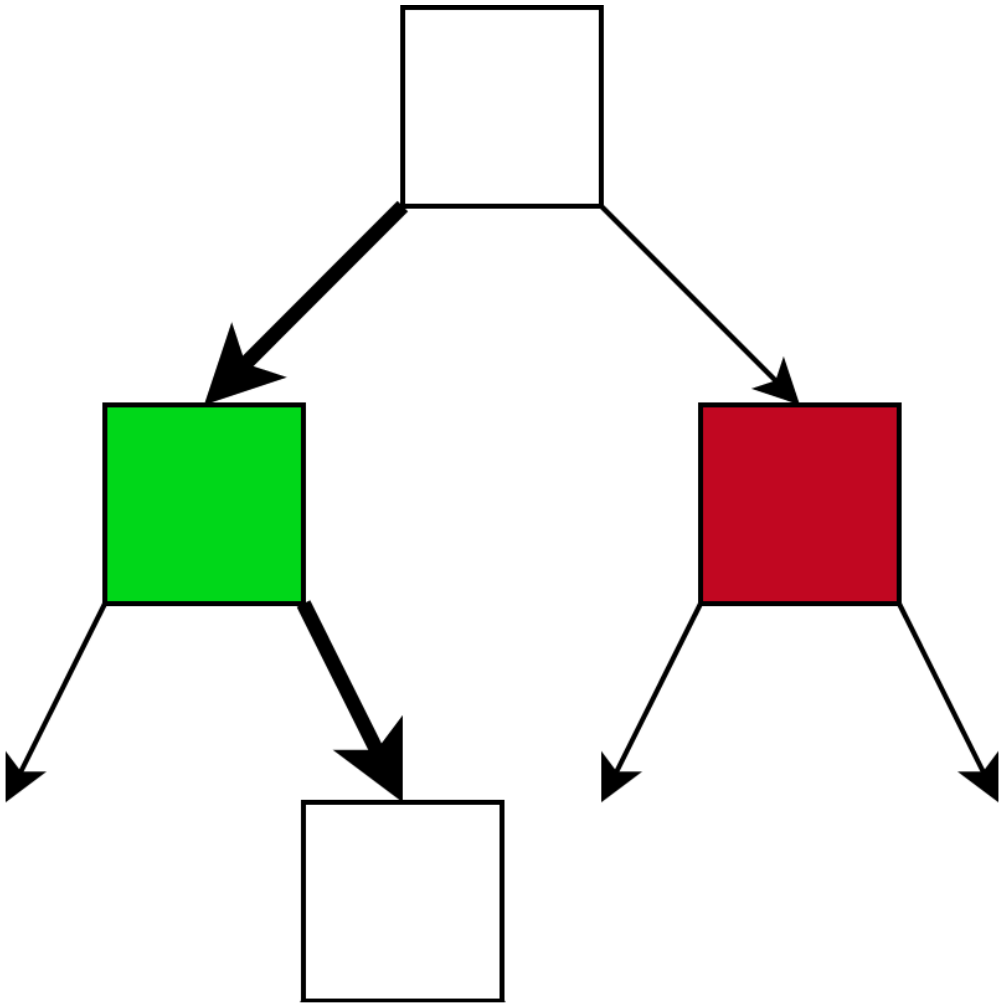
MCTS Beispiel



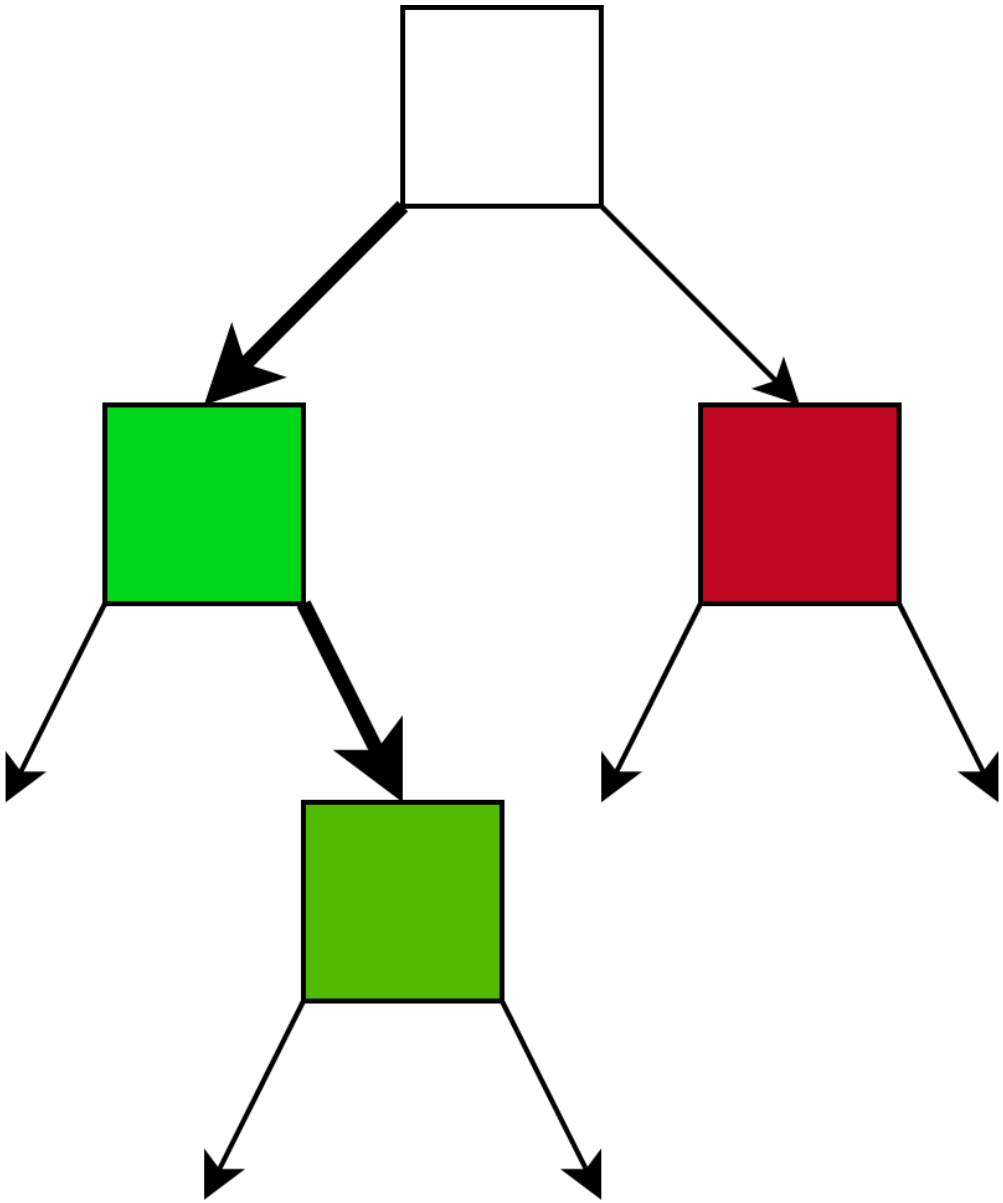
MCTS Beispiel



MCTS Beispiel



MCTS Beispiel



AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning

AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere mehrere Netzwerke

AlphaGo

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

AlphaGo

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

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

AlphaGo

- Kernidee: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere 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: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere 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: Kombinere MCTS mit Deep Learning
- Trainingsprozess involviert aber kein MCTS
- Trainere 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

AlphaZero

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

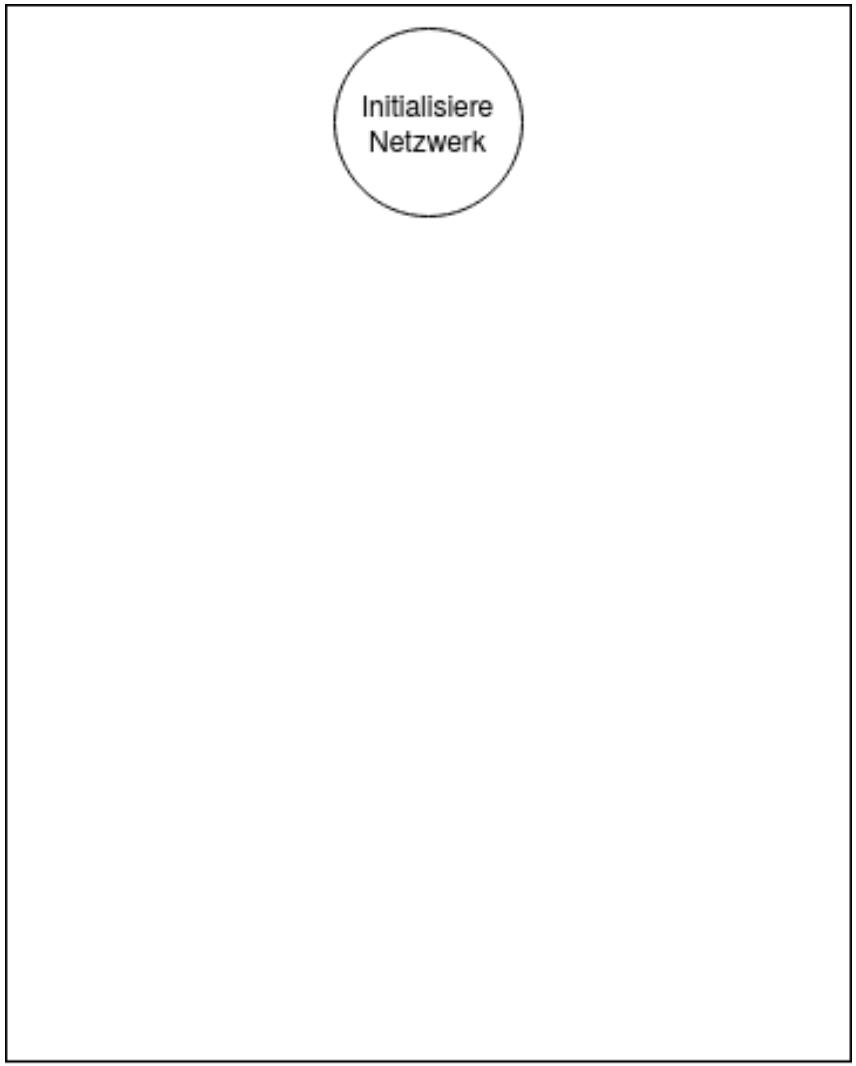
AlphaZero

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

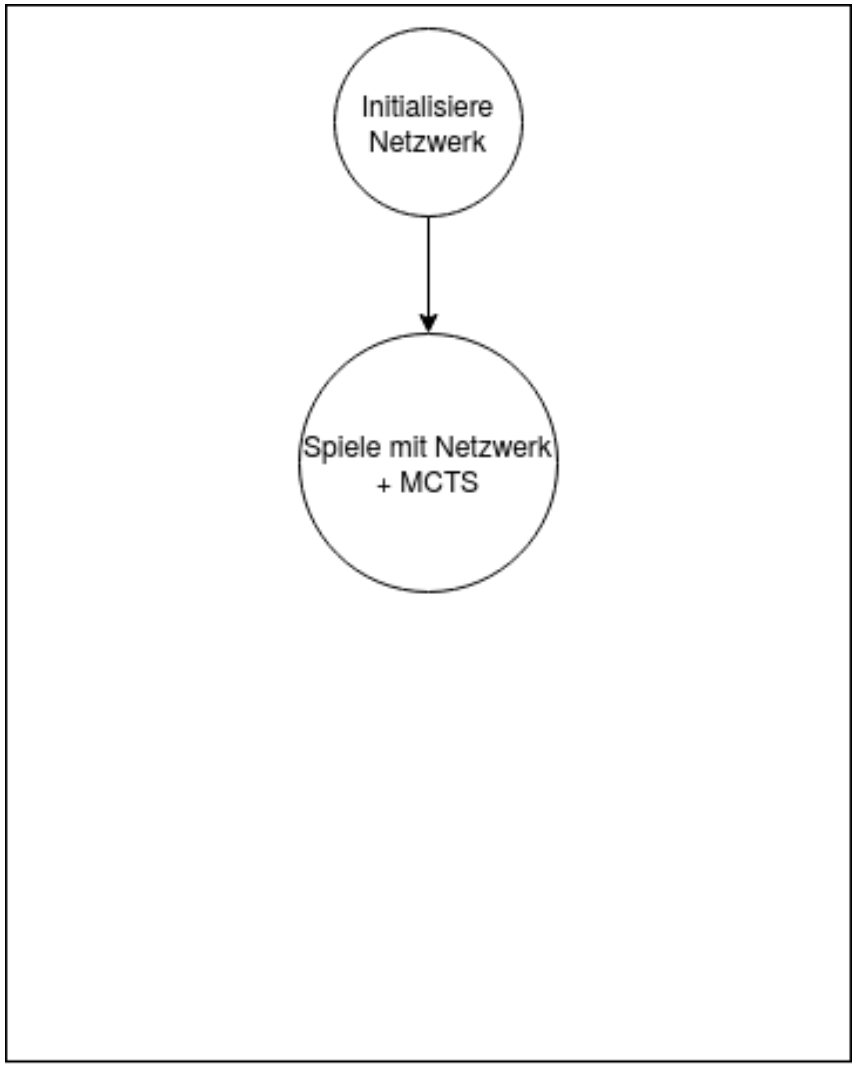
AlphaZero

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

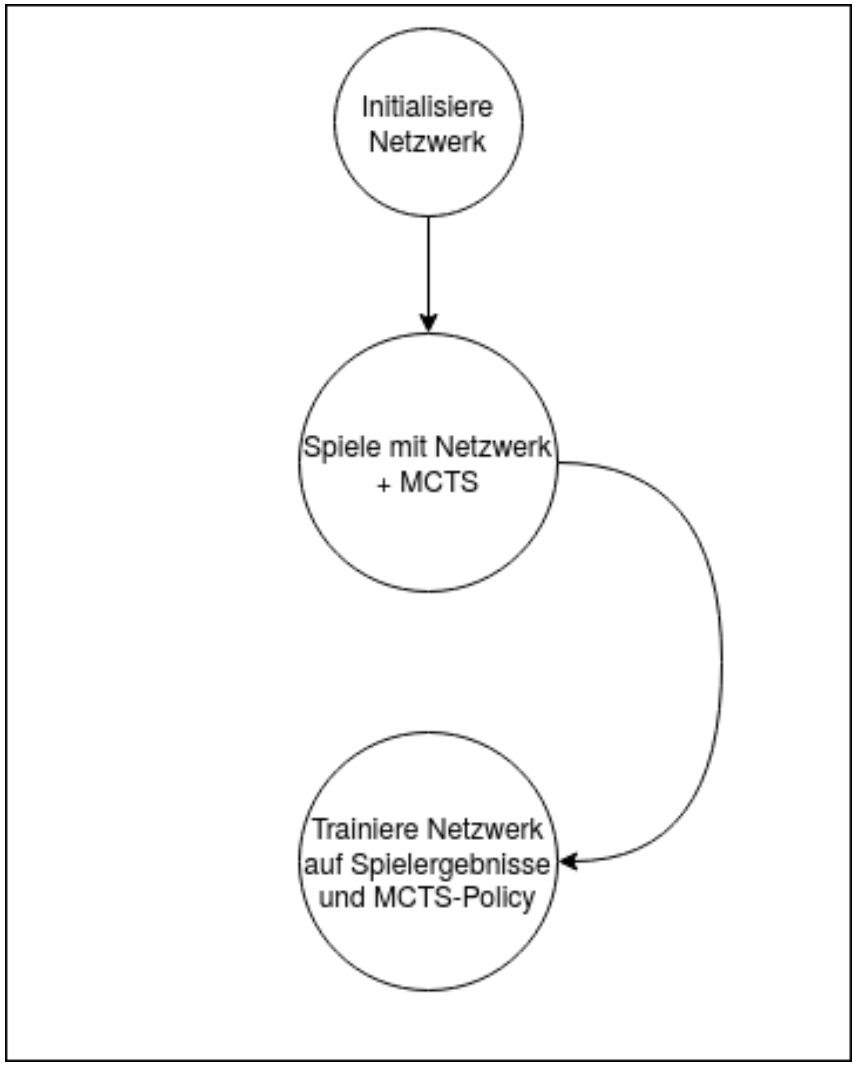
AlphaZero: Trainingsablauf



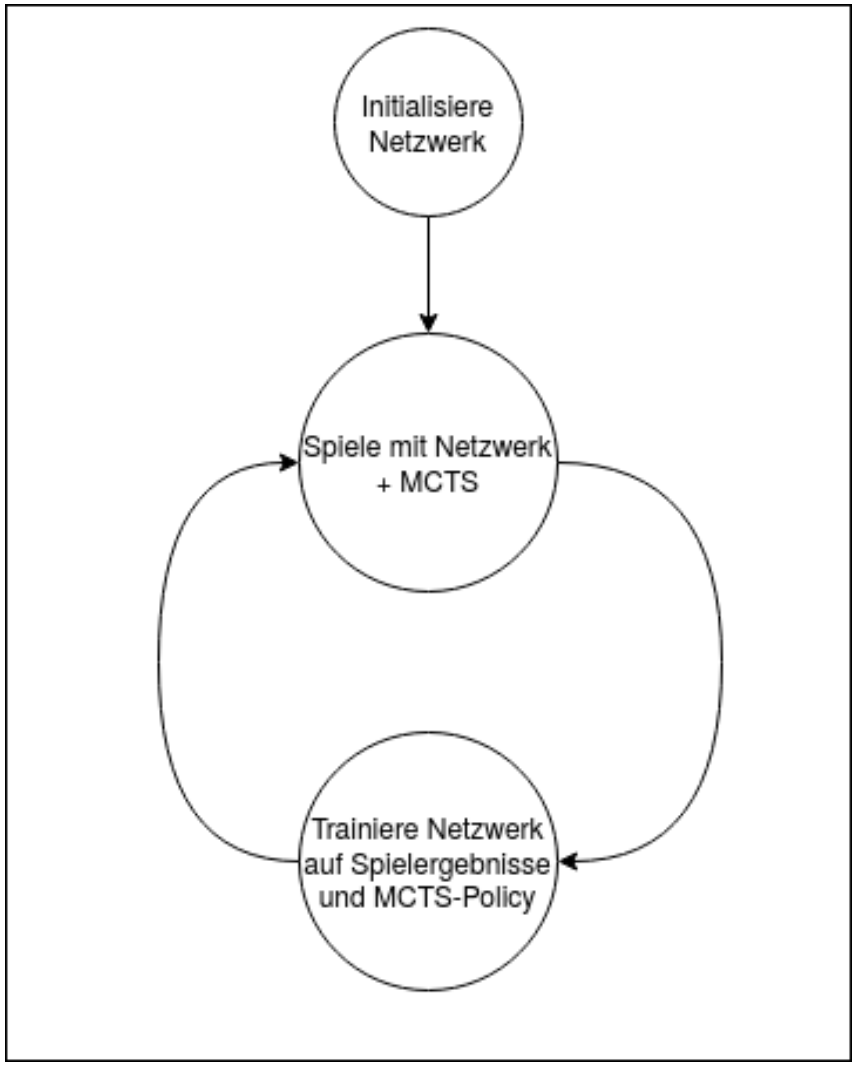
AlphaZero: Trainingsablauf



AlphaZero: Trainingsablauf

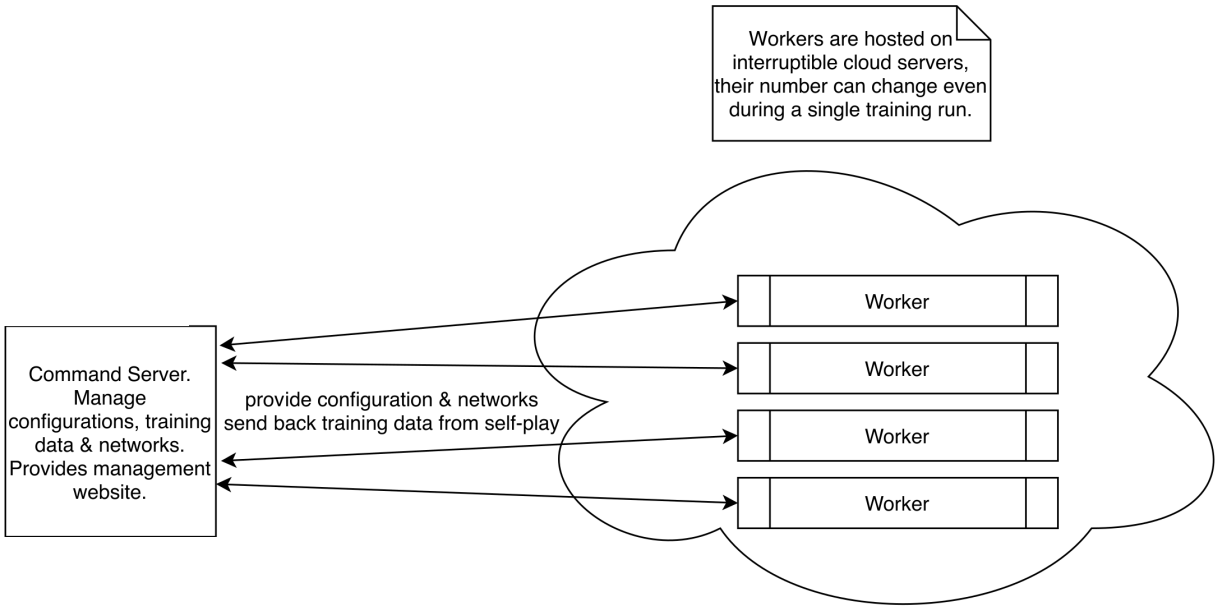


AlphaZero: Trainingsablauf

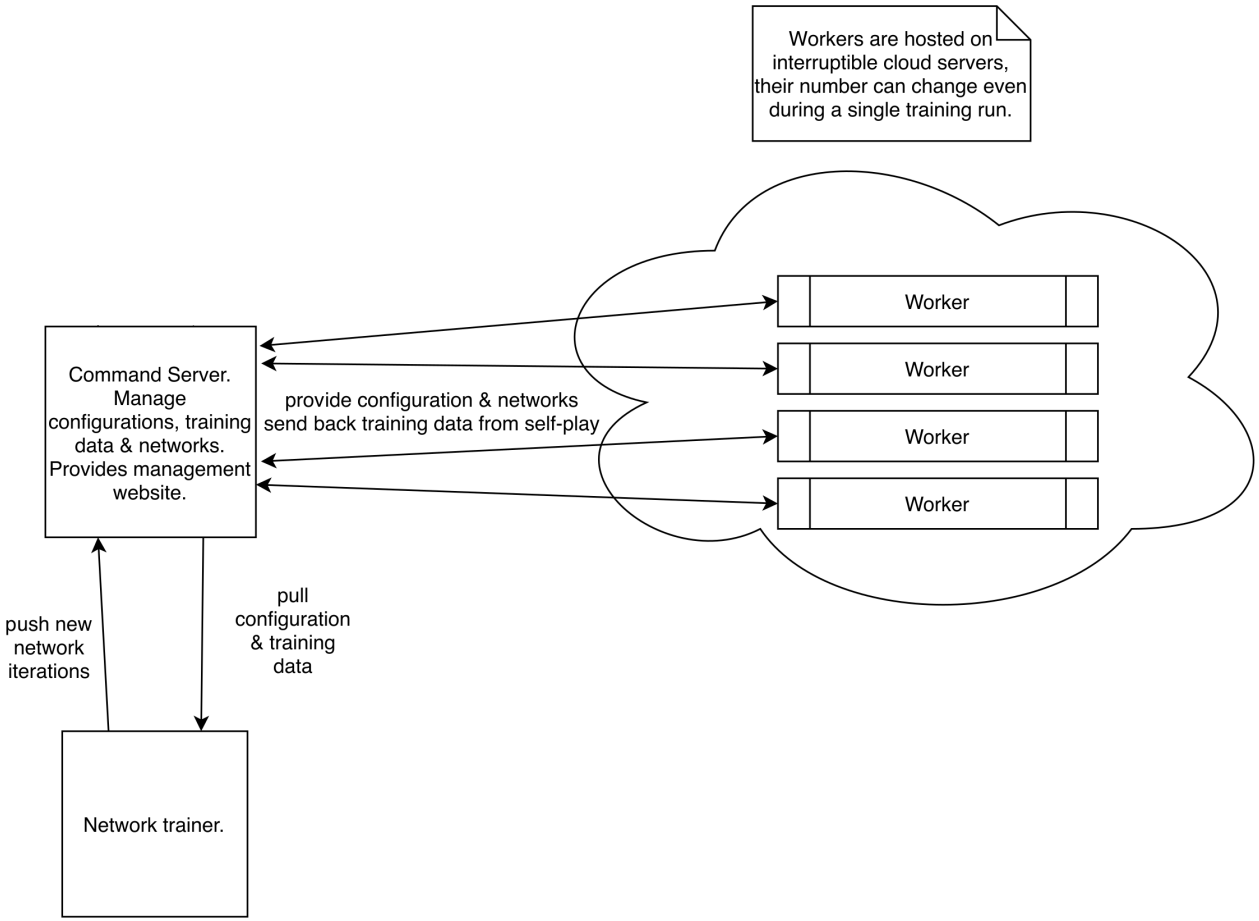


Aufbau der Experimente

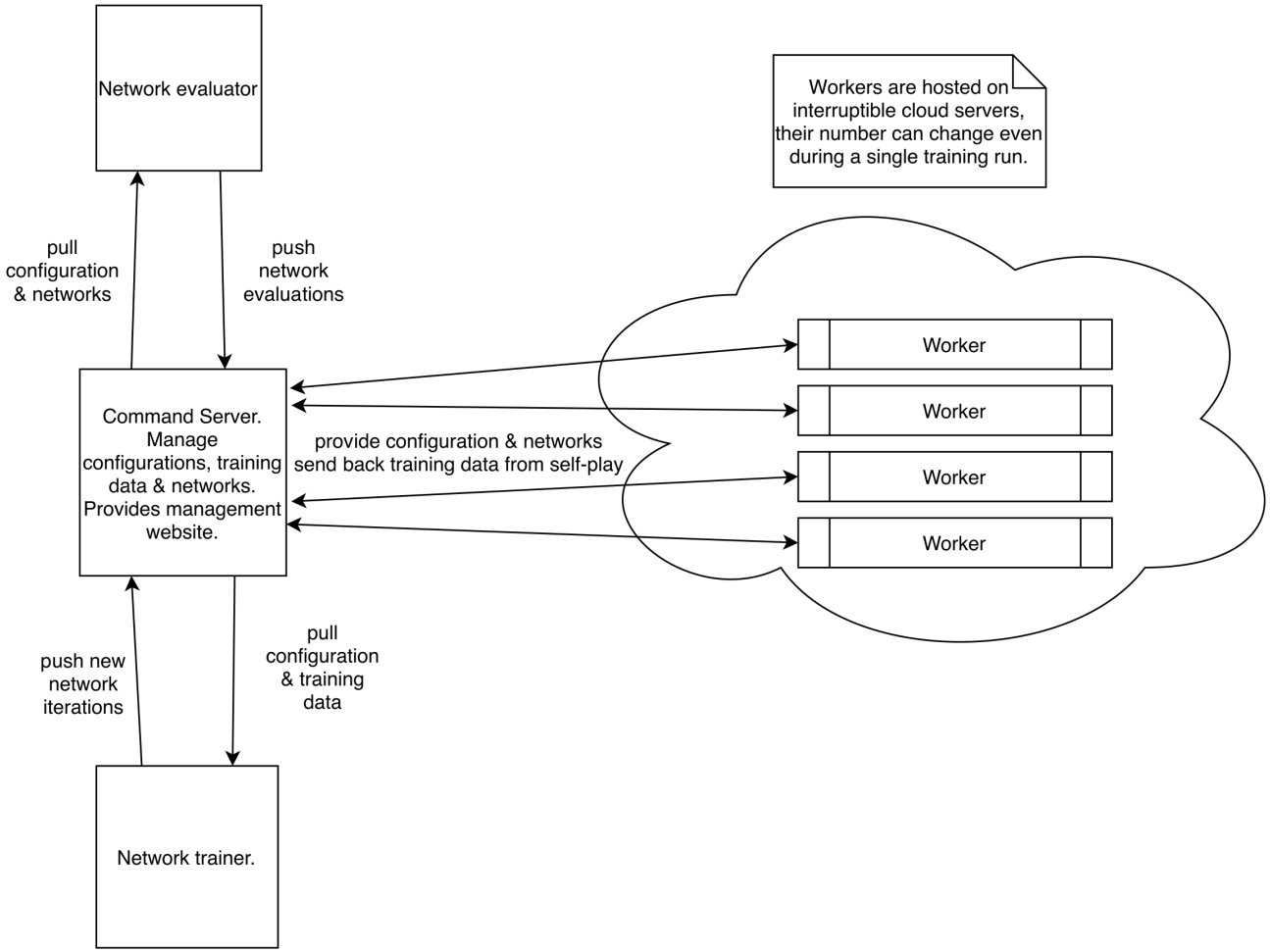
Aufbau der Experimente



Aufbau der Experimente



Aufbau der Experimente



Hyperparametersuche

Hyperparametersuche

- Optimiere wichtige Hyperparameter für Vier gewinnt

Hyperparametersuche

- Optimiere wichtige Hyperparameter für Vier gewinnt
- Bayesian Optimization Package verwendet

Hyperparametersuche

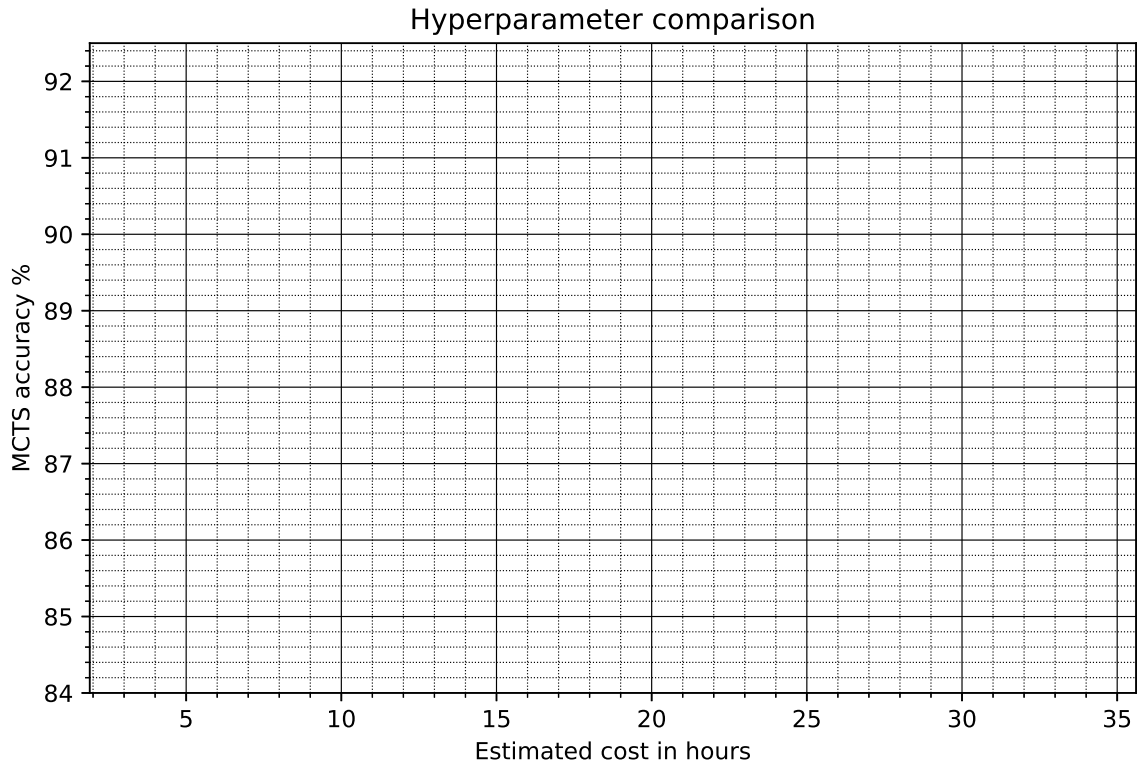
- Optimiere wichtige Hyperparameter für Vier gewinnt
- Bayesian Optimization Package verwendet
- Fitnessfunktion: Trainiere über 2 Stunden, messe Accuracy

Hyperparametersuche

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

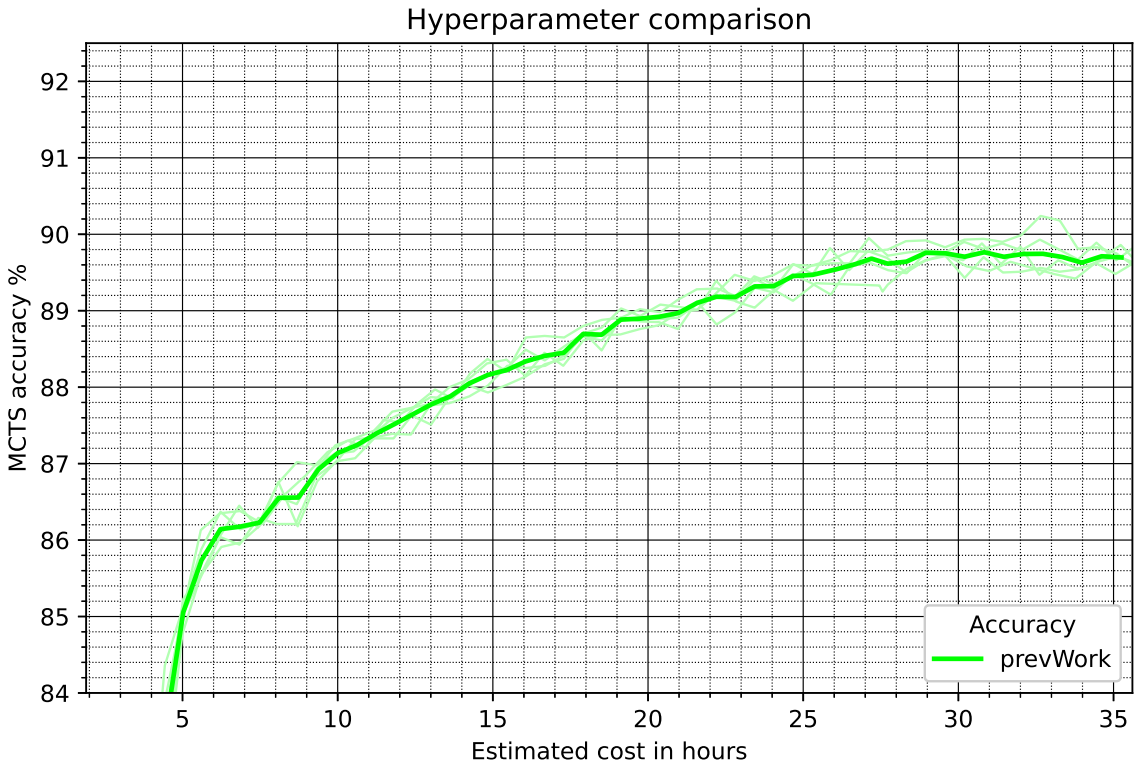
Hyperparametersuche

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



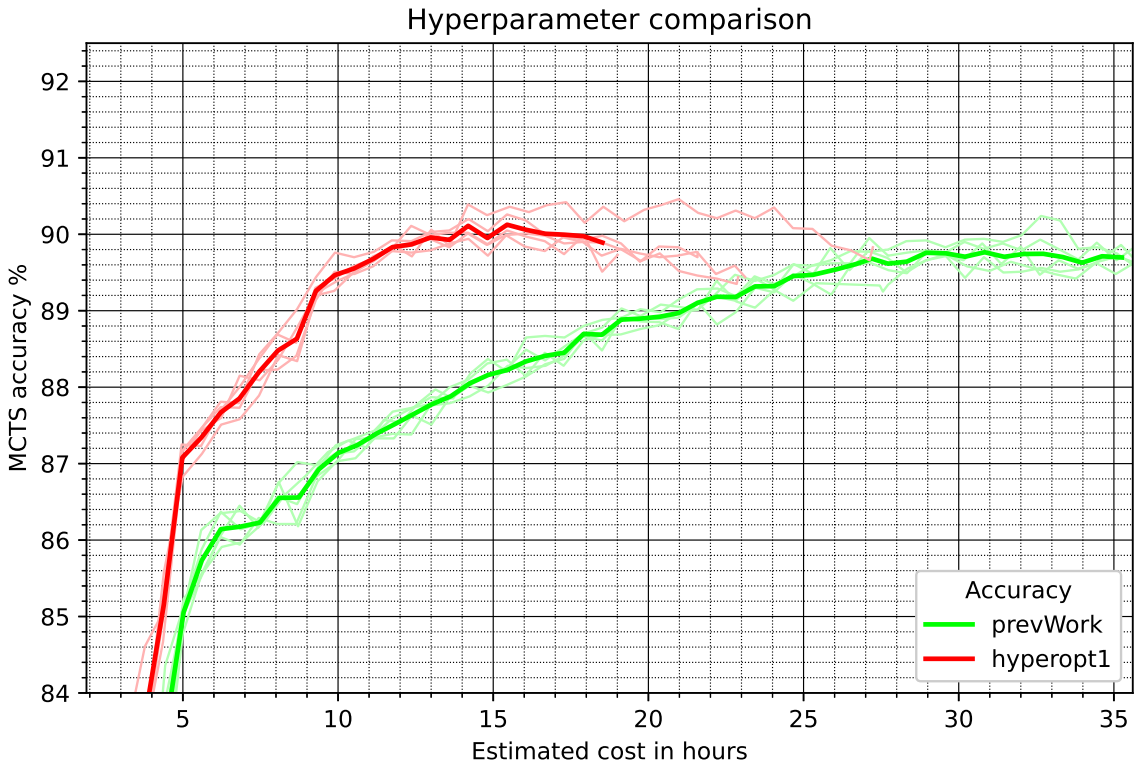
Hyperparametersuche

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



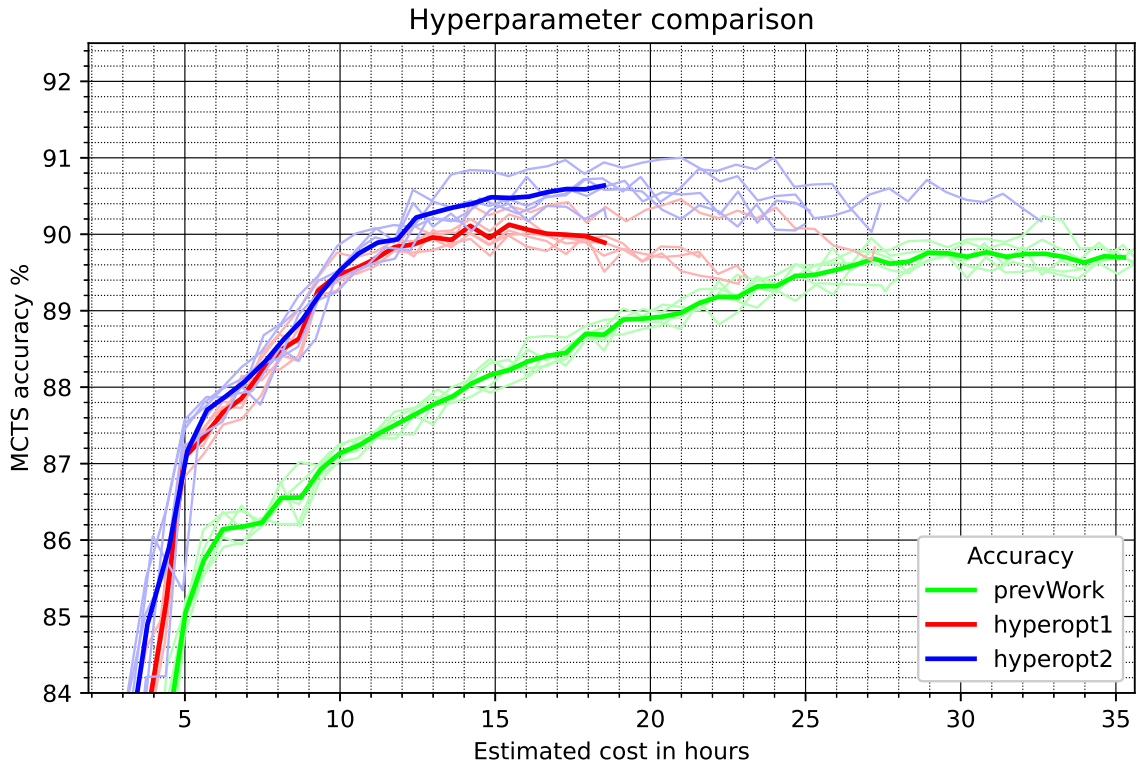
Hyperparametersuche

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



Hyperparametersuche

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



Extended baseline

Extended baseline

- Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten

Extended baseline

- Erweitere die Baselineimplementierung mit Verbesserungen anderer Arbeiten
  - Deduplikation

Extended baseline

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

Extended baseline

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

Extended baseline

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

Extended baseline

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

Extended baseline

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

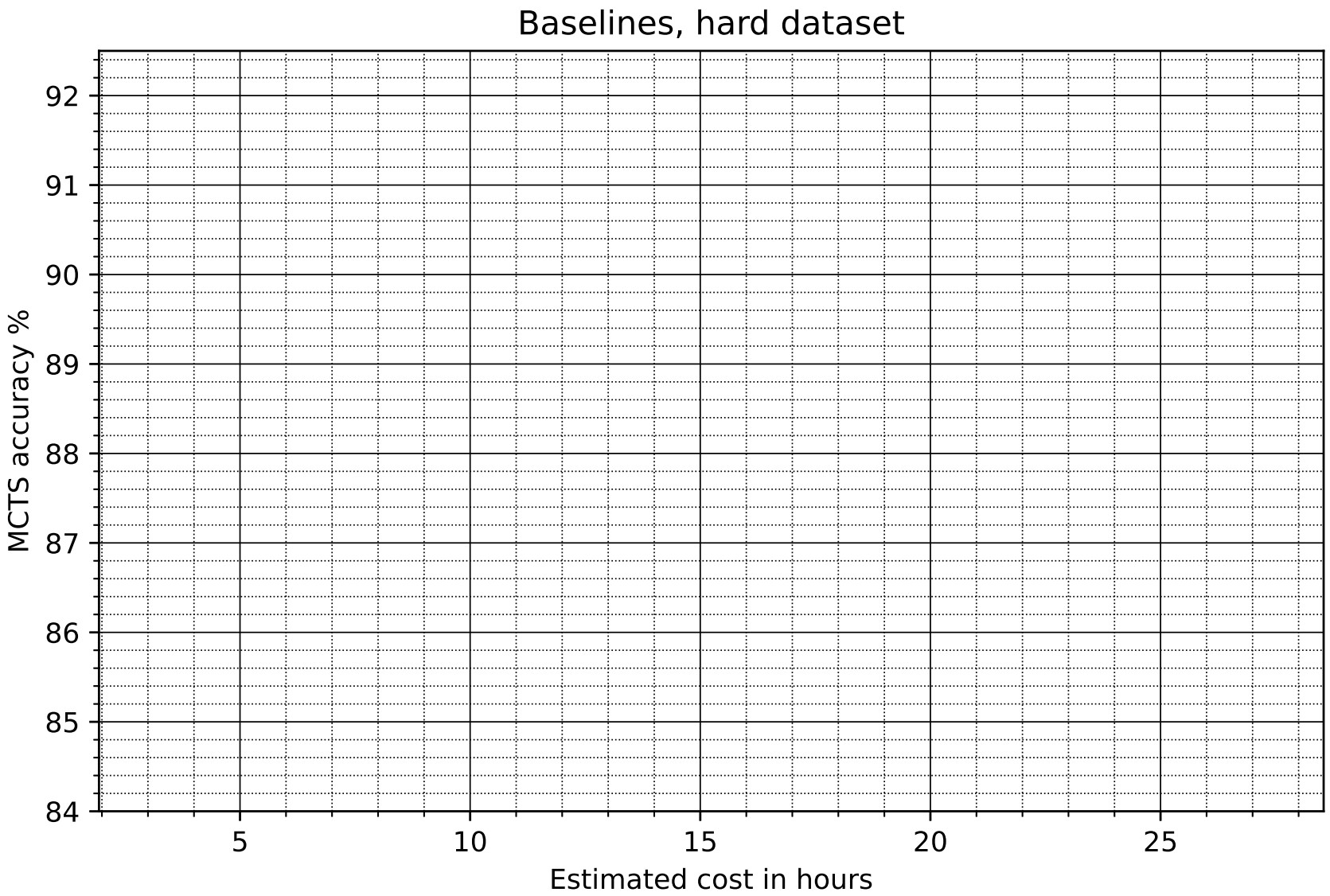
Extended baseline

- 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

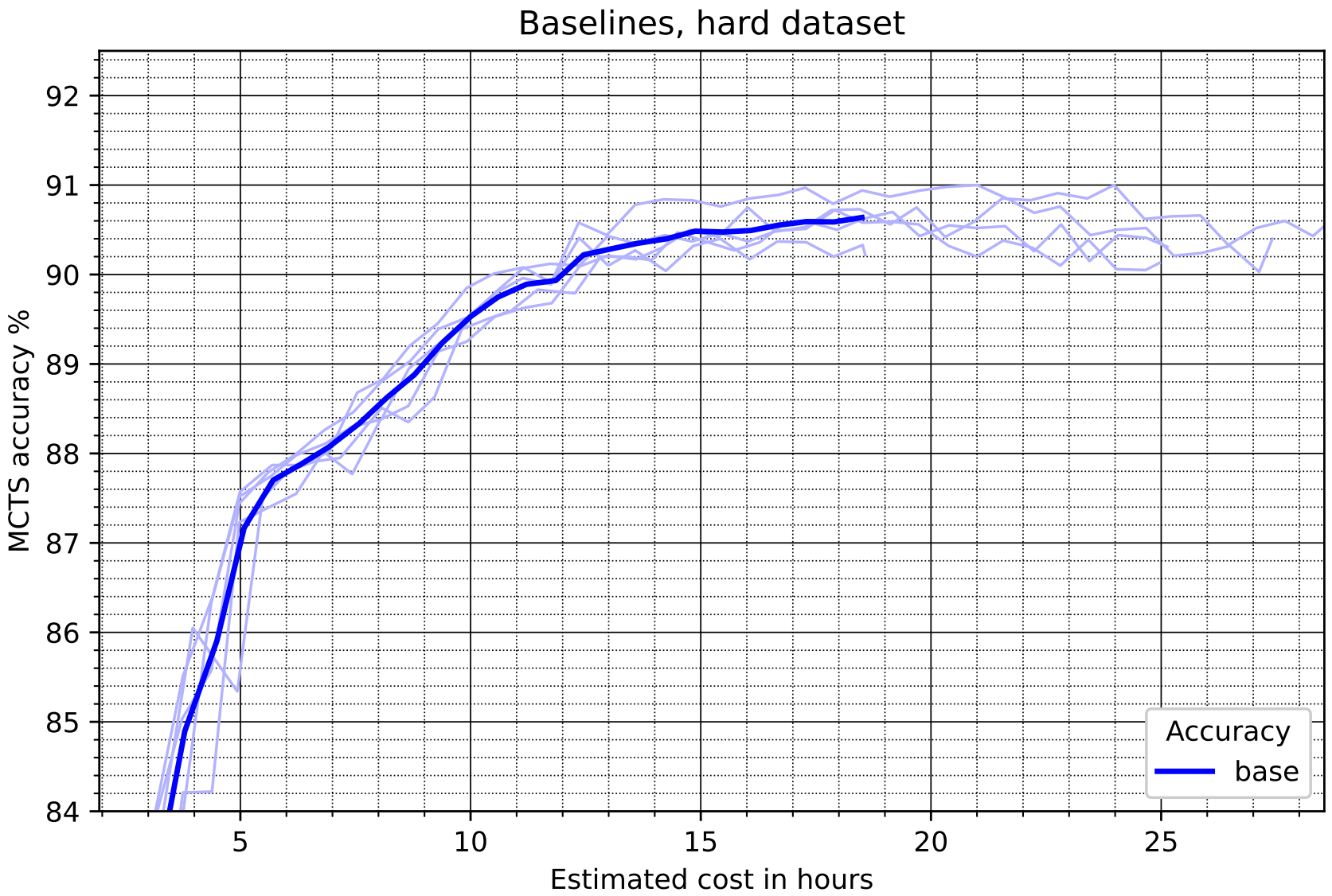
Extended baseline

- 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.

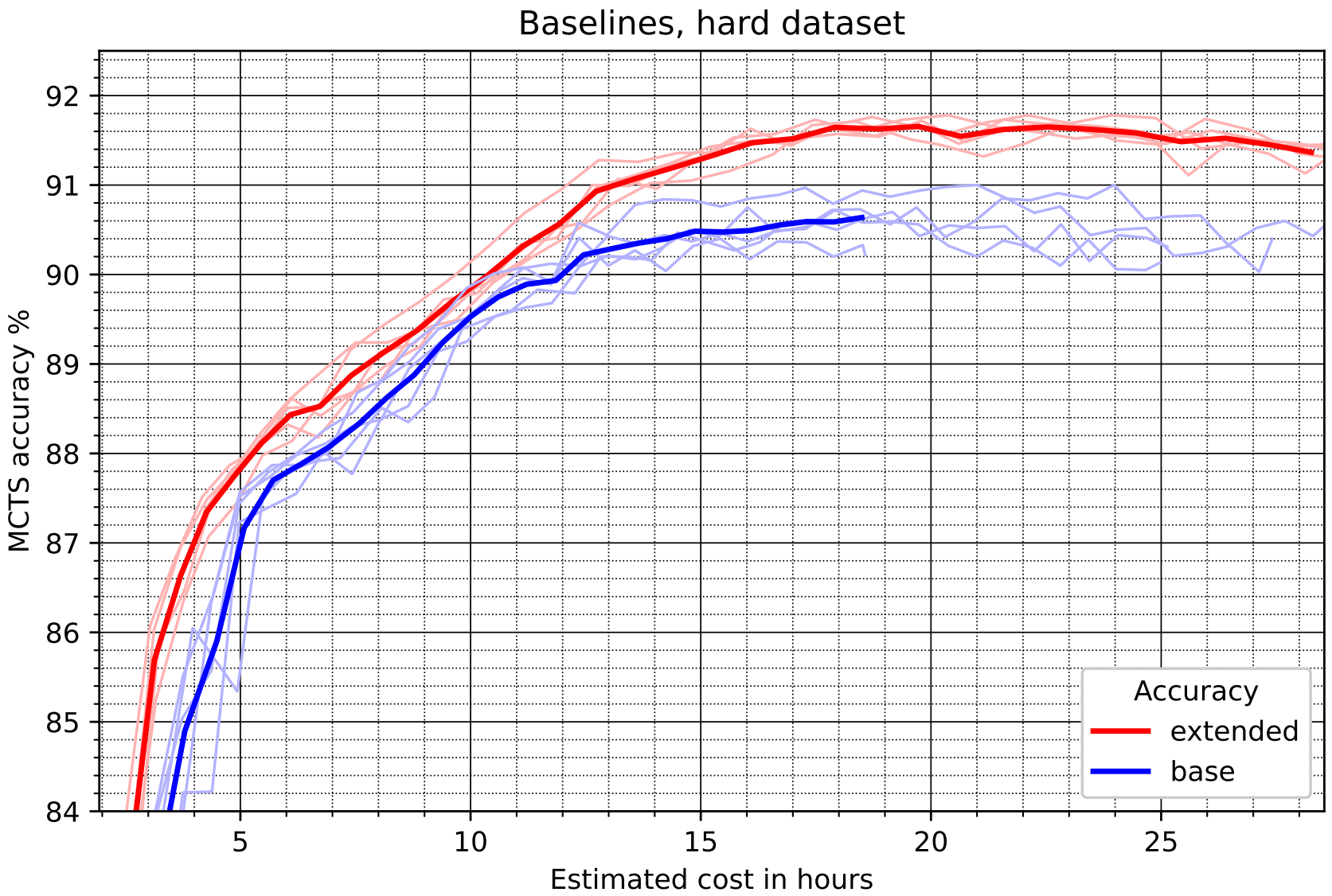
Extended baseline Ergebnisse



Extended baseline Ergebnisse



Extended baseline Ergebnisse



Inhalt

Implementierung: Evolutionary Self-play

Implementierung: Evolutionary Self-play

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

Implementierung: Evolutionary Self-play

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

Implementierung: Evolutionary Self-play

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

Implementierung: Evolutionary Self-play

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

Implementierung: Evolutionary Self-play

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

Implementierung: Evolutionary Self-play

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

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

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

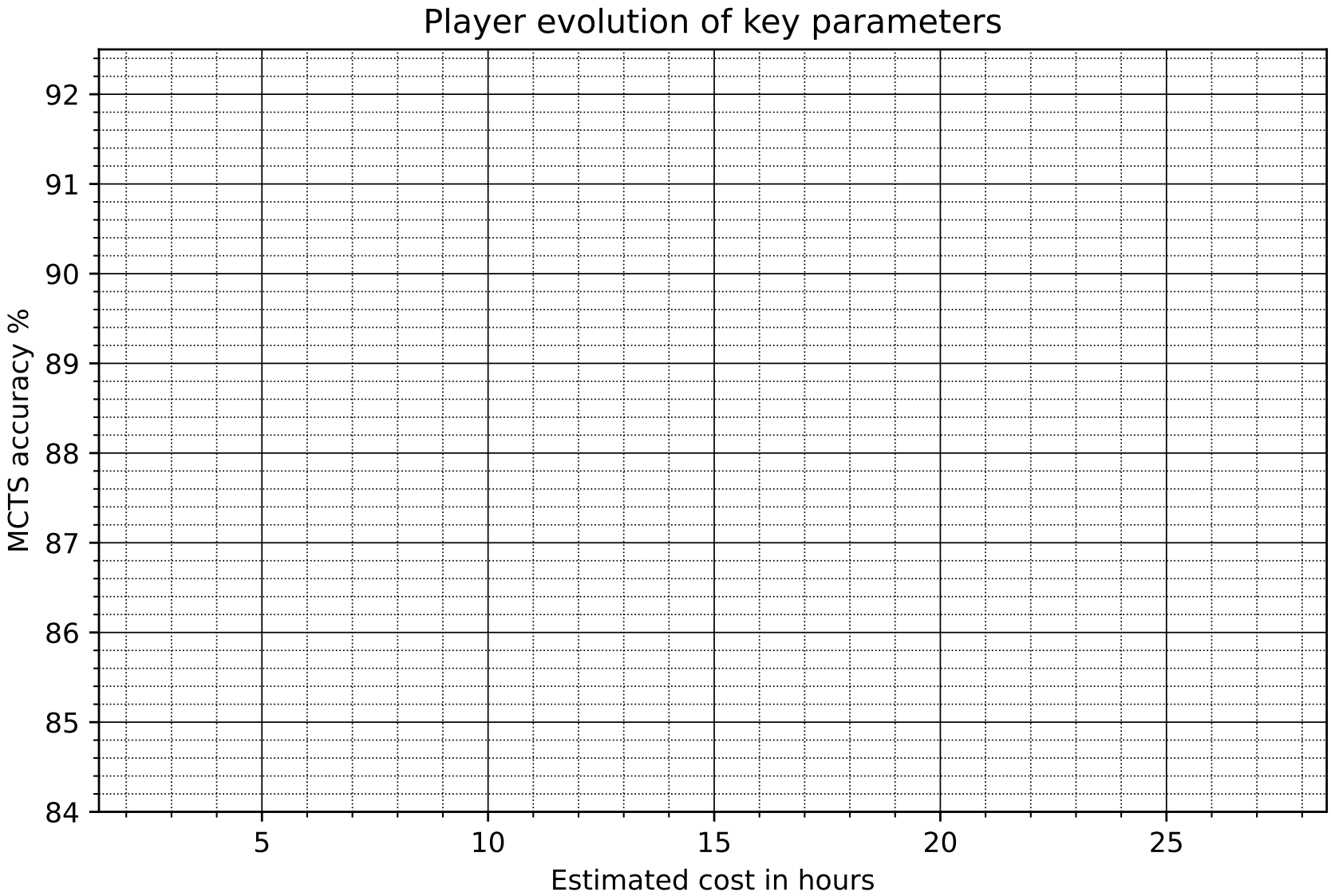
- Zwei Arten von Hyperparametern untersucht
  - Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.

Implementierung: Evolutionary Self-play

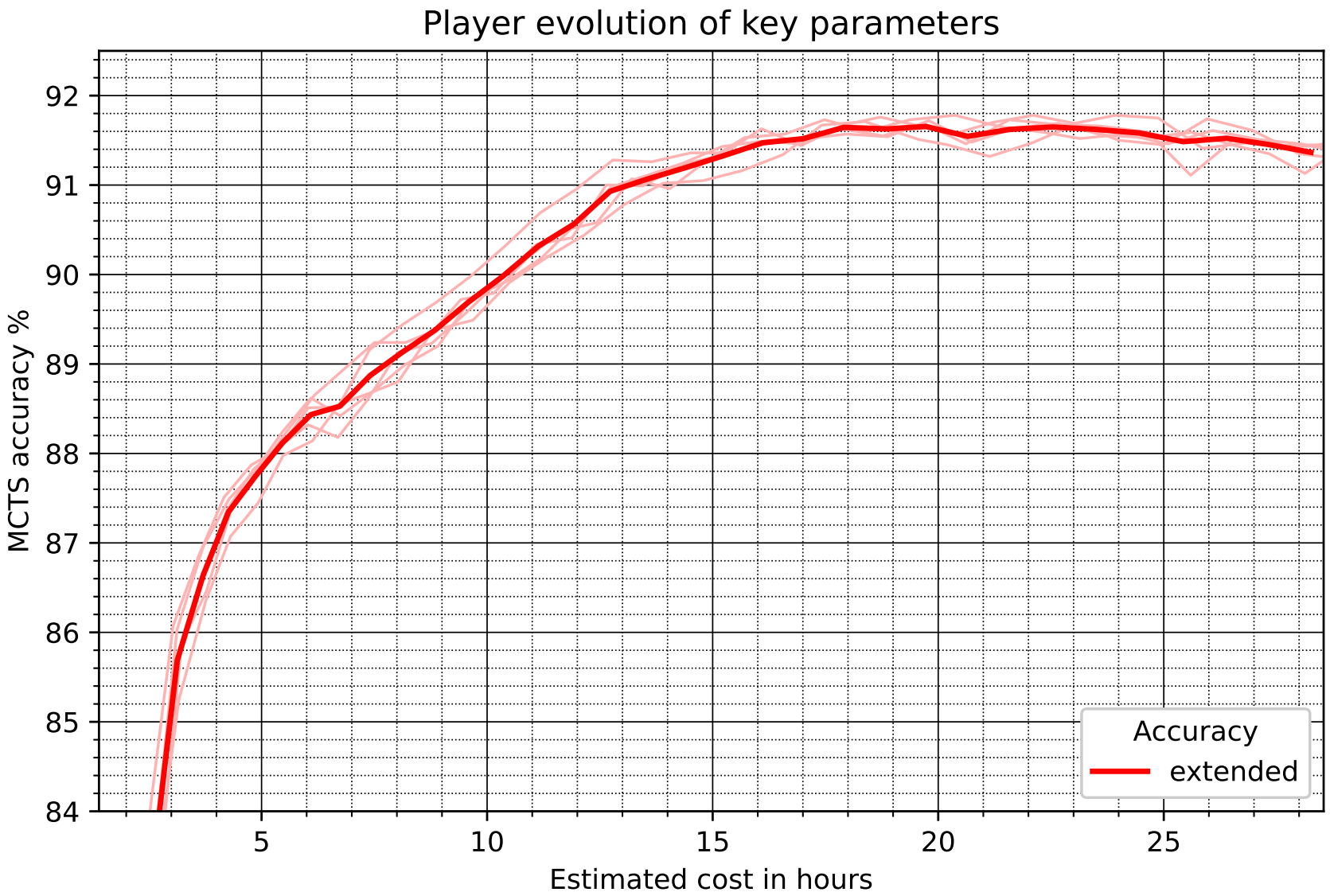
- Verwende die Self-play-phase zur Evolution von Hyperparametern
- Beschränkt auf Hyperparameter, welche sich leicht ändern lassen
- Implementiert als eine Liga von Spielern.
- Ein Spieler ist ein Hyperparameterset
- Bewerte Spieler mit Elo
- Verwende Gaussian Mutation um die besten Spieler zu mutieren

- Zwei Arten von Hyperparametern untersucht
  - Verwendung von Kullback-Leibler divergence um "Denkzeit" zu wählen.
  - MCTS Parameter: cpuct, fpu, drawValue

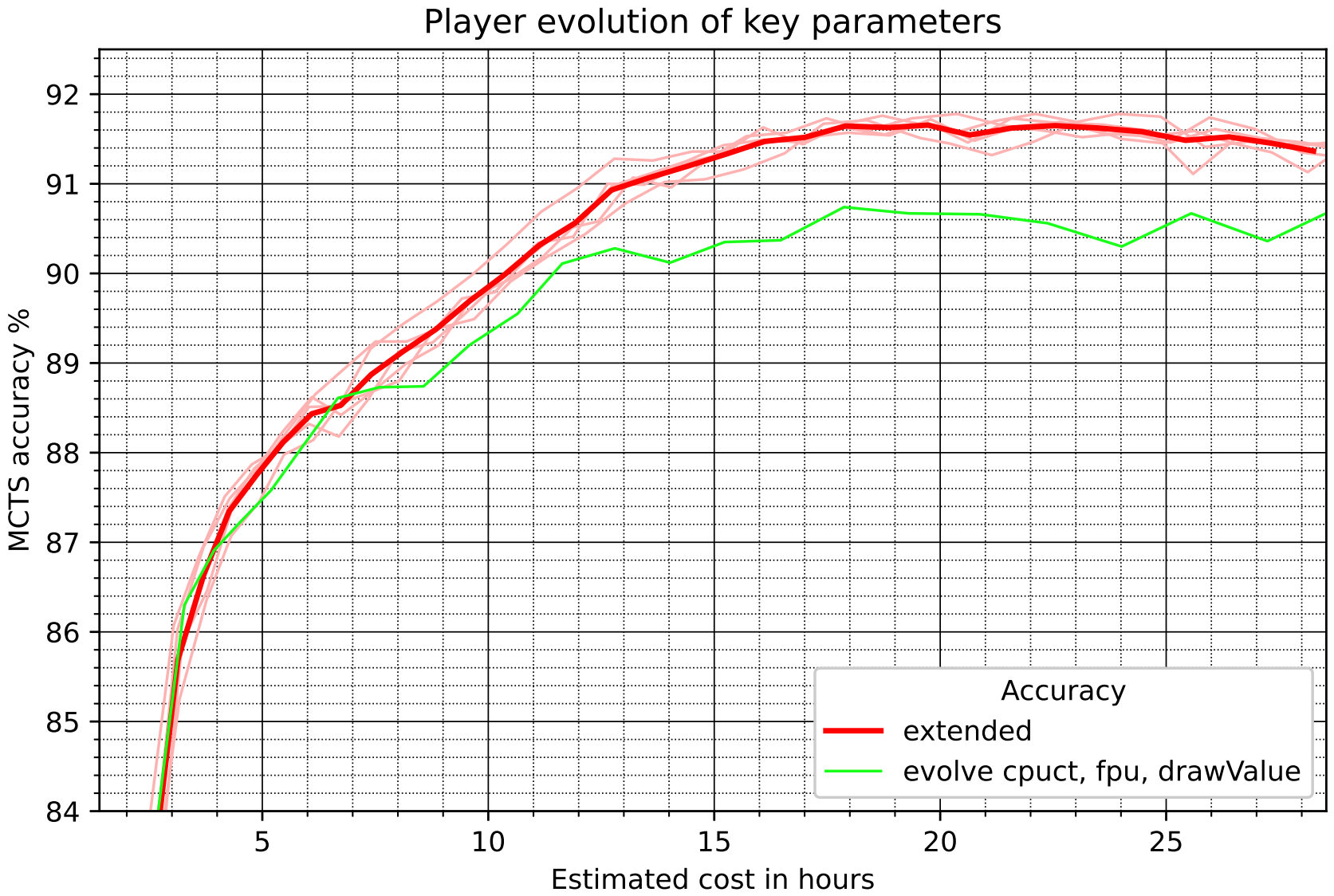
Erste Ergebnisse



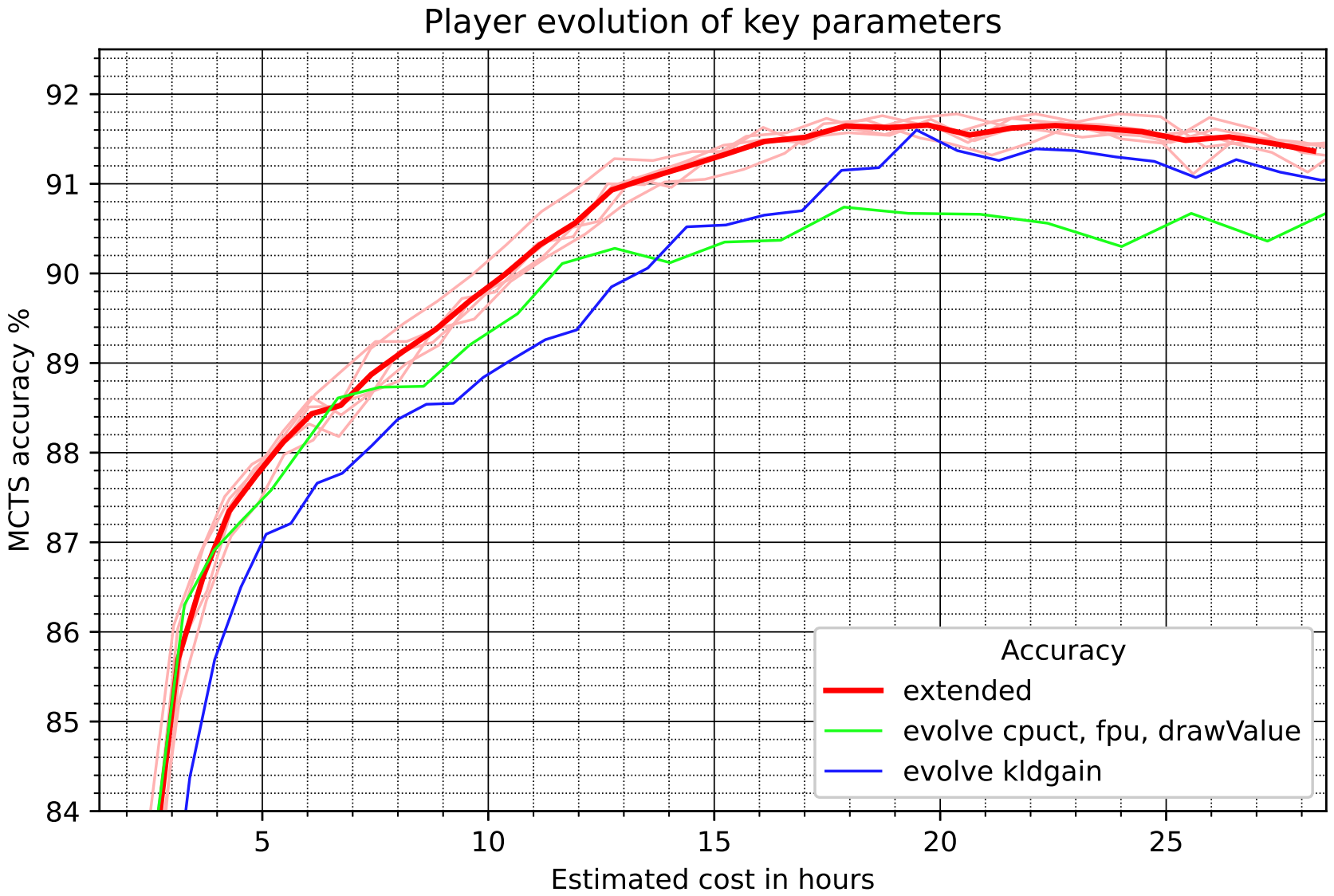
Erste Ergebnisse



Erste Ergebnisse



Erste Ergebnisse



Untersuchung

Untersuchung

- Bedingungen für erfolgreiche Evolution:

Untersuchung

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

Untersuchung

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

Erkennt die Liga gute Parameter?

Erkennt die Liga gute Parameter?

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

Erkennt die Liga gute Parameter?

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

Erkennt die Liga gute Parameter?

- "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?

- "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?

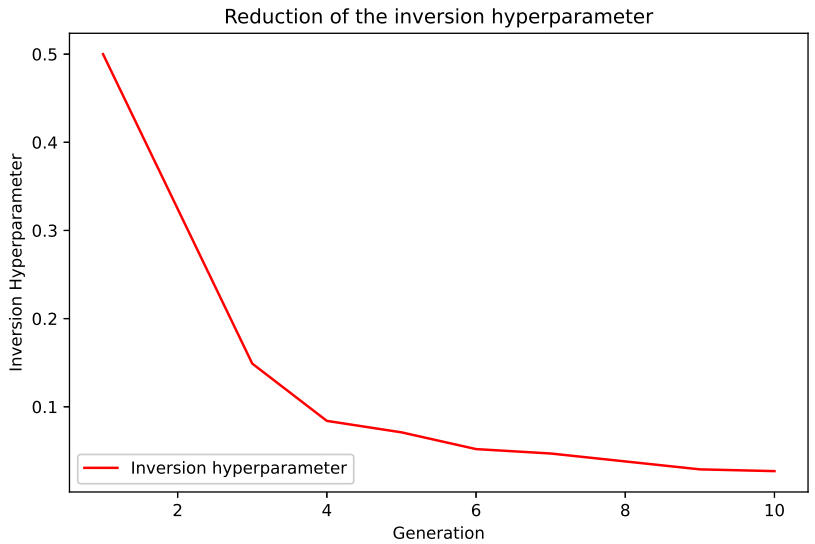
- "Gut" im Sinne der Evolution: Gewinne viele Spiele
- Erfinde einen neuen Parameter für den der optimale Wert klar ist
- Stelle die Bewertung der Spielzüge auf den Kopf.
- Wertebereich von 0 bis 1.
  - Bei 0 hat der Parameter keinen Effekt

Erkennt die Liga gute Parameter?

- "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.

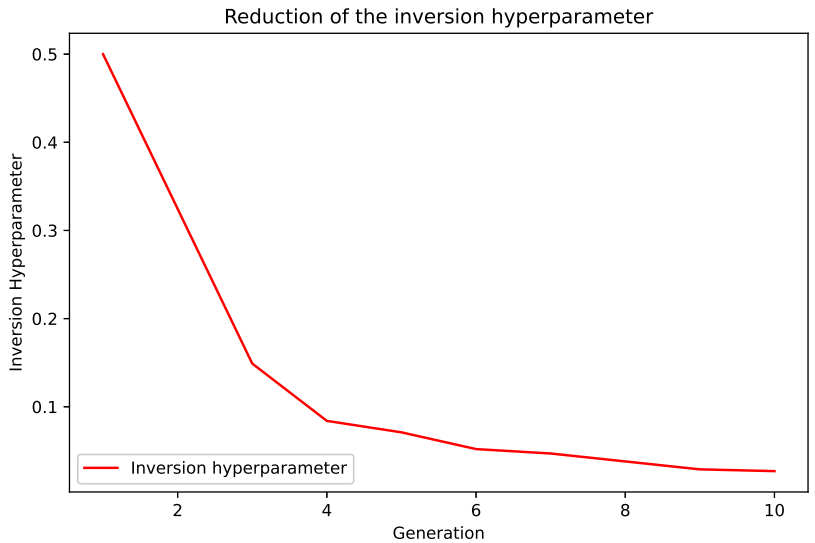
Erkennt die Liga gute Parameter?

- "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?

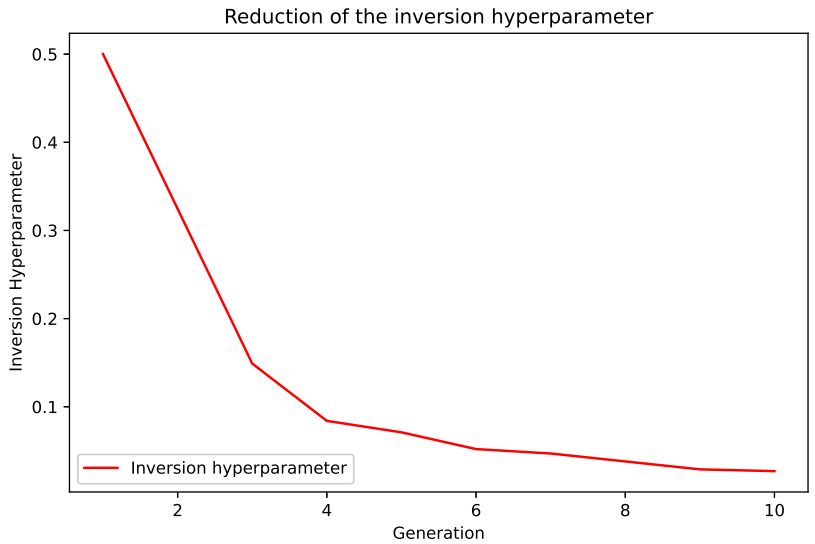
- "Gut" im Sinne der Evolution: Gewinne viele Spiele
- Erfinde einen neuen Parameter für den der optimale Wert klar ist
- Stelle die Bewertung der Spielzüge auf den Kopf.
- Wertebereich von 0 bis 1.
  - Bei 0 hat der Parameter keinen Effekt
  - Bei 1 werden gute Züge selten gespielt, schlechte am häufigsten.



- Ein voller Trainingslauf hat eher 30 bis 50 Generationen

Erkennt die Liga gute Parameter?

- "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.



- Ein voller Trainingslauf hat eher 30 bis 50 Generationen
- Die Liga mit Evolution funktioniert

Folgt Lernfortschritt aus mehr Siegen?

Folgt Lernfortschritt aus mehr Siegen?

- Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver

Folgt Lernfortschritt aus mehr Siegen?

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

Folgt Lernfortschritt aus mehr Siegen?

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

Folgt Lernfortschritt aus mehr Siegen?

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

Folgt Lernfortschritt aus mehr Siegen?

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

Folgt Lernfortschritt aus mehr Siegen?

- Lernfortschritt bedeutet höhere Übereinstimmung mit dem Solver
- Vergleiche 3 Hyperparametersets
  - Bester Spieler aus der Evolution
  - Baseline Parameter
  - Bayesian Optimization
- Vergleiche die 3 Optionen in 1000 Spiele Matches

Folgt Lernfortschritt aus mehr Siegen?

Folgt Lernfortschritt aus mehr Siegen?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

Folgt Lernfortschritt aus mehr Siegen?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- Klarer Sieger: Die Evolution

Folgt Lernfortschritt aus mehr Siegen?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- Klarer Sieger: Die Evolution
- Viele gewonne Spiele bedeuten also nicht hoher Lernfortschritt

Folgt Lernfortschritt aus mehr Siegen?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- 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?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- 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?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- 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?

Evolved Player vs...	
Baseline	557W, 155L, 228D
Bayesian Opt.	442W, 388L, 190D

- 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

Implementierung: Games as trees

- Exploration durch Zurücksetzen an kritische Positionen

Implementierung: Games as trees

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

Implementierung: Games as trees

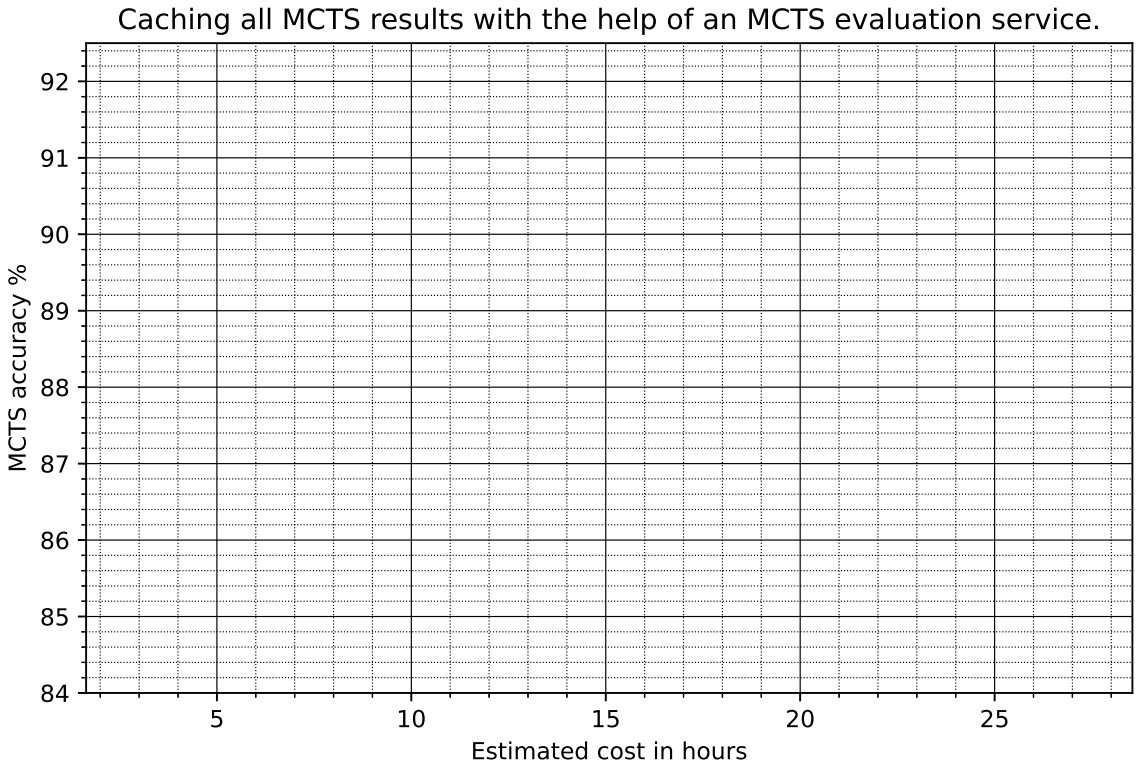
- Exploration durch Zurücksetzen an kritische Positionen
- Spiele als MCTS-Baum
- Notwendigkeit für MCTS-Evaluation Service

Implementierung: Games as trees

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

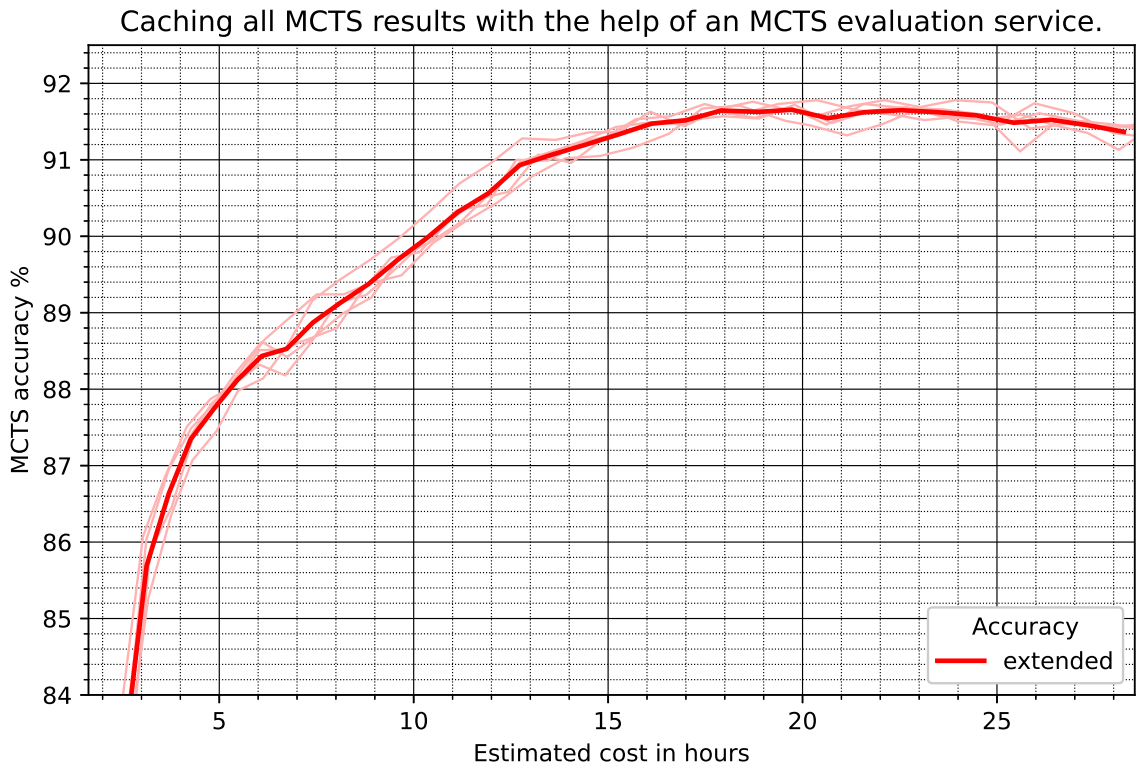
Implementierung: Games as trees

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



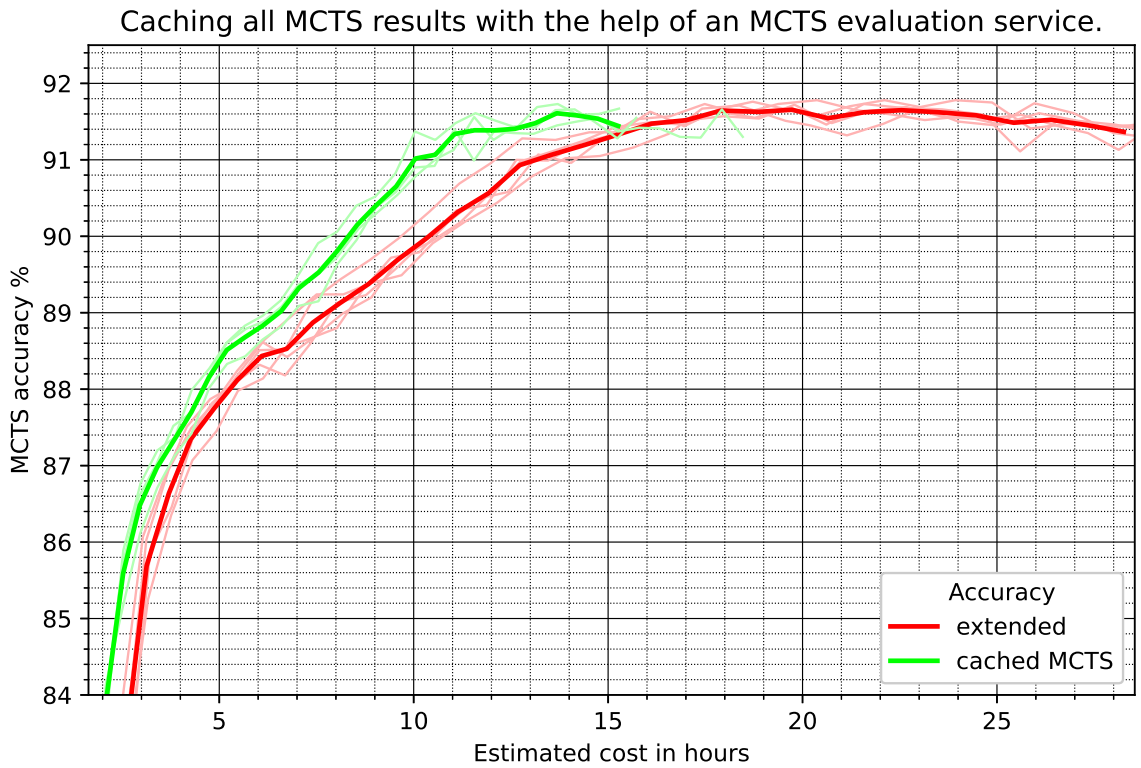
Implementierung: Games as trees

- 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

- 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

Zurücksetzen auf kritische Position

- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen

Zurücksetzen auf kritische Position

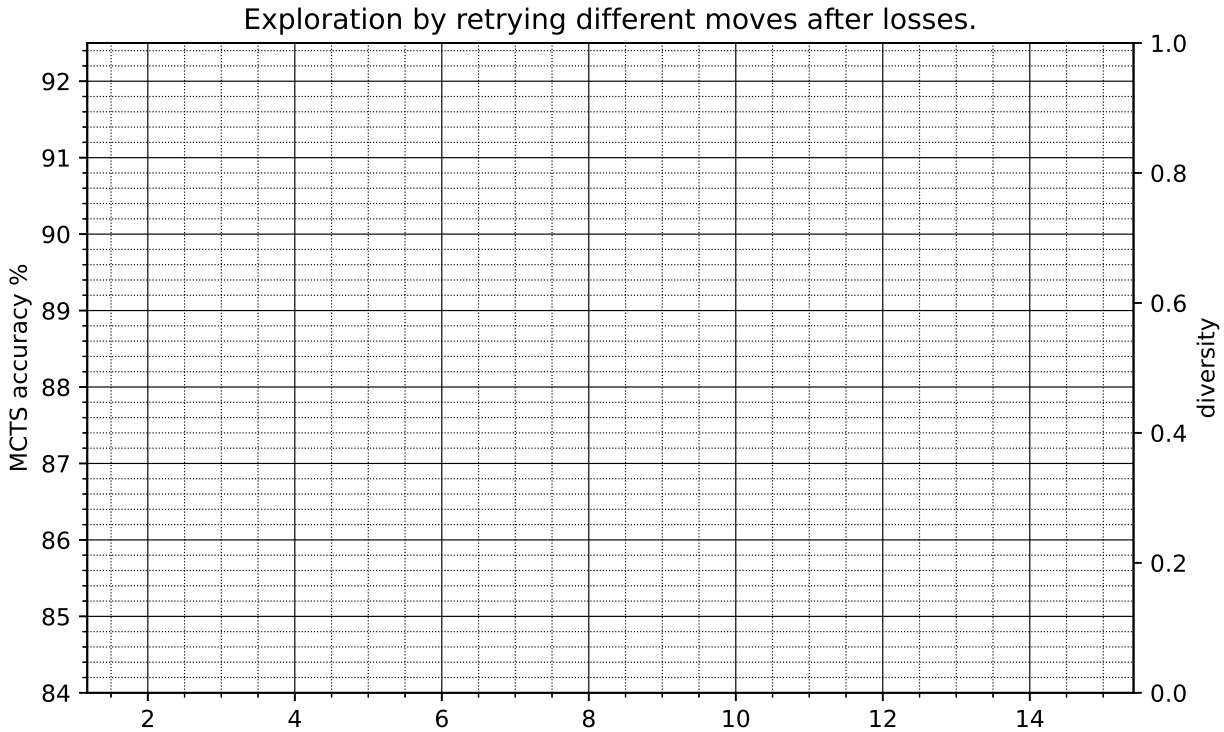
- Nach einer Niederlage darf der Verlierer einen Zug zurücknehmen
- Wähle Position anhand der Entwicklung der Positionsevaluation

Zurücksetzen auf kritische Position

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

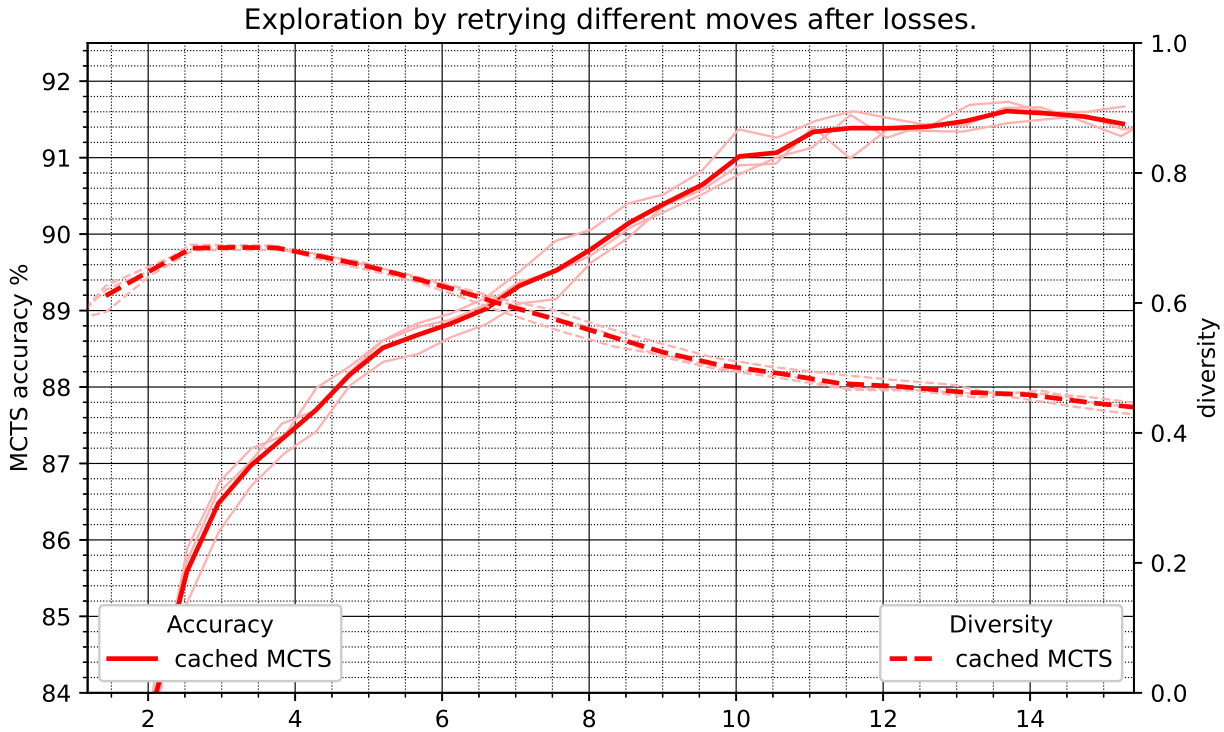
Zurücksetzen auf kritische Position

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



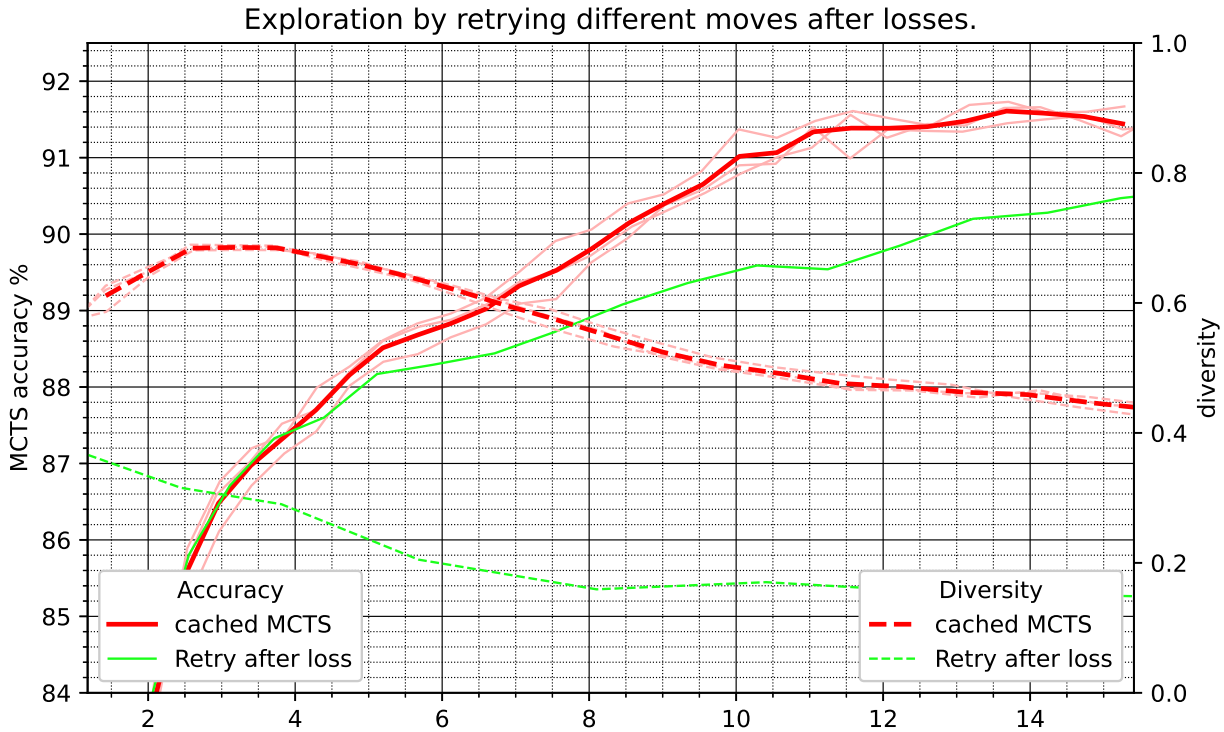
Zurücksetzen auf kritische Position

- 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

- 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

Spiele als MCTS-Baum

- Exploration-Exploitation: MCTS macht das

Spiele als MCTS-Baum

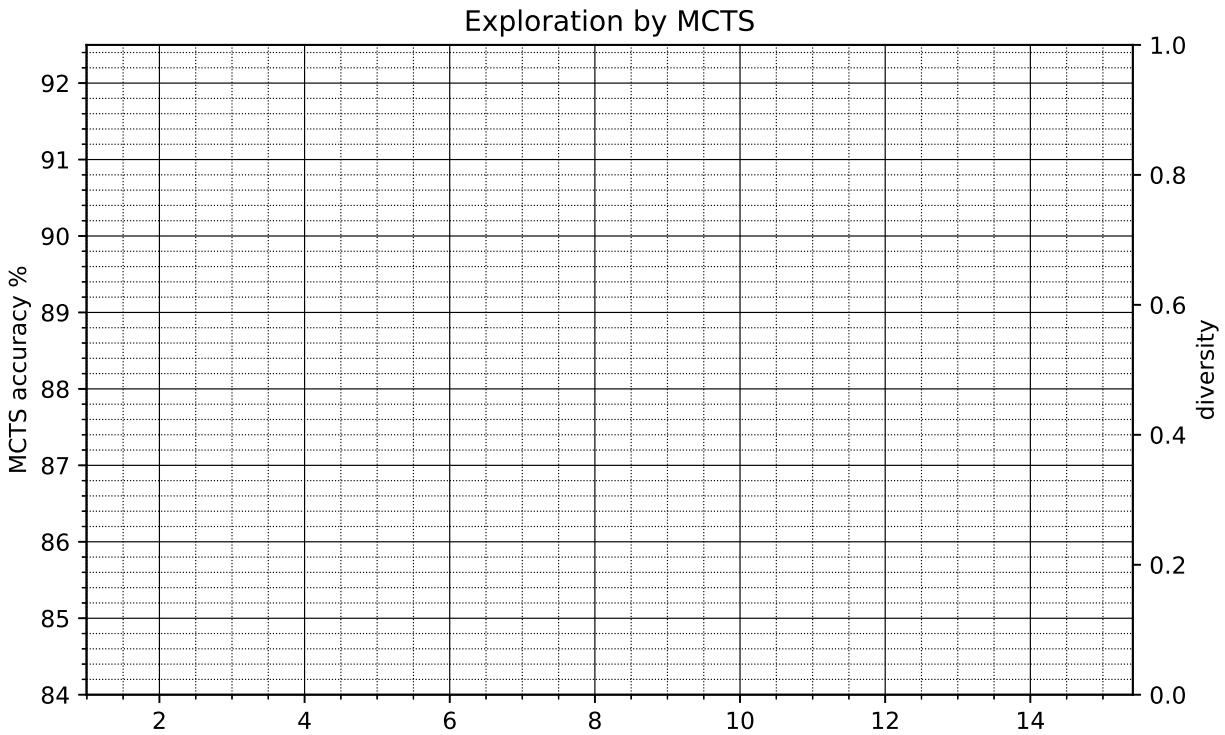
- Exploration-Exploitation: MCTS macht das
- Baue einen einzigen MCTS Baum

Spiele als MCTS-Baum

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

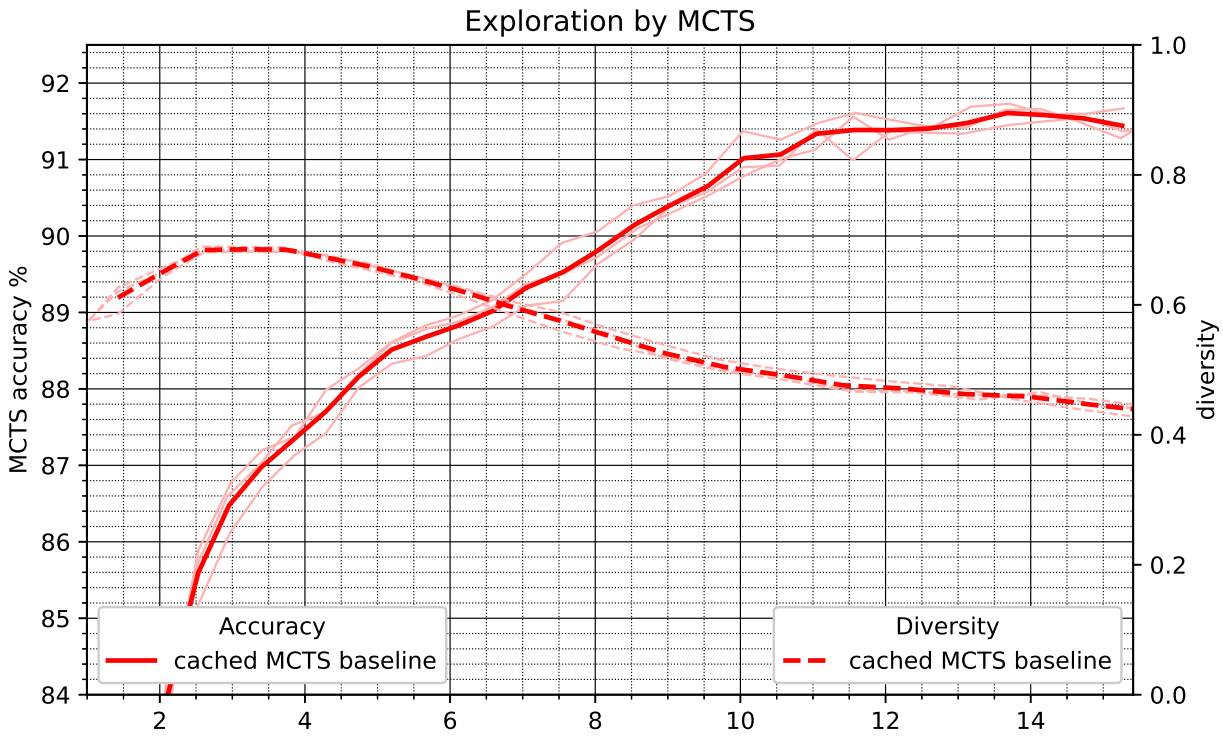
Spiele als MCTS-Baum

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



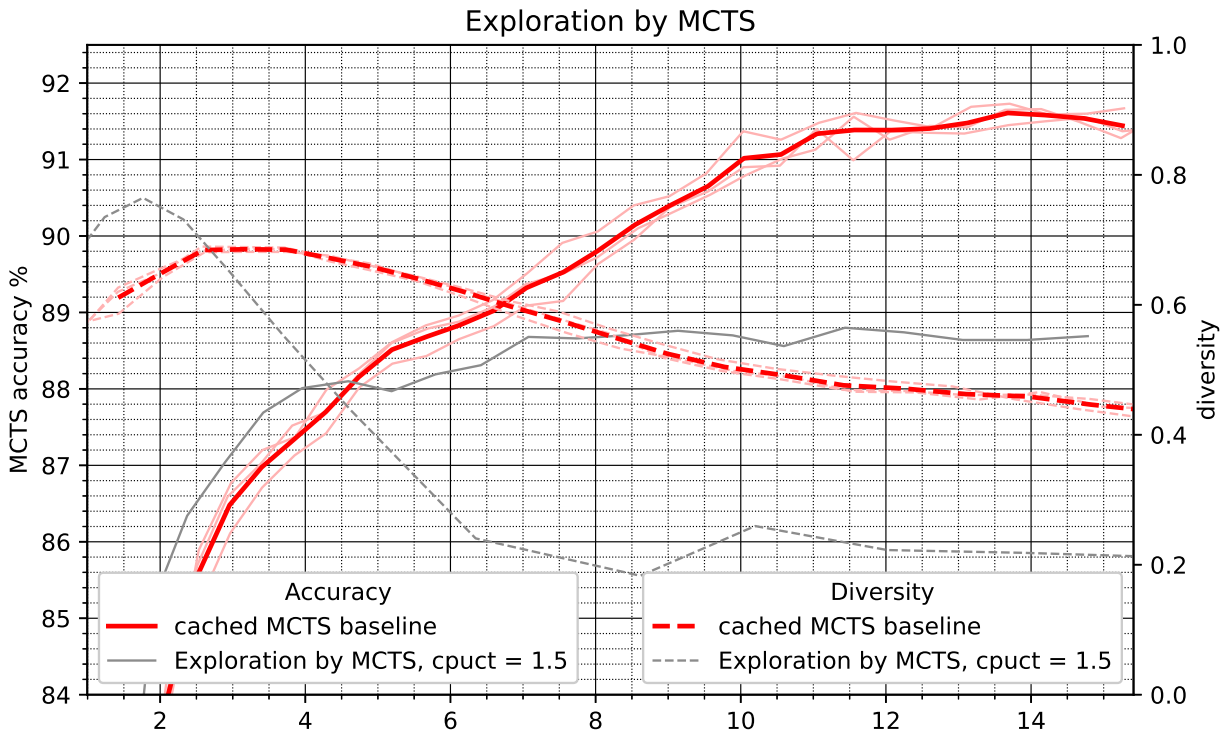
Spiele als MCTS-Baum

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



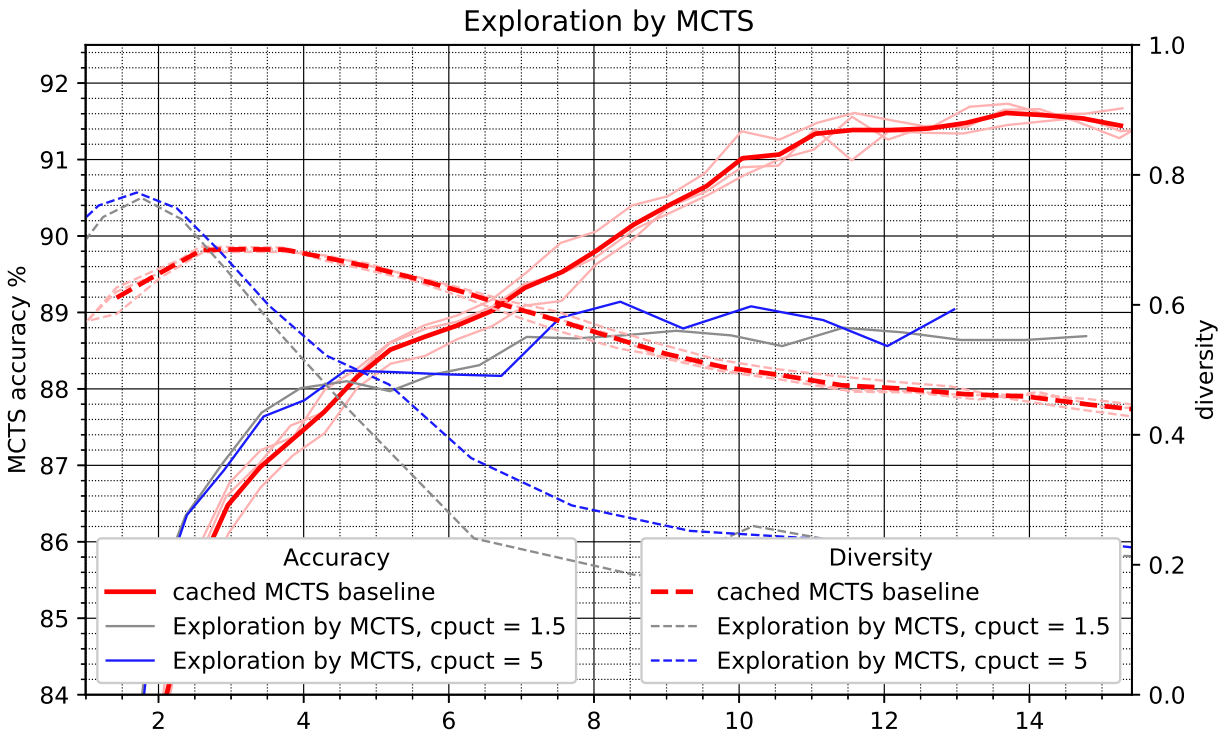
Spiele als MCTS-Baum

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



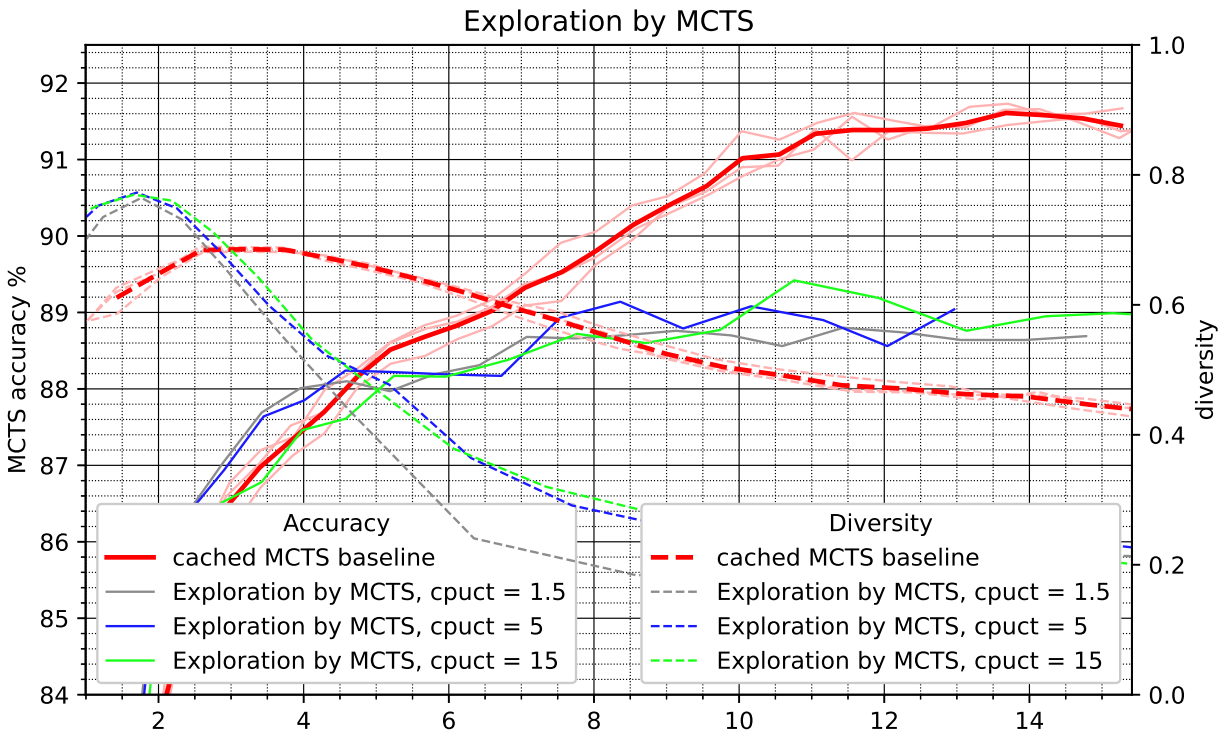
Spiele als MCTS-Baum

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



Spiele als MCTS-Baum

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



Implementierung: Auxiliary features

Implementierung: Auxiliary features

- Trainiere ein kleines Netzwerk, ca. 70k Parameter

Implementierung: Auxiliary features

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

Implementierung: 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

- 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

Untersuchung

Untersuchung

- Positiver Effekt auf initialen Anstieg der Spielstärke

Untersuchung

- Positiver Effekt auf initialen Anstieg der Spielstärke
- Negativer Effekt auf finale Spielstärke

Untersuchung

- Positiver Effekt auf initialen Anstieg der Spielstärke
- Negativer Effekt auf finale Spielstärke
- Ein zufälliges Netzwerk zeigt diesen Effekt deutlich weniger

Untersuchung

- 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.

Untersuchung

- 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:

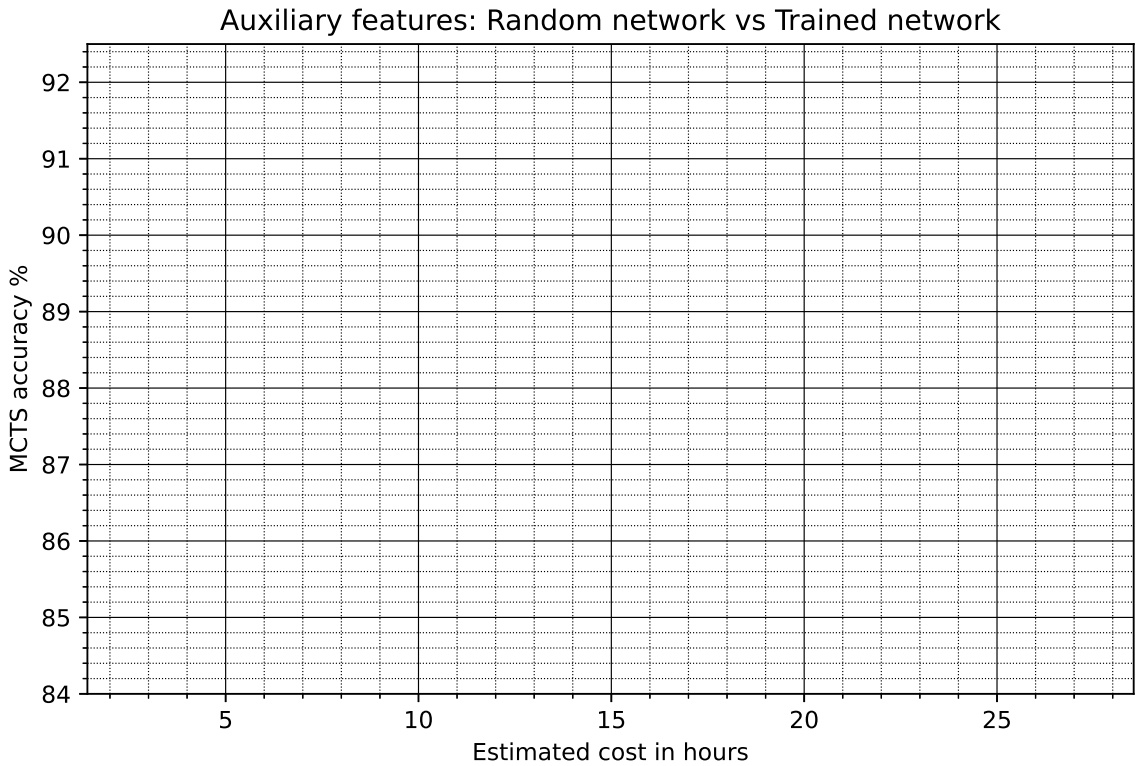
Untersuchung

- 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 Netzwerks

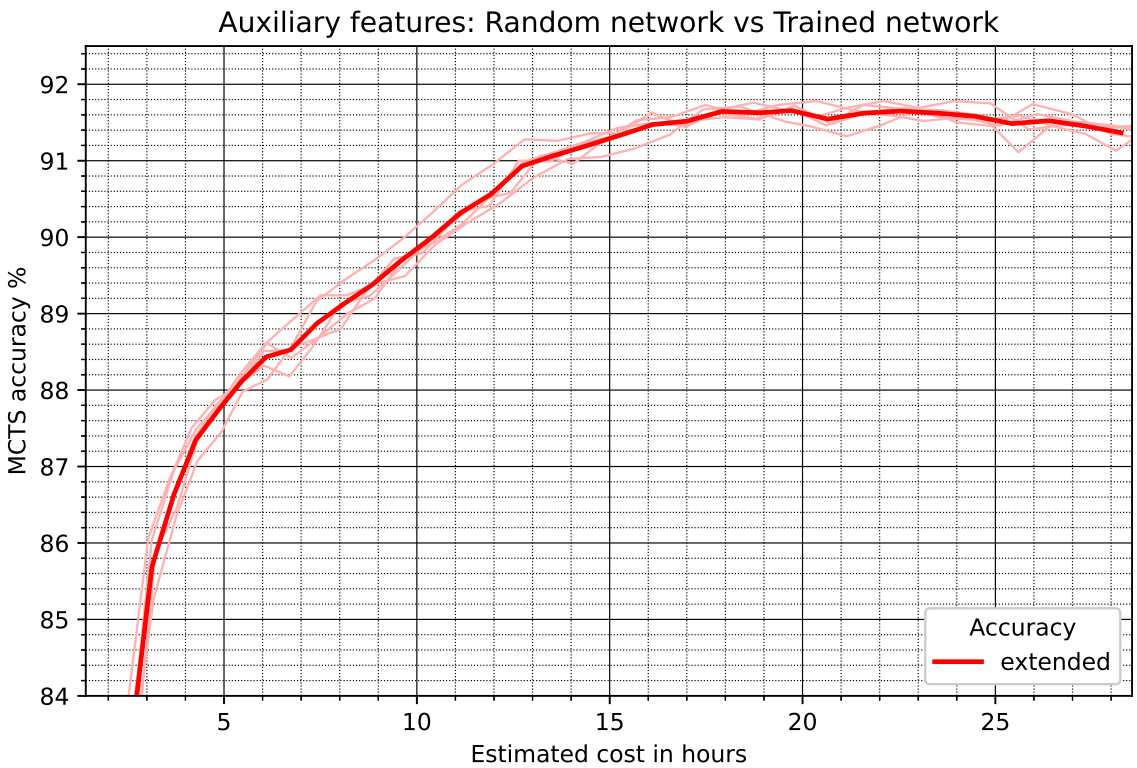
Untersuchung

- 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 Netzwerks
  - Finale Spielstärke wird gestört

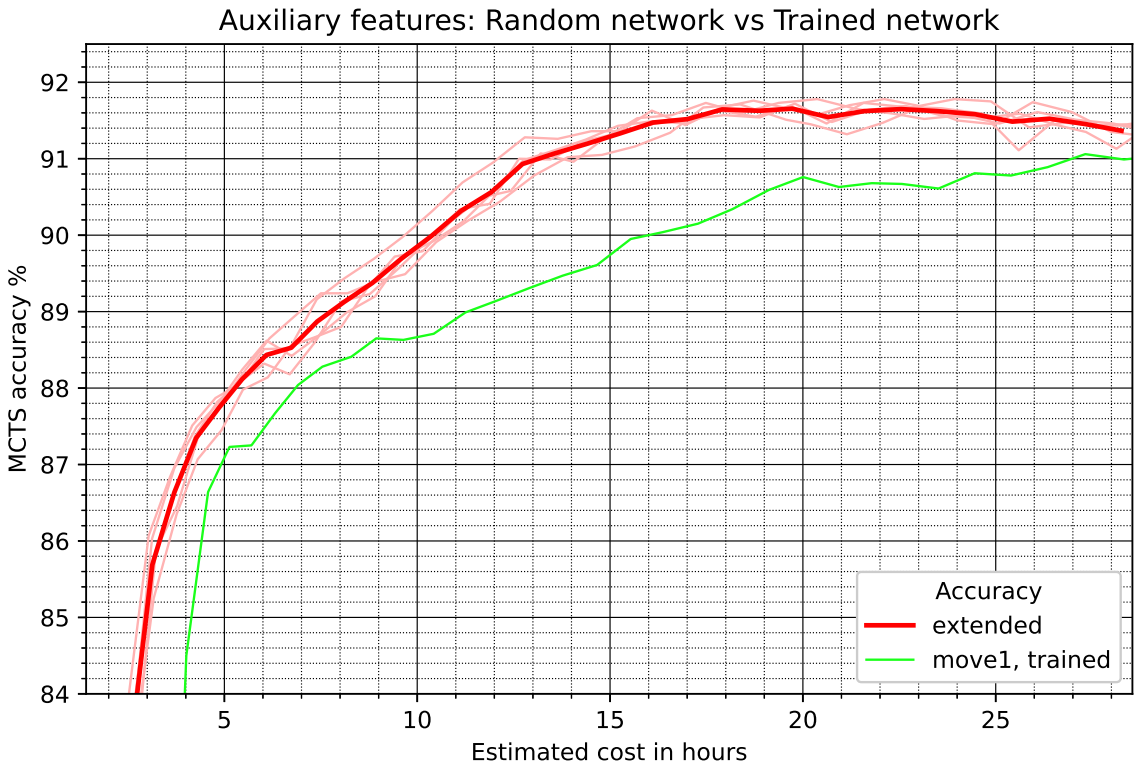
Untersuchung



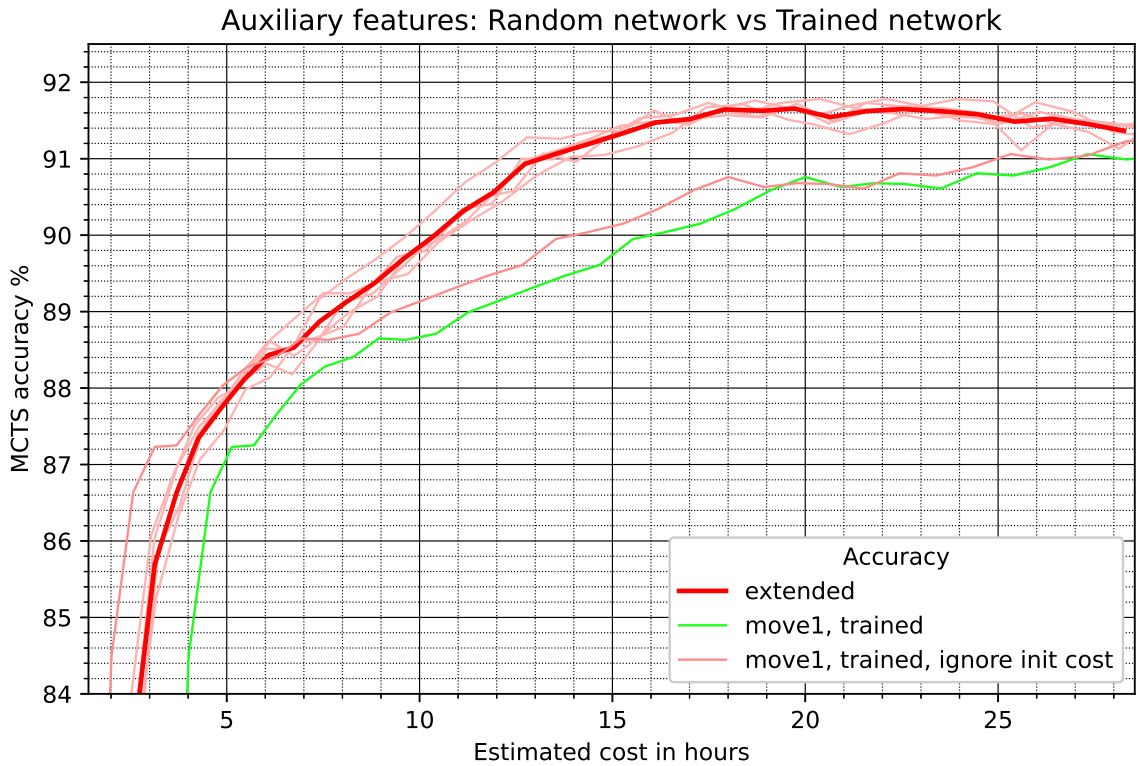
Untersuchung



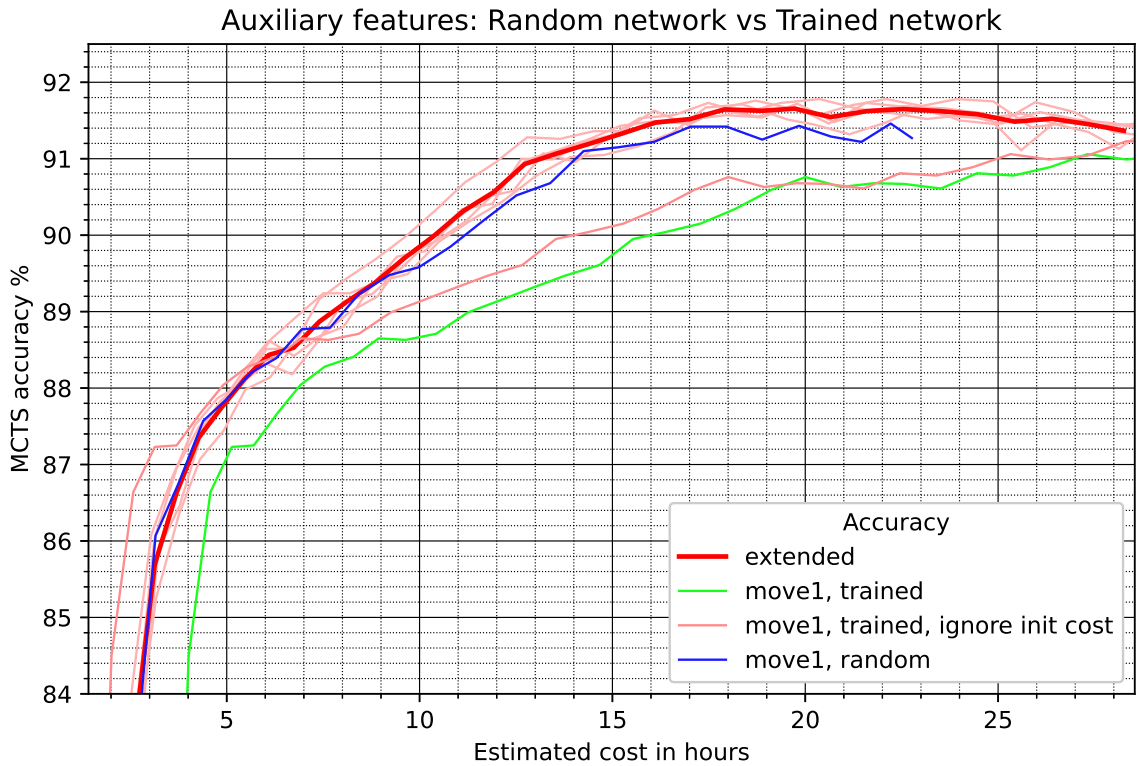
Untersuchung



Untersuchung



Untersuchung



Das Netzwerk wachsen lassen

Das Netzwerk wachsen lassen

- Beginne den AlphaZero Trainingslauf mit kleinem Netzwerk

Das Netzwerk wachsen lassen

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

Das Netzwerk wachsen lassen

- 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

- 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

- 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

Finale Spielstärke darf nicht geschadet werden

- Auxiliary features dürfen nicht der finalen Spielstärke schaden

Finale Spielstärke darf nicht geschadet werden

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

Finale Spielstärke darf nicht geschadet werden

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

Fazit

Fazit

- AlphaZero Experimentalframework entwickelt

Fazit

- AlphaZero Experimentalframework entwickelt
- Keine großen Verbesserungen gefunden

Fazit

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

Fazit

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

Fazit

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

Fazit

- 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

Fazit

- 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

Fazit

- 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

Fazit

- 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

Fazit

- 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

Inhalt

Referenzen

Danke für Ihre Aufmerksamkeit.

- Maybe do some references here...