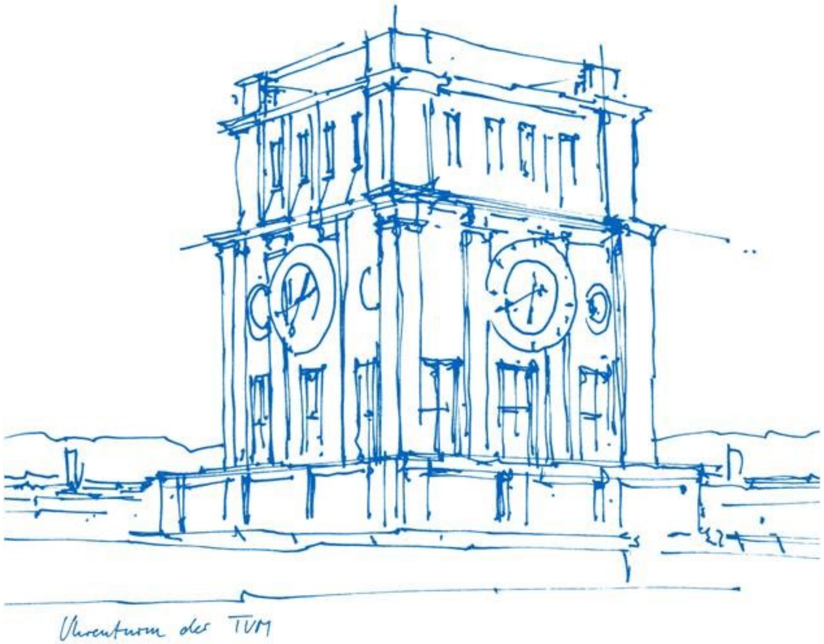
Chess & Reinforcement Learning

Guillaume Pelat

November 12th, 2021



Advisor:

Prof. Matthias Nießner

Content

- Anatomy of a Chess Program
- Related Work
- Motivation and Goal
- Network Structure
- Move Matching
- Reinforcement by self-play
- Tree Search Enhancements
- Results
- Future Work

Anatomy of a Chess Program: Evaluation

How favorable is a position?

- Piece Value;
- Piece-Square table;
- Mobility (number of legal moves);
- Stage of the game…

-50	-40	-30	-30	-30	-30	-40	-50
-40	-20	0	0	0	0	-20	-40
-30	0	10	15	15	10	0	-30
-30	5	15	20	20	15	5	-30
-30	0	15	20	20	15	0	-30
-30	5	10	15	15	10	5	-30
-40	-20	0	5	5	0	-20	-40
-50	-40	-30	-30	-30	-30	-40	-50

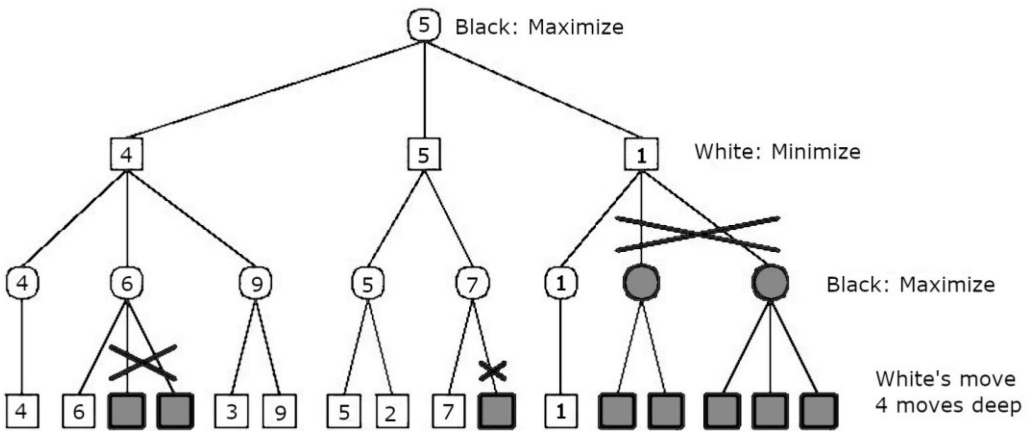
Piece-Square table for a knight

Usual piece values

Millions of parameters that need to be designed, tuned and combined.

➢ Very hard without expert knowledge !

Anatomy of a Chess Program: Search



Alpha-Beta: an optimized Minimax

➢ Depends on move ordering !

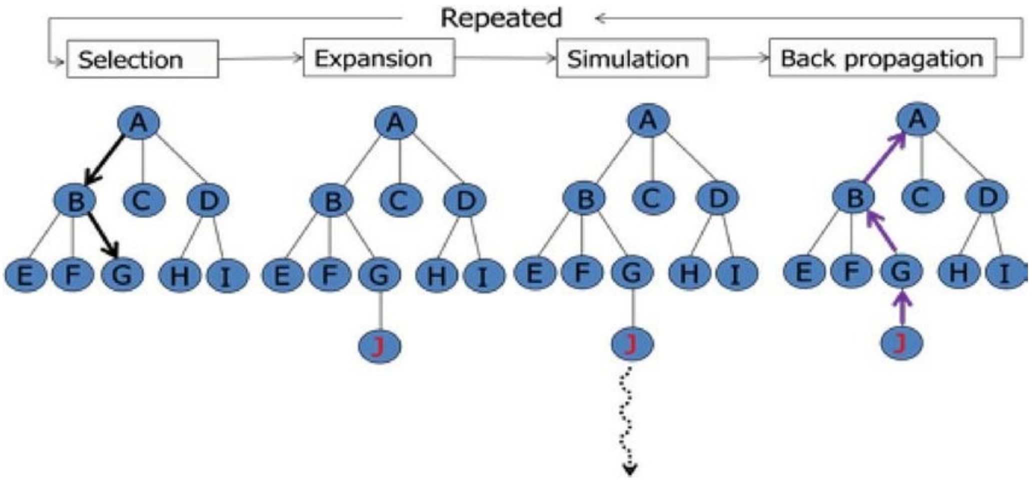
number of leaves with depth n and b = 40

depth n	b^n	$b^{\lceil n/2 \rceil} + b^{\lfloor n/2 \rfloor} - 1$
0	1	1
1	40	40
2	1,600	79
3	64,000	1,639
4	2,560,000	3,199
5	102,400,000	65,569
6	4,096,000,000	127,999
7	163,840,000,000	2,623,999
8	6,553,600,000,000	5,119,999

From https://www.chessprogramming.org/Alpha-Beta

Anatomy of a Chess Program: Search

Monte-Carlo Tree Search



➢ Need a good policy for simulation !

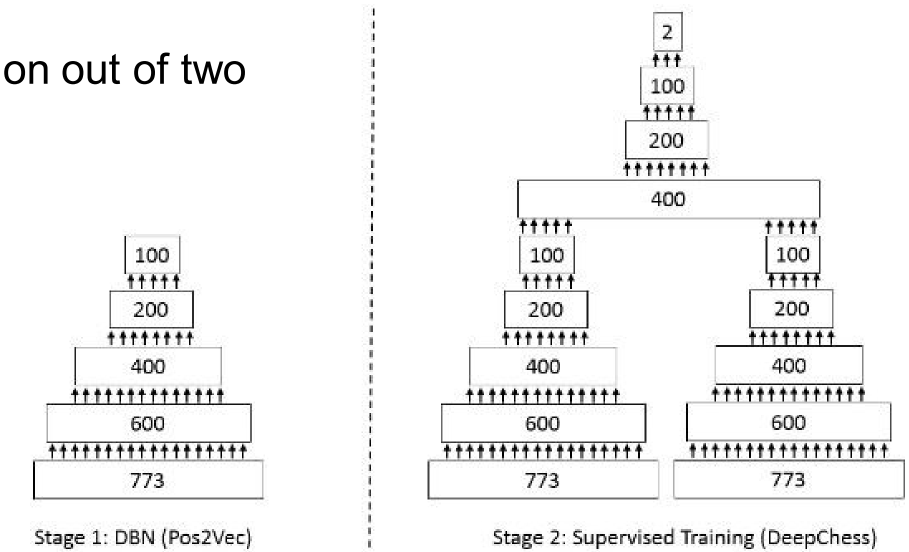
Related Work: DeepChess

DeepChess: End-to-End Deep Neural Network for Automatic Learning in Chess
Eli David, Nathan S. Nethanyahu, Lior Wolf - 2016

« DeepChess is the first end-to-end machine learning-based method that results in a grandmaster-level chess playing performance »

- ➢ Learns to find the most favorable position out of two
- ➢ Uses modified version of Alpha-Beta

Problem: can be improved by Network Distillation, but the search is very slow.

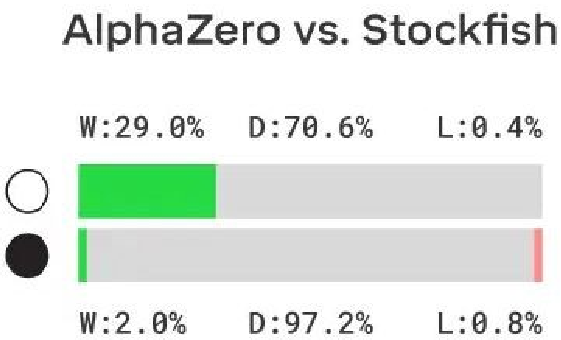


DeepChess' Architecture

Related Work: AlphaZero

Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm
David Silver et al. - 2017

- First model trained entirely through self-play.
- Beat Stockfish after 4 hours of training (on 5000 TPUs)
- Network Architecture is generalized to Go and Shogi, feature representation without game-specific knowledge



From https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

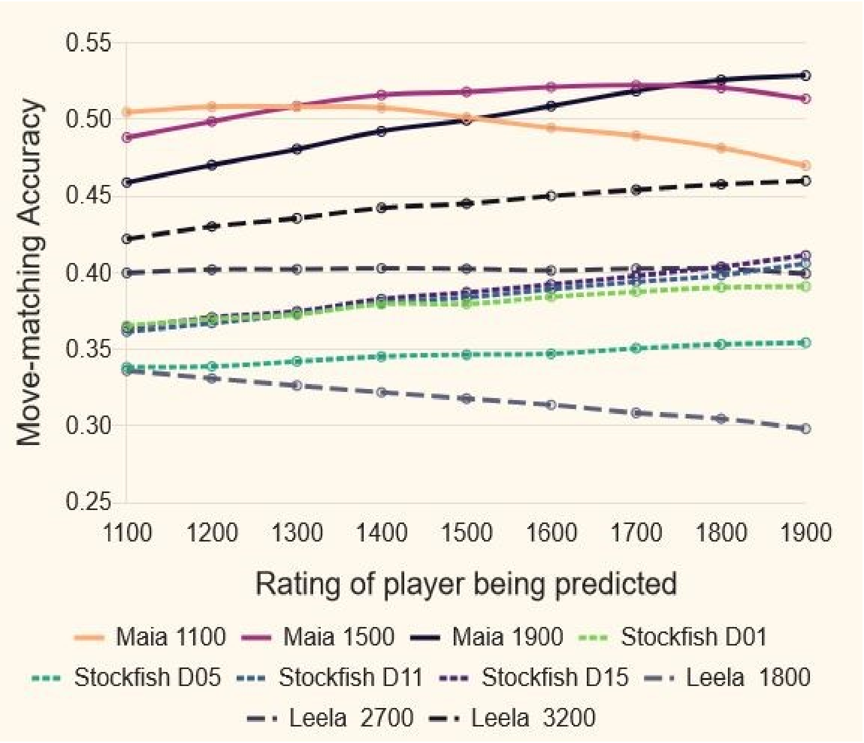
Related Work: Maia

Aligning Superhuman AI with Human Behavior: Chess as a Model System

Reid McIlroy-Young, Siddharta Sen, John Kleinberg, Ashton Anderson - 2020

AlphaZero's architecture, but with supervised learning.

- Aims at mimicking human players at a given Elo; also trained to predict mistakes.
- Can be tuned to a particular player with up to 65% accuracy



From https://maiachess.com/

Motivation and goals

Most engines before AlphaZero only train an evaluation function.

- ➢ Can we obtain a good engine by only learning a policy?

Using reinforcement learning takes a very long time.

- ➢ Leela Chess Zero took 3 years to replicate AlphaZero’s results.

On the opposite, supervised learning requires huge databases.

- ➢ Maia needs 12 million games per target level.

Goals

- ➢ Mix supervised and reinforcement learning to accelerate training.
- ➢ The obtained network should be able to serve as a training partner for a human player.

Network Structure: Features

41 8x8 feature planes:

- 3 last positions using each 13 planes
- Two planes for color and number of moves played

Position description: one plane per piece type and color (6x2), one for free squares

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Free squares plane

0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Black queen plane

Network Structure: Architecture

Base network:

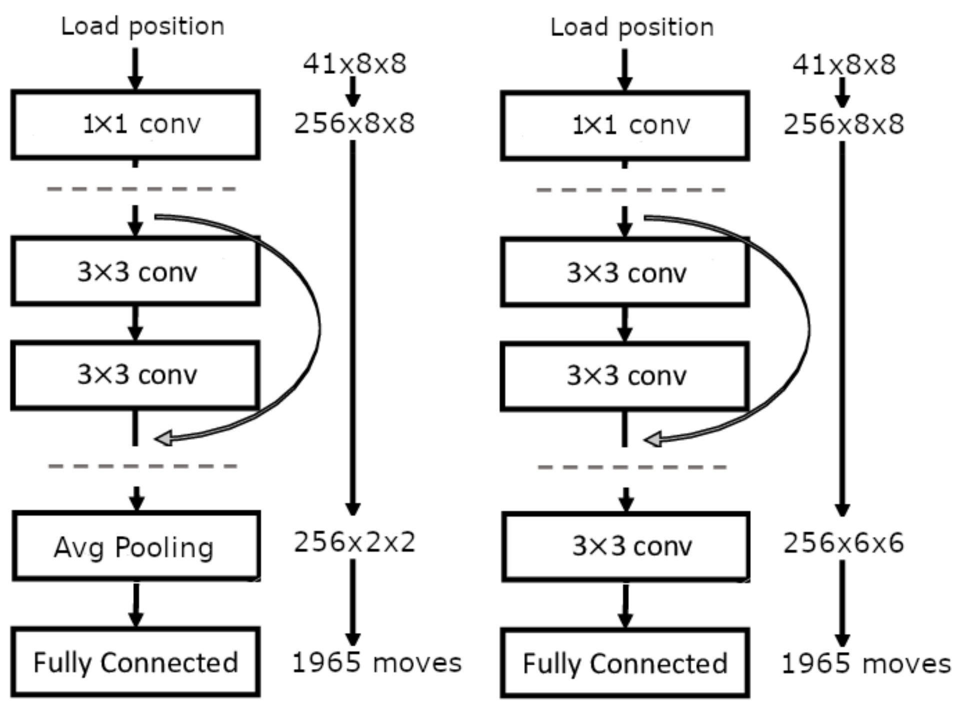
1. A 1x1 convolution for feature extraction

2. 3 residual blocks with 3x3 convolutions

Two slightly different heads for Move Matching and Self-Play.

➢ The residual tower is only trained through supervised learning.

➢ Self-Play focuses on extracting the best move



Architecture for Move Matching (left) and Self-Play(right)
Left: 6,5m weights / Right: 20m weights

Move Matching

We want to train the residual tower to extract a representation for move selection.

- Task: replicate the moves of human players with above 2200 points on Lichess (top 1%).
- Used to test different architectures and feature representation.

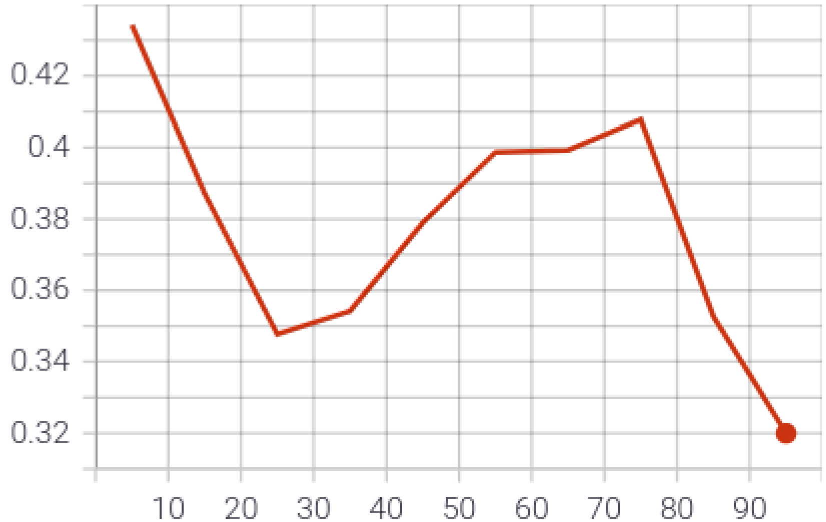
During training, the loss is computed only on legal moves, not all moves.

- We do not penalize rarely occuring moves like promotions

Move Matching Results

Accuracy of different architectures
(trained on 900k positions)

Method	Accuracy
Dense layers	0,24
Model	**0,38**
Convolution Layers	0,34
Model + Loss on all	0,36
Maia	0,35



Move-matching performance depending on the stage of the game

Self-Play: Ensure Exploration

Idea: use output of the network to choose the next move.

However, irrelevant moves tend to accumulate.

But we can use the database !

1. Select a game from the database.

2. Cut the last k moves and start self-play from this position.
   - Ensure that played positions happen in “real” games;
   - We can use the first moves as additional training examples;
   - We know the theoretical winner !

Self-Play: Rewards

		Reward for player	
Supposed Winner	Actual Result		
		1	$ignored$
		0	1
		1	0
		$ignored$	1
	Draw	$1/n$	1
	Draw	1	$1/n$

Tree Search Enhancements

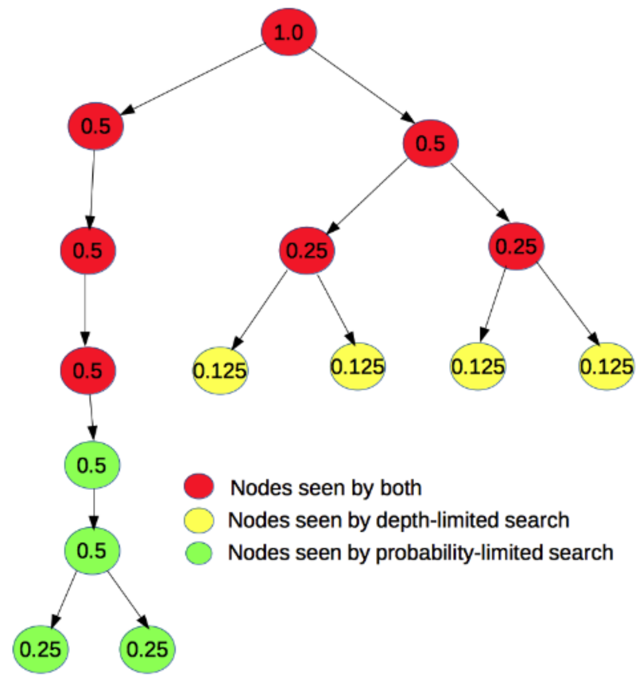
Idea: network output can be used to improve Alpha-Beta search.

- Improve search speed by move ordering
  - up to x10 speed compared to normal

Tree Search Enhancements

Idea: network output can be used to improve Alpha-Beta search.

- Improve search speed by move ordering
  - up to x10 speed compared to normal
- Probability search instead of depth search
  - Limit search to nodes with a high occurrence probability
  - Goes deeper with a similar amount of visited nodes
  - $p(n_k) = \prod_{i=0}^{k} m_i$



From Giraffe: Using deep reinforcement learning to play chess, 2015

Tree Search Enhancements

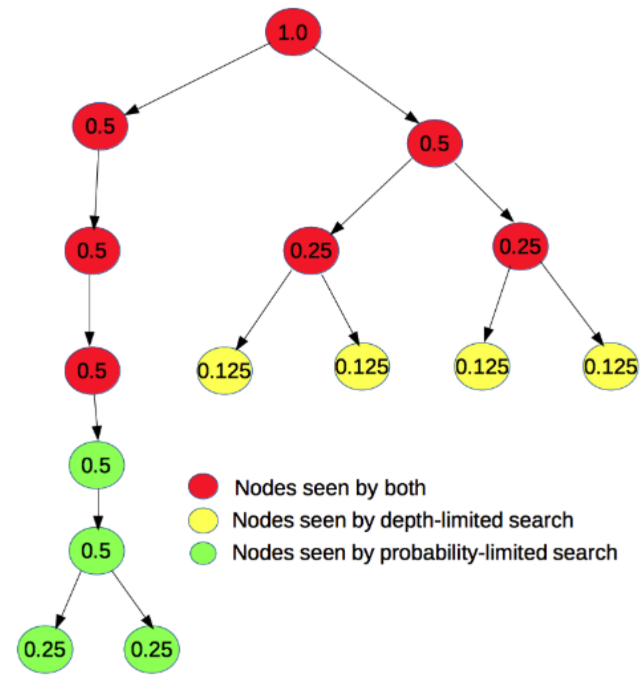
Idea: network output can be used to improve Alpha-Beta search.

- Improve search speed by move ordering
  - up to x10 speed compared to normal

- Probability search instead of depth search
  - Limit search to nodes with a high occurrence probability
  - Goes deeper with a similar amount of visited nodes
  - $p(n_k) = \prod_{i=0}^{k} m_i$

Pure Monte-Carlo search: simulate n games for each possible move

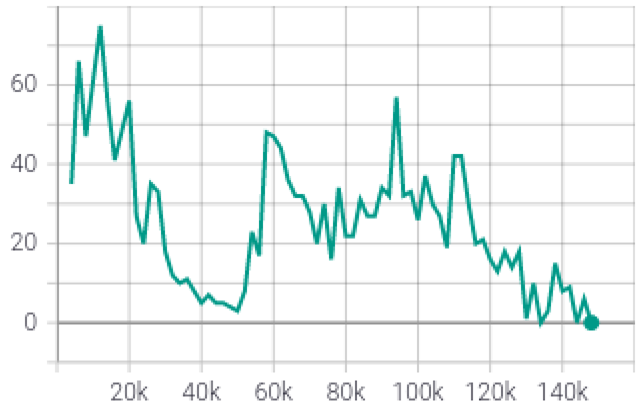
- Much faster than the above strategies !



From Giraffe: Using deep reinforcement learning to play chess, 2015

Results

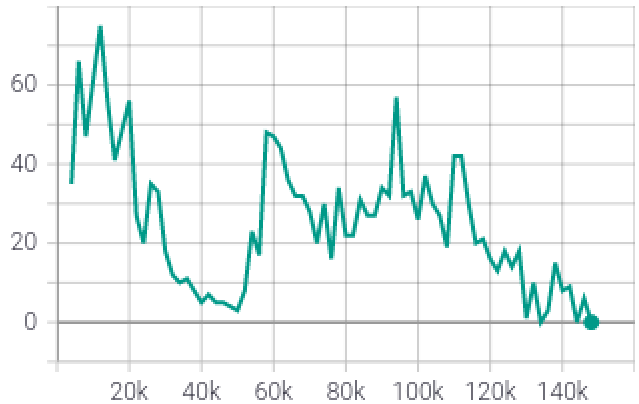
- The network quickly learns how to shorten a game.



Number of aborted games (length > 300 moves, then 200 after 50k steps) relative to number of training games

Results

- The network quickly learns how to shorten a game.



Number of aborted games (length > 350 moves, then 250 after 50k steps) relative to number of training games

- Winrate over 100 games against différents versions:
  - ➢ Up to +220 Elo with MCTS 40 !

-	AB	Base
AB search	-	32
Base	68	-
Prob search	73.5	57

-	Base
MCTS 3	71
MCTS 5	76
MCTS 10	77
MCTS 40	78

Future Work

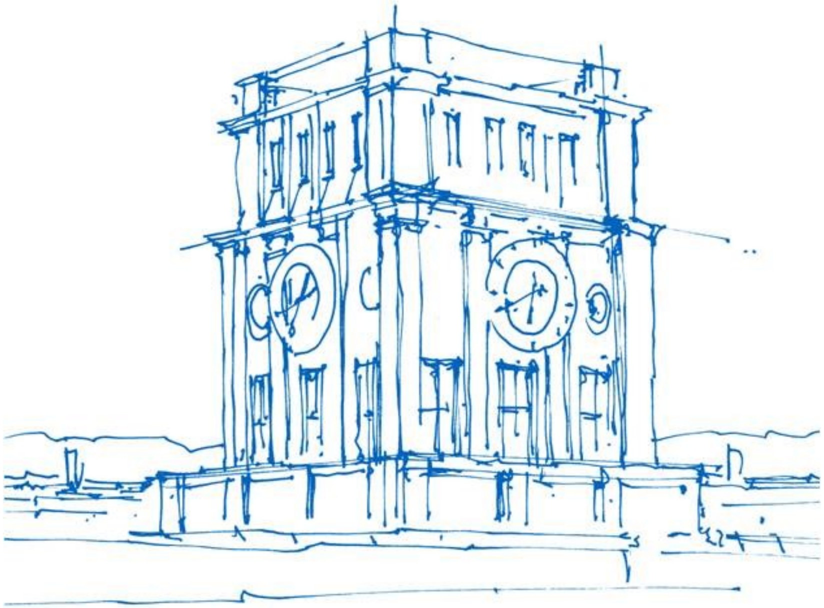
Implement a time control strategy.

- Using searches (except MCTS) takes a very long time
- Using the bare model or MCTS with a small parameter is very fast, and very unhuman

Train the network using the results of MCTS searches

- Can aggregate many moves in a single step
- How would it fare in terms of training speed?

Thank you for your attention !



Uhrenturm der TUM