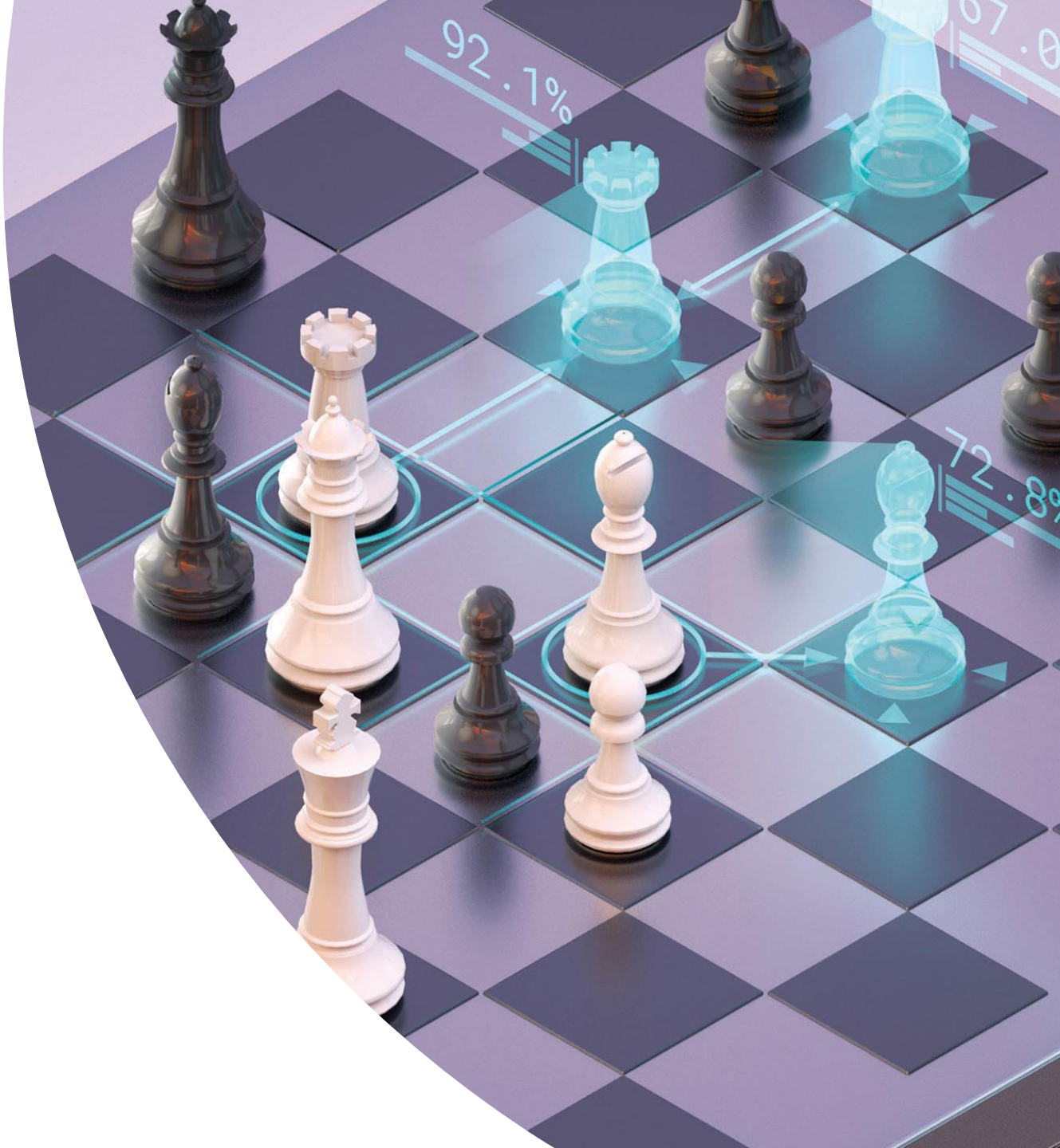
AlphaZero

The new Chess King

How a general reinforcement learning algorithm became the world's strongest chess engine after 9 hours of self-play

By Rune Djurhuus (Chess Grandmaster and Senior Software Engineer at Microsoft Development Center Norway) – Rune.Durhuus@microsoft.com

Presented October 17, 2019 at Department of Informatics at University of Oslo



Sources

- AlphaZero creator DeepMind: https://deepmind.com/blog/alphazero-shedding-new-light-grand-games-chess-shogi-and-go/
- Science Journal article of Dec 7, 2018: “A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play”
  - https://science.sciencemag.org/content/362/6419/1140 / (behind pay wall)
  - Open access version of Science paper (PDF format)
- The book «Game Changer», by Matthew Sadler and Natasha Regan (New in Chess, 2019): https://www.newinchess.com/game-changer
- Leela Chess Zero: https://en.wikipedia.org/wiki/Leela_Chess_Zero
  - free, open-source, neural network based chess engine that beat Stockfish 53,5 – 46,5 (+14 -7 =79) in the Superfinal of season 15 of Top Chess Engine Championship (TCEC) in May 2019.
- Chess Programming Wiki: https://www.chessprogramming.org/Main_Page

Complexity of a Chess Game

- **20** possible start moves, **20** possible replies, etc.
- **400** possible positions after **2** ply (half moves)
- **197 281** positions after **4** ply
- 7^{13} positions after 10 ply (5 White moves and 5 Black moves)
- **Exponential explosion**!
- Approximately **40 legal moves** in a typical position
- About $\mathbf{10^{120}}$ possible chess games and $\mathbf{10^{47}}$ different chess positions

Solving Chess, is it a myth?

Chess Complexity Space

- The estimated number of possible chess games is 10^{120}
  - Claude E. Shannon
  - 1 followed by 120 zeroes!!!
- The estimated number of reachable chess positions is 10^{47}
  - Shirish Chinchalkar, 1996
- Modern GPU's performs 10^{13} flops
- If we assume one million GPUs with 10 flops per position we can calculate 10^{18} positions per second
- It will take us 1 600 000 000 000 000 000 000 years to solve chess

Assuming Moore's law works in the future

- Todays top supercomputers delivers 10^{16} flops
- Assuming 100 operations per position yields 10^{14} positions per second
- Doing retrograde analysis on supercomputers for 4 months we can calculate 10^{21} positions.
- When will Moore's law allow us to reach 10^{47} positions?
- Answer: in 128 years, or around year 2142!

http://chessgpgpu.blogspot.no/2013/06/solving-chess-facts-and-fiction.html

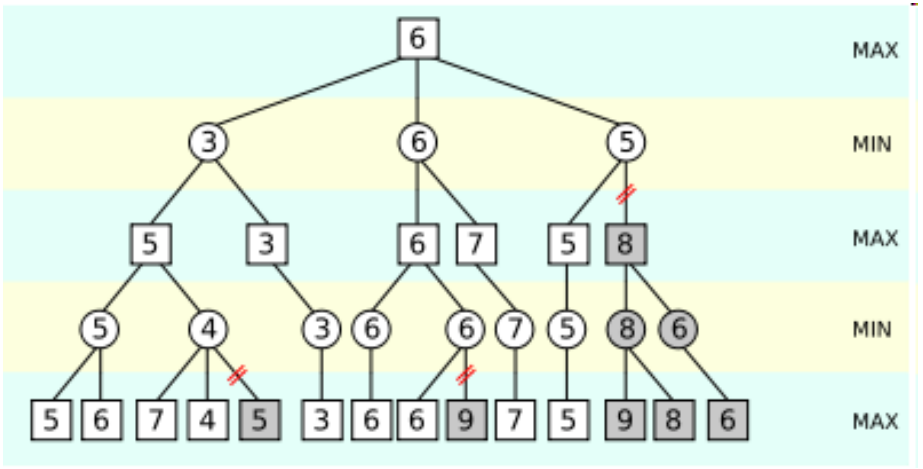
History of Computer Chess

- Chess is a good fit for computers: **Clearly defined rules, Game of complete information, Easy to evaluate (judge) positions, Search tree is not too small or too big**
- 1950: Programming a Computer for Playing Chess (Claude Shannon)
- 1951: First chess playing program (on paper) (Alan Turing)
- 1958: First computer program that can play a complete chess game
- 1981: Cray Blitz wins a tournament in Mississippi and achieves master rating
- 1989: Deep Thought loses 0-2 against World Champion Garry Kasparov
- 1996: Deep Blue wins a game against Kasparov, but loses match 2-4
- 1997: Upgraded Dee Blue wins 3.5-2.5 against Kasparov
- 2005: Hydra destroys GM Michael Adams 5.5-0.5
- 2006: World Champion Vladimir Kramnik looses 2-4 against Deep Fritz (PC chess engine)
- 2014: Magnus Carlsen launches “Play Magnus “ app on iOS where anyone can play against a chess engine that emulates the World Champion’s play at different ages
- 2017: AlphaZero beats world champion program Stockfish 64-34 without losing a game after learning chess from scratch by 9 hours of self-playing
- 2019: Leela Chess Zero beats Stockfish 53,5-46,5 in TCEC season 15 superfinal

Traditional Chess Engines

Traditional chess engines (including world computer chess champion Stockfish):

- Highly optimized **alpha-beta search algorithm**
- Striving for an optimal **move ordering** (analyze the best move first) in order to prune the search tree the most
- **Linear evaluation function** (of chess positions) with carefully tuned weights for a myriad of positional and dynamic features.
- Final evaluation of root node corresponds to score of leaf node in the **principal variation** (PV) – consisting of best moves from White and Black

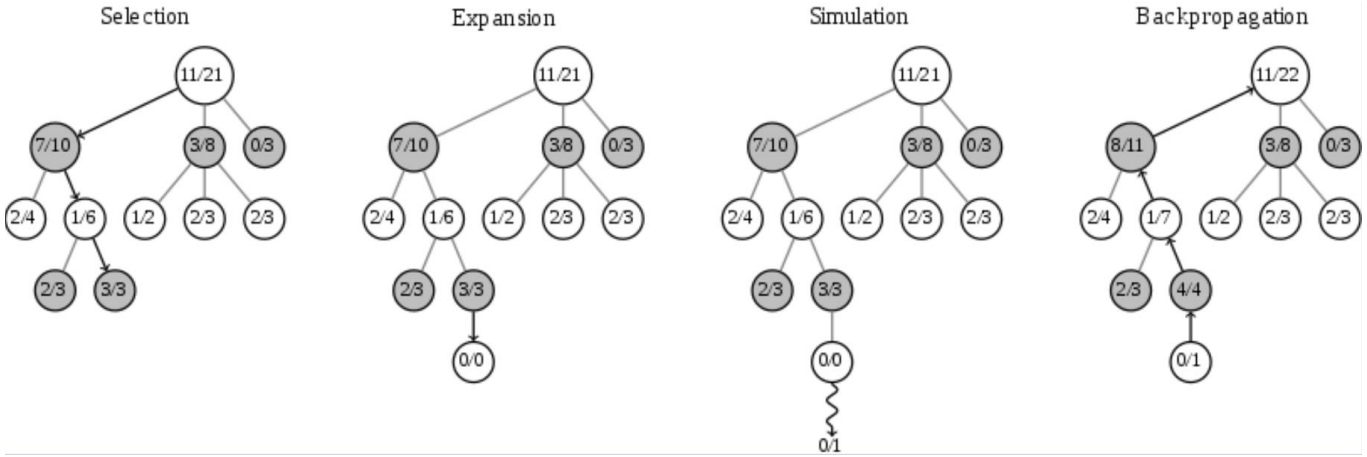


AlphaZero in two Sentences

- AlphaZero uses **Monte Carlo tree search** (MCTS) in combination with a **policy network** (for move probabilities) and a **value network** (for evaluating a position).
- Starting from **random play**, and given **no domain knowledge** except the game rules, AlphaZero by **self-playing** was able within 24 hours to train its neural networks up to **superhuman** level of play in the games of **chess** and **shogi** (Japanese chess) as well as **Go**, and convincingly defeated a world-champion program in each case.

AlphaZero

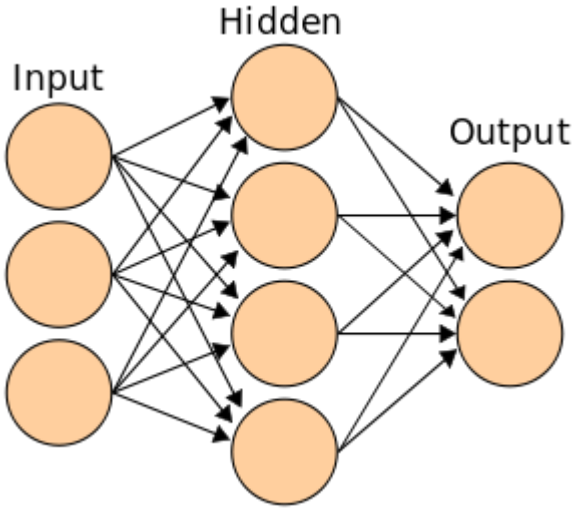
- Monte Carlo tree search (**MCTS**) algorithm (instead of alpha-beta)
- **Deep neural network** for move probabilities (policy network) and position evaluation (value network, with 8x8 numeric grid) (instead of linear, hand-crafted evaluation function)
- **Reinforcement learning algorithm**, starting from scratch ("tabula rasa")
- **No domain knowledge** beyond the basic chess rules (how the pieces moves, size of board, etc)



- The focus of Monte Carlo tree search is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space.
- AlphaZero uses playouts of games during self-training, but not during match play.

Deep Neural Network

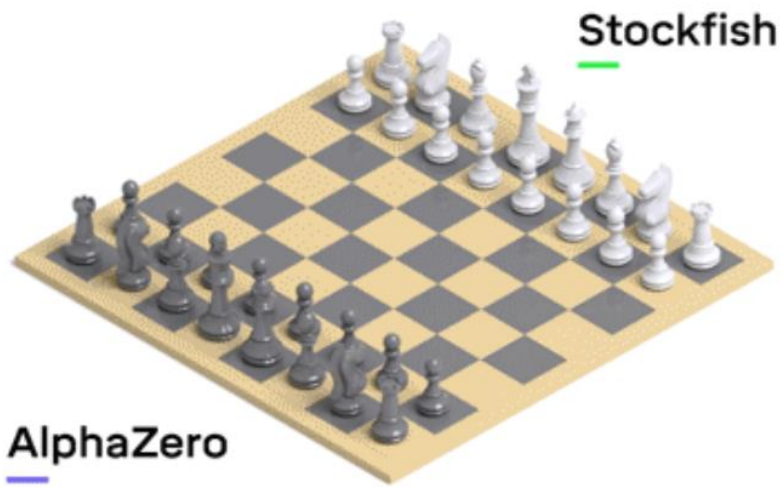
- (Artificial) Neural Network is a type of graphs inspired by the human brain, with signals flowing from a set of input nodes to a set up output nodes.
- The nodes ("artificial neurons") of one or more (hidden) layers receive one or more inputs, and after being weighted, sum them to produce an output.
- The sum is passed through a nonlinear function known as an activation function.
- A deep neural network (DNN) is a neural network with multiple layers (>2, but could be 100's) between the input and output layers.
- Alpha Zero uses one DNN for finding candidate moves (policy network) and one DNN for evaluating a chess position (value network).

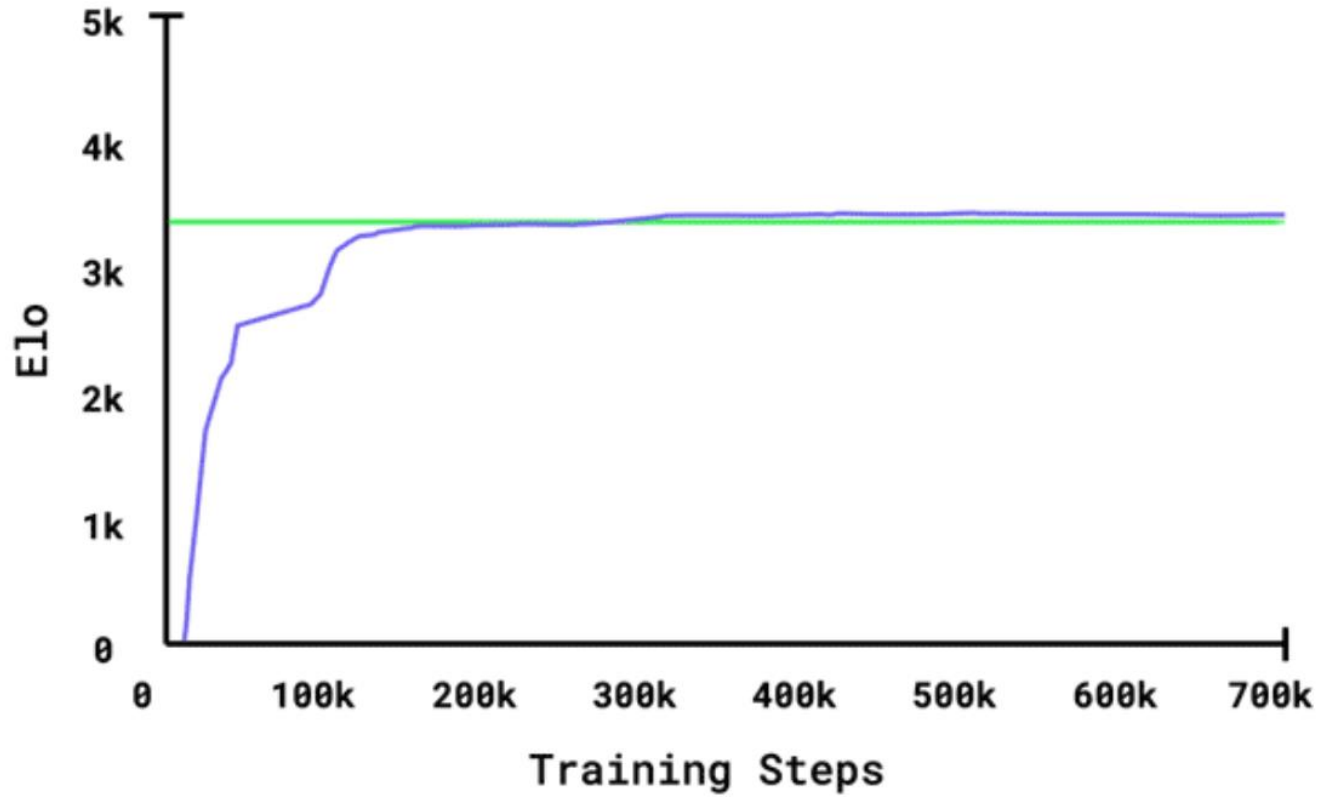


Reinforcement learning

- To learn each game, an untrained neural network plays millions of games against itself via a process of trial and error called **reinforcement learning**.
- For each move during self-play MCTS performed 800 simulations, each extending the search by one move while assessing the value of the resulting position.
- At first, it plays completely randomly.
  - Too avoid endless random games, they were stopped after X moves and judged a draw.
  - Now and then some random game would result in a win or loss.
- Over time the system learns from wins, losses, and draws to adjust the parameters of the neural network, making it more likely to choose advantageous moves in the future.
- Positions occurring during a won (lost) game is adjusted positively (negatively) in value network
- After a won game, connections in policy network are strengthen for moves recommended (and played) by AlphaZero.
- During 9 hours of self play, AlphaZero played 44 millions games against itself (1K games / sec).
- Training for a longer period gave diminishing return, probably due to the large number of draws (>90%) that started to occur during self-play.

Chess



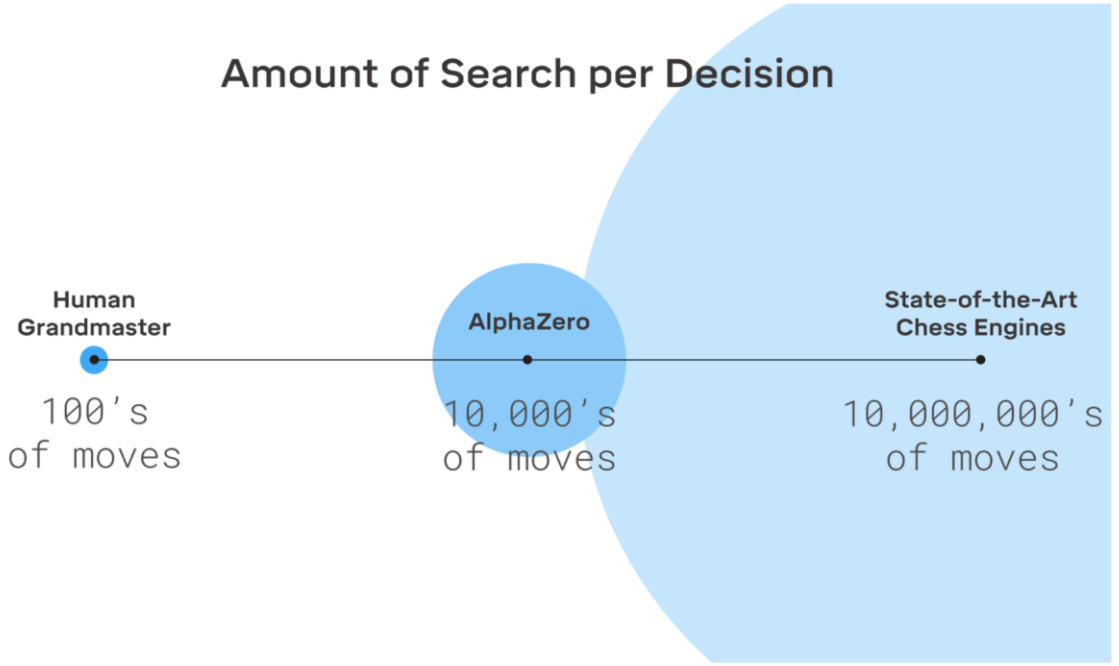


In chess, AlphaZero first outperformed Stockfish after just 4 hours; in shogi, AlphaZero first outperformed Elmo after 2 hours; and in Go, AlphaZero first outperformed the version of AlphaGo that beat the legendary player Lee Sedol in 2016 after 30 hours.
Note: each training step represents 4,096 board positions.

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

During Match Play

- The trained policy and value networks are used to guide MCTS to select the most promising moves in games.
- For each move, AlphaZero searches only a small fraction of the positions considered by traditional chess engines.
- It searches only 60 thousand positions per second, compared to roughly 60 million for Stockfish



Monte Carlo Tree Search Example

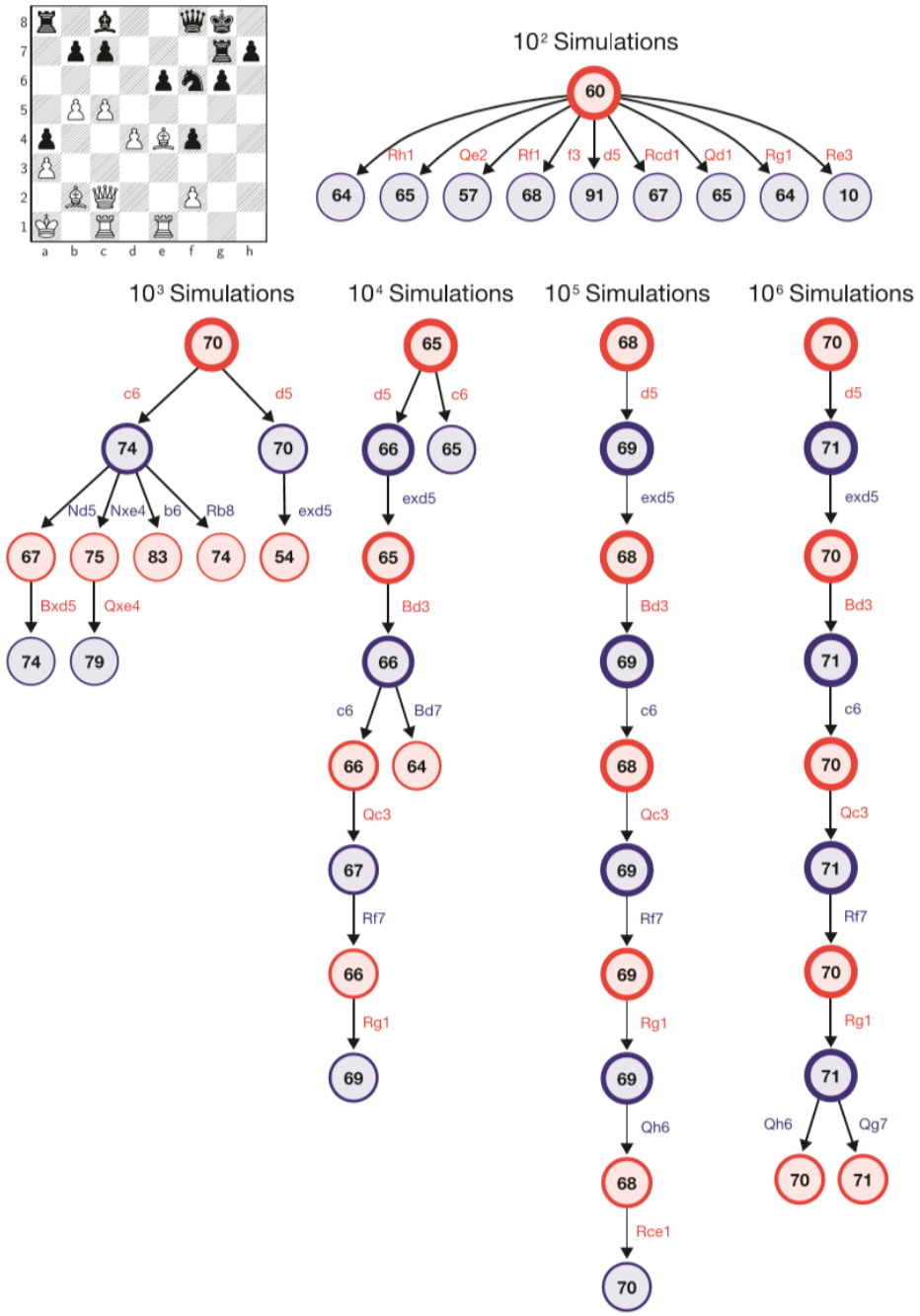
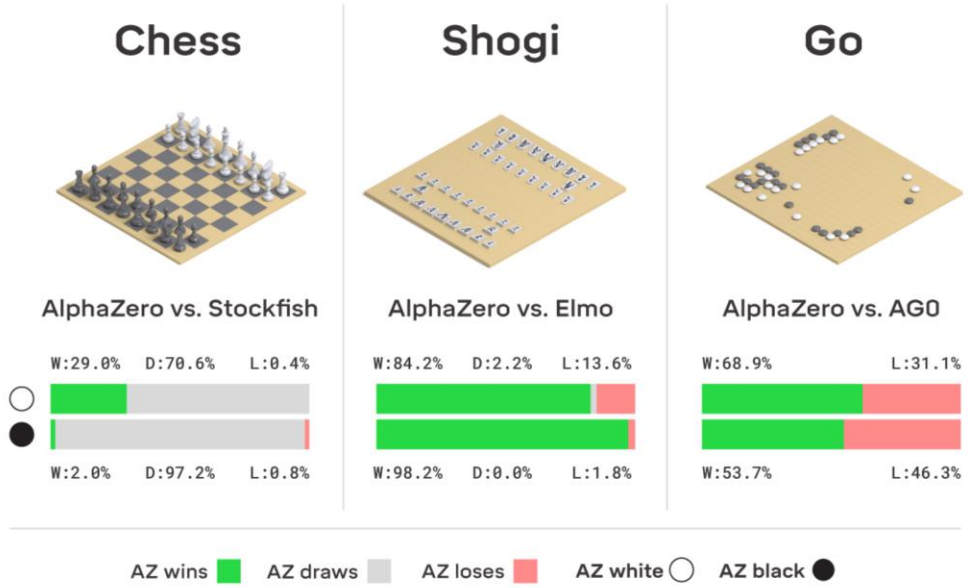


Figure 4: **AlphaZero's search procedure.** The search is illustrated for a position (inset) from game 1 (table S6) between AlphaZero (white) and Stockfish (black) after 29. ... Qf8. The internal state of AlphaZero's MCTS is summarized after $10^2, ..., 10^6$ simulations. Each summary shows the 10 most visited states. The estimated value is shown in each state, from white's perspective, scaled to the range $[0, 100]$. The visit count of each state, relative to the root state of that tree, is proportional to the thickness of the border circle. AlphaZero considers 30. c6 but eventually plays 30. d5.

(From page 9 of [open access version](#) of Science Journal paper "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play")

Squaring off against Stockfish

- The fully trained AlphaZero were tested against Stockfish which is considered the strongest commercial chess engine.
- Stockfish used 44 CPU cores whereas AlphaZero used a single machine with 4 first-generation **TPUs** and 44 CPU cores. (A first generation TPU is roughly similar in inference speed to commodity hardware such as an **NVIDIA Titan V GPU**, although the architectures are not directly comparable.)
- All matches were played using time controls of three hours per game, plus an additional 15 seconds for each move.
- AlphaZero convincingly defeated 2016 edition of Stockfish, winning 155 games and losing just six games out of 1,000.
- Recently (May 2019) Leela Chess Zero, a free, open-source neural network based chess engine, beat Stockfish 53,5 – 46,5 (+14 -7 =79) in the Superfinal of season 15 of Top Chess Engine Championship (TCEC).
- Very recently (October 2019) Stockfish fought back and beat AllieStein (which uses Leela's network) 54,5-45,5 in the Superfinal of TCEC season 16.



Non-Linear vs Linear Eval Function

- A non-linear, deep neural network provides a more powerful evaluation function, but may also introduce larger worst-case generalization errors then a linear function.
- When combining a non-linear eval function with alpha-beta search which computes an explicit minimax, the biggest errors in eval function are typically propagated directly to the root of the subtree (Principle Variation - PV).
- By contrast, AlphaZero's MCTS averages over the position evaluations within a subtree, rather than computing the minimax evaluation of that subtree.
- Authors of AlphaZero suggest that the approximation errors introduced by neural networks might tend to cancel out when evaluating a large subtree using MCTS.

Non-Linear vs Linear Eval Function

Two examples where Stockfish's handcrafted linear evaluation function was outperformed by AlphaZero's non-linear deep neural net:

- AlpaZero grasps when to occupy an open line with a rook, and when to ignore the open line
- AlphaZero has superb understanding of piece activity (mobility) – for both itself and the opponent – and how piece activity can compensate for material deficit

AlphaZero Playing Style

- AlphaZero independently discovered and played common human motifs during its self-play training such as **openings**, **king safety**, **pawn structure** and **piece mobility**.
- Being self-taught and therefore unconstrained by conventional wisdom about the game (including any pre-defined **value** of each **chess piece**), it developed its own **intuition** and **strategies** (e.g. restrict opponents king by advancing with rook's pawn up the board).
- Since AlphaZero takes an average of assessed outcomes as the value of the root node (instead of PV), it has a "**human**" way of playing: *"It looks like I will have all the chances here – there must be something good"*, without calculating all the way to checkmate.

AlphaZero Playing Style

“ Traditional engines are exceptionally strong and make few obvious mistakes, but can drift when faced with positions with no concrete and calculable solution. It's precisely in such positions where ‘feeling’, ‘insight’ or ‘intuition’ is required that AlphaZero comes into its own.“ (Matthew Sadler, author of “Game Changer” (New in Chess, 2019).

- AlphaZero likes to attack the opponent’s king (1 open line + 1 open diagonal + restricted king > 1 pawn). It was not constraint by conventional view on materiel: bishop = 3 pawns, rook = 5 pawns, etc, but rather combining all materiel and positional and dynamic features into a (probabilistic) expected score
- AlphaZero likes to keep its own king out of danger. AlphaZero makes sure the central situation is stable before launching a wing attack.
- AlphaZero loves attacking with opposite-colored bishops.

AlphaZero Game Examples

DEMOS

1. Attack, attack, attack (1 open line + 1 open diagonal + restricted king > 1 pawn)
2. Securing an outpost for the knight at all cost
3. Drive one or more opponent's pieces into passivity and then exchange off his remaining active piece(s)
4. Attacking with opposite color bishops
5. Restricting opponent's king (and handling of an open line with a pair of rooks on the board)

Microsoft is Hiring!

- If you're interested in a **full-time** position in **Europe** click here
  - *To be eligible for a **Full-Time position**, you must currently be graduating from a Bachelors, Masters or PhD program in 2020.*
- If you're interested in an **intern** position in **Europe** click here
  - *To be eligible for an **Internship**, you must currently be enrolled in a Bachelors, Masters or PhD program and have at least one term/semester of studies remaining.*
- *Deadline for applying is **Friday October 18, 2019***