Lecture 20
Game AI and AlphaGo Series

Bolei Zhou

The Chinese University of Hong Kong

Outline

- Background of Game AI
- Minimax and Monte-Carlo Tree Search
- AlphaGo Series:
  - AlphaGo (Jan. 2016)
  - AlphaGo Zero (Oct. 2017)
  - AlphaZero (Dec. 2017)
  - MuZero (Nov. 2019)

AI for Classic Board Games

- Simple rules but deep concepts
- Studied for hundreds or thousands of years
- Board game is a way to measure the ‘intelligence’
- Games are fun




A brief history of AI Game-playing

1944: Alan Turing and Donald Michie simulate by hand their chess algorithms during lunches at Bletchley Park

1959: Arthur Samuel's checkers algorithm (machine learning)

1961: Michie's Matchbox Educable Noughts And Crosses Engine (MENACE)

1991: Computer solves chess endgame thought draw: KRB beats KNN (223 moves)

1992: TD Gammon trains for Backgammon by self-play reinforcement learning

1997: Computers best in world at Chess (Deep Blue beats Kasparov)

2007: Checkers "solved" by computer (guaranteed optimal play)

2016: Computers best at Go (AlphaGo beats Lee Sodol)

2017: AlphaZero extends AlphaGo to best at go, chess, shogi

AI for Classic Board Games

- Previously, state of the art for many games include chess was minimax search with alpha-beta pruning
- Most top-performing game-playing programs didn’t do learning
- Two-player zero-sum game




A brief history of game-playing for AI

1944: Alan Turing and Donald Michie simulate by hand their chess algorithms during lunches at Bletchley Park

1959: Arthur Samuel's checkers algorithm (machine learning)

1961: Michie's Matchbox Educable Noughts And Crosses Engine (MENACE)

1991: Computer solves chess endgame thought draw: KRB beats KNN (223 moves)

1992: TD Gammon trains for Backgammon by self-play reinforcement learning

1997: Computers best in world at Chess (Deep Blue beats Kasparov)

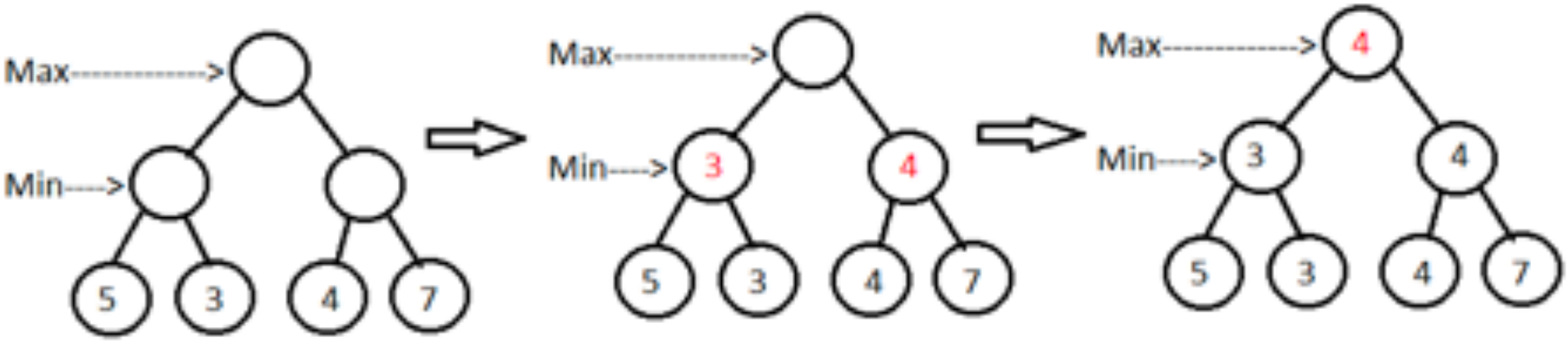
2007: Checkers "solved" by computer (guaranteed optimal play)

2016: Computers best at Go (AlphaGo beats Lee Sodol)

2017: AlphaZero extends AlphaGo to best at go, chess, shogi

Minimax Value

- Two players' decision on a tree
- The minimax value of the root is the utility of the outcome if both you and your opponent make optimal moves



Minimax as a Decision Rule

- A value function defines the expected total reward given joint policies

$$v_\pi(s) = \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \qquad \pi = \left\langle \pi^1, \pi^2 \right\rangle$$

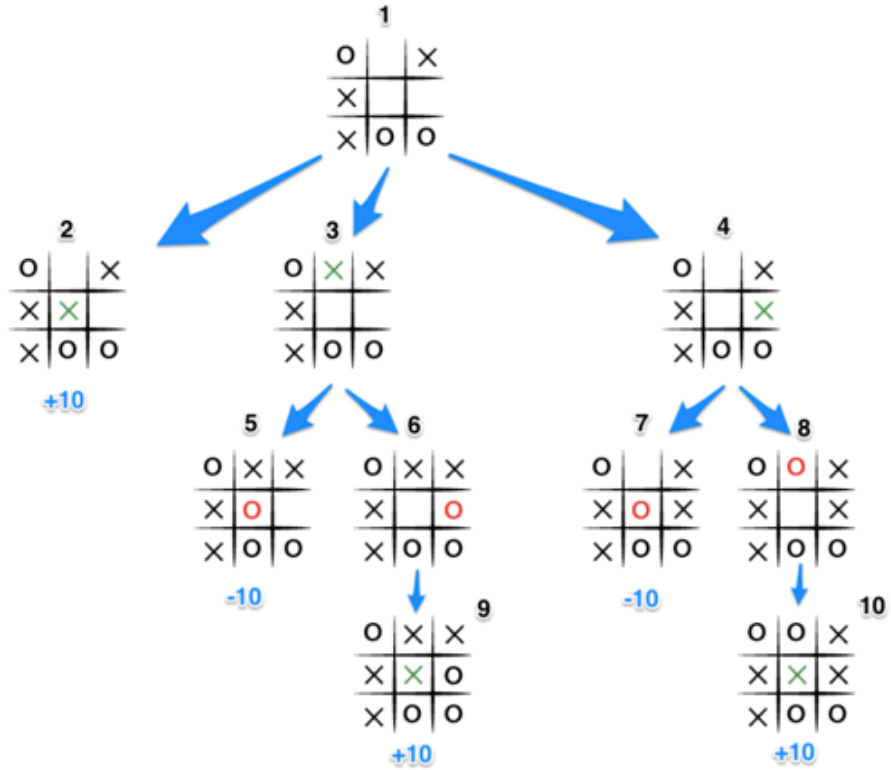
- A minimax value function maximizes white's expected return while minimizing black's expected return

$$v_*(s) = \max_{\pi^1} \min_{\pi^2} v_\pi(s)$$

- A minimax policy is a joint policy that achieves the minimax values
- A minimax policy is a Nash equilibrium

Minimax Search

- Minimax values can be found by depth-first game-tree search
- Game tree: tree representation of legal game states and moves
- Introduced by Claude Shannon: Programming a Computer for Playing Chess



http://perfecttictactoe.herokuapp.com/

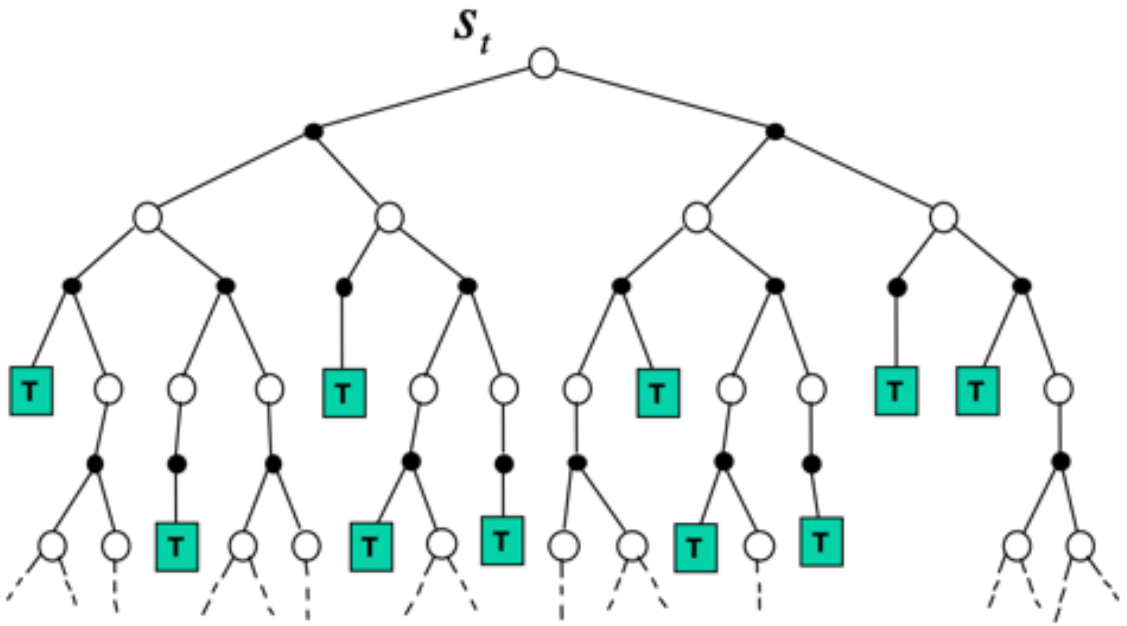
Value Function in Minimax Search

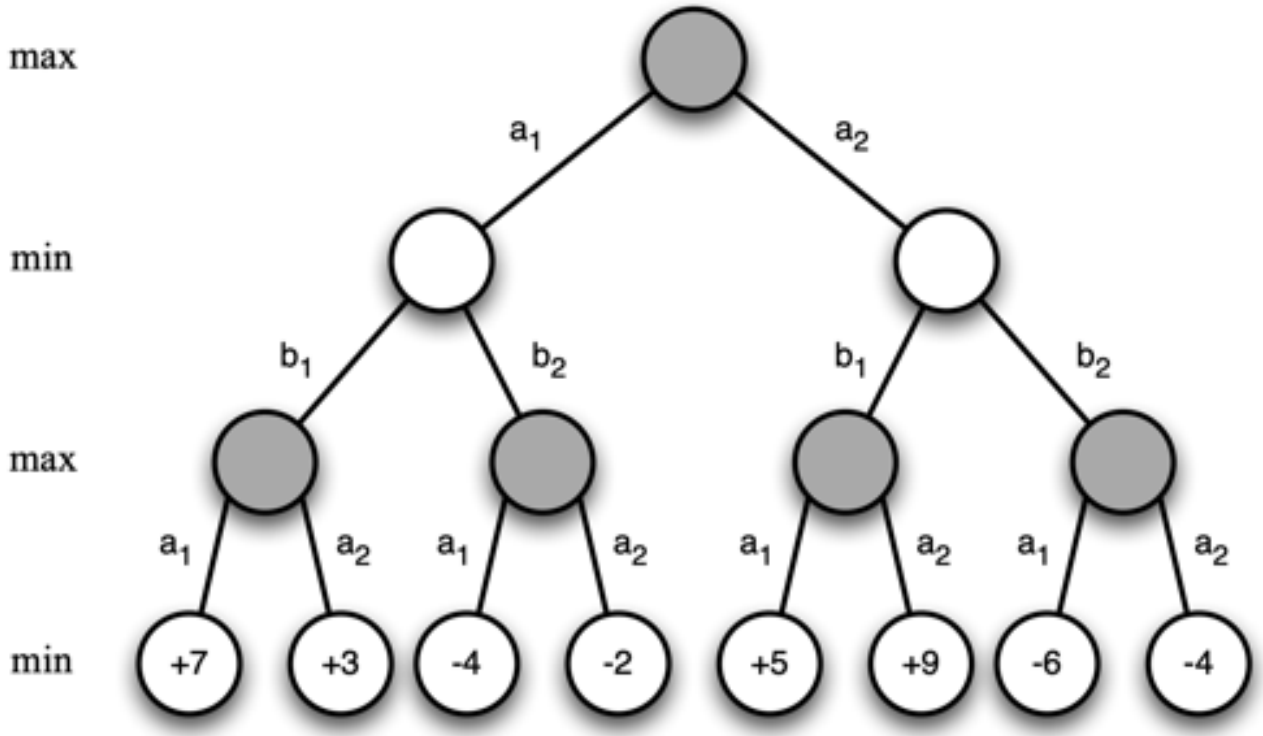
- Search tree grows exponentially with the branching factors
- Impractical to search to the end of the game
- Instead use value function approximator

$$v(s, \mathbf{w}) \approx v_*(s)$$

- Use value function to estimate minimax value at leaf nodes
- Minimax search run to fixed depth with respect to leaf values

Minimax Search for Value Iteration





Binary-Linear Value Function

- Binary feature vector x(s): e.g. one feature per piece
- Weight vector w: e.g. value of each piece
- Position is evaluated by summing weights of active features.

$$v(s, \mathbf{w}) = \mathbf{x}(s) \cdot \mathbf{w} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} +5 \\ +3 \\ +1 \\ -5 \\ -3 \\ -1 \\ \vdots \end{bmatrix}$$

$$v(s, \mathbf{w}) = 5 + 3 - 5 = 3$$

DeepBlue by IBM

Knowledge:

- 8000 handcrafted chess features
- Binary-linear value function
- Weights largely hand-tuned by human experts

Search

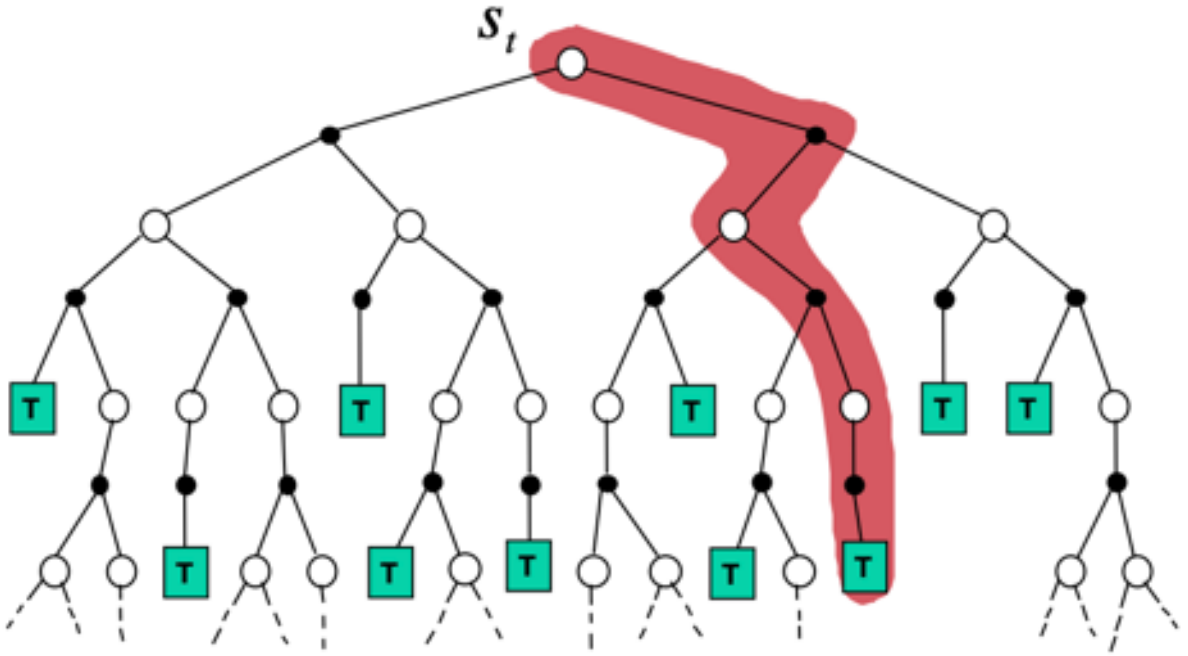
- High performance parallel alpha-beta search with minimax
- Searched 200 million positions/second

Results

- Defeated human champion Ksparov 4-2 (1997)
- Most watched event in internet history

Simulation-based Search

- Forward search paradigm using sample-based planning
- Simulate episodes of experience from now with the model
- Apply model-free RL to simulated episodes



Monte Carlo Search (Evaluation)

- Given a model $\mathcal{M}_\nu$
- Simulate K episodes from current state s_t using current simulation policy π

$$\{s_t, A_t^k, R_{t+1}^k, S_{t+1}^k, ..., S_T^k\}_{k=1}^K \sim \mathcal{M}_\nu, \pi$$

- Build a search tree containing visited states and actions
- Evaluate states $Q(s, a)$ by mean return of episodes from s, a

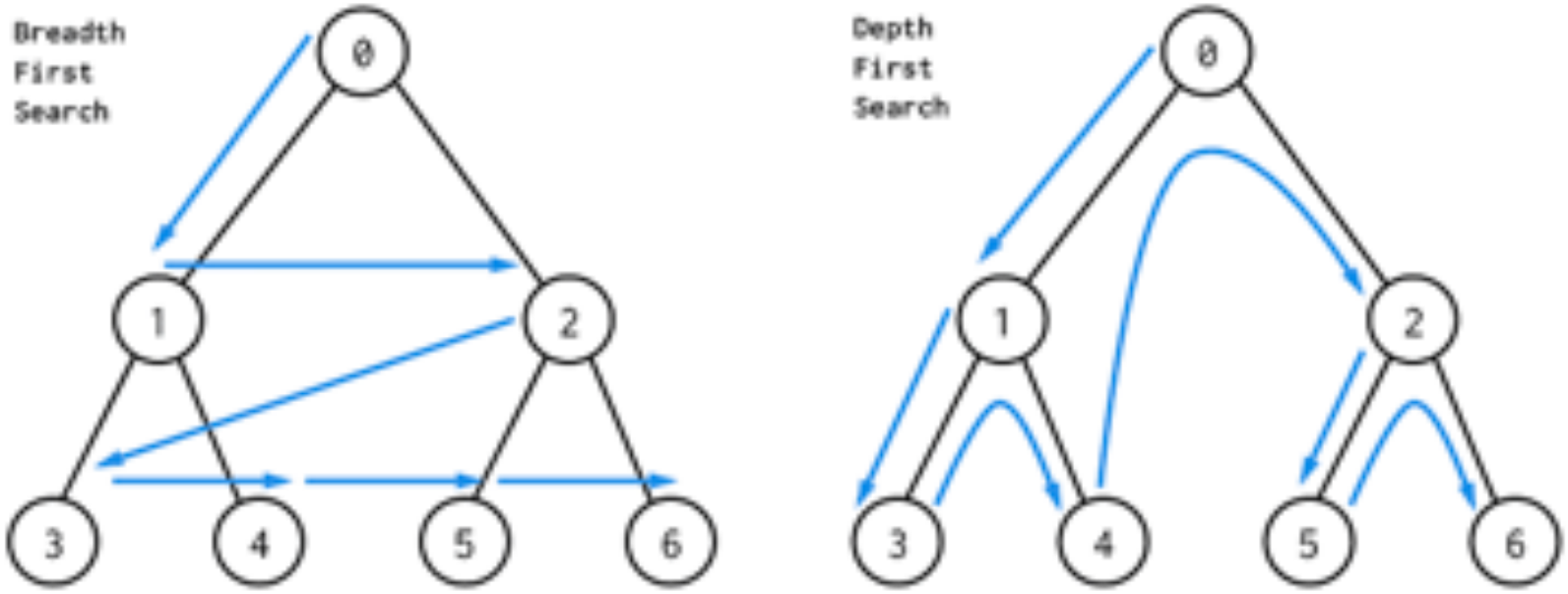
$$Q(s, a) = \frac{1}{N(s, a)} \sum_{k=1}^{K} \sum_{u=t}^{T} \mathbf{1}(S_u, A_u = s, a) G_u \xrightarrow{P} q_\pi(s, a)$$

- After search is finished, select current (real) action with maximum value in search tree

$$a_t = \underset{a \in \mathcal{A}}{\text{argmax}}\ Q(s_t, a)$$

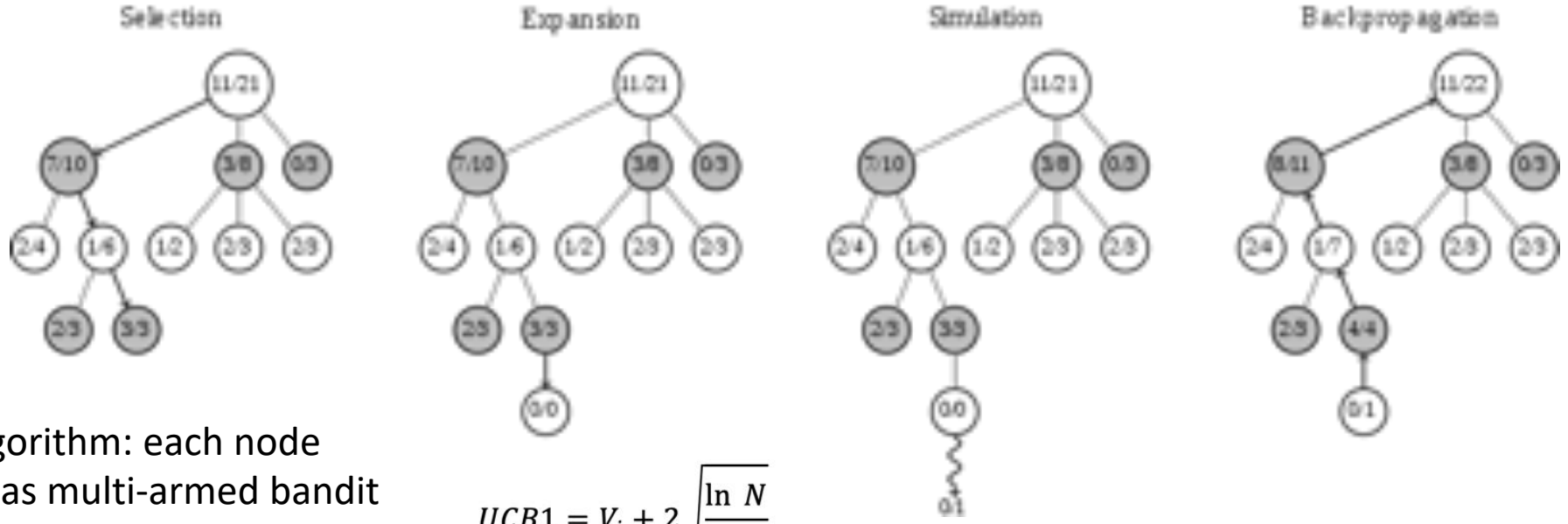
Tree Search

- Uninformed search: Depth first search (DFS) and Breadth first search (BFS)



Monte-Carlo Tree Search

- Iteration over 4 operations
- Simulation policy improves
  - Tree policy (improves): pick actions to maximize Q(S, A)
  - Default policy (fixed): pick actions randomly



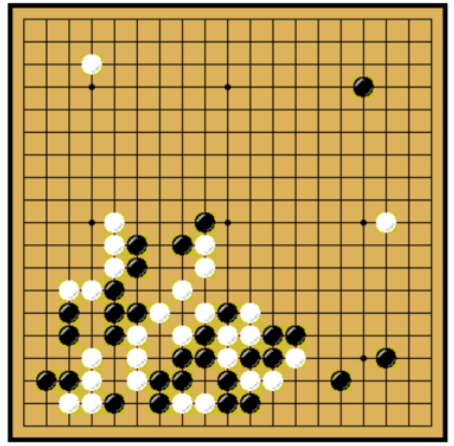
By UCB1 algorithm: each node considered as multi-armed bandit problem

$$UCB1 = V_i + 2\sqrt{\frac{\ln N}{n_i}}$$

epsilon-greedy

Game of Go as Grand Challenge for AI

- Intro
  - 2500 years old board game, and hardest classic board game
  - Grand challenge task for AI
  - Traditional game-tree search has failed in Go
- Basic rules
  - Two-player, deterministic turn-based game (Black and White)
  - Goal: place tiles to capture more territory and opponent pieces
- Properties:
  - Perfect information
  - Discrete state and action spaces
  - Translational/rotation invariance



Complexity of Go

- State-space complexity: 10^120 in Chess v.s 10^174 in Go

Position Evaluation in Go

- How good is a position s?
- Reward function

$$R_t = 0 \text{ for all non-terminal steps } t < T$$

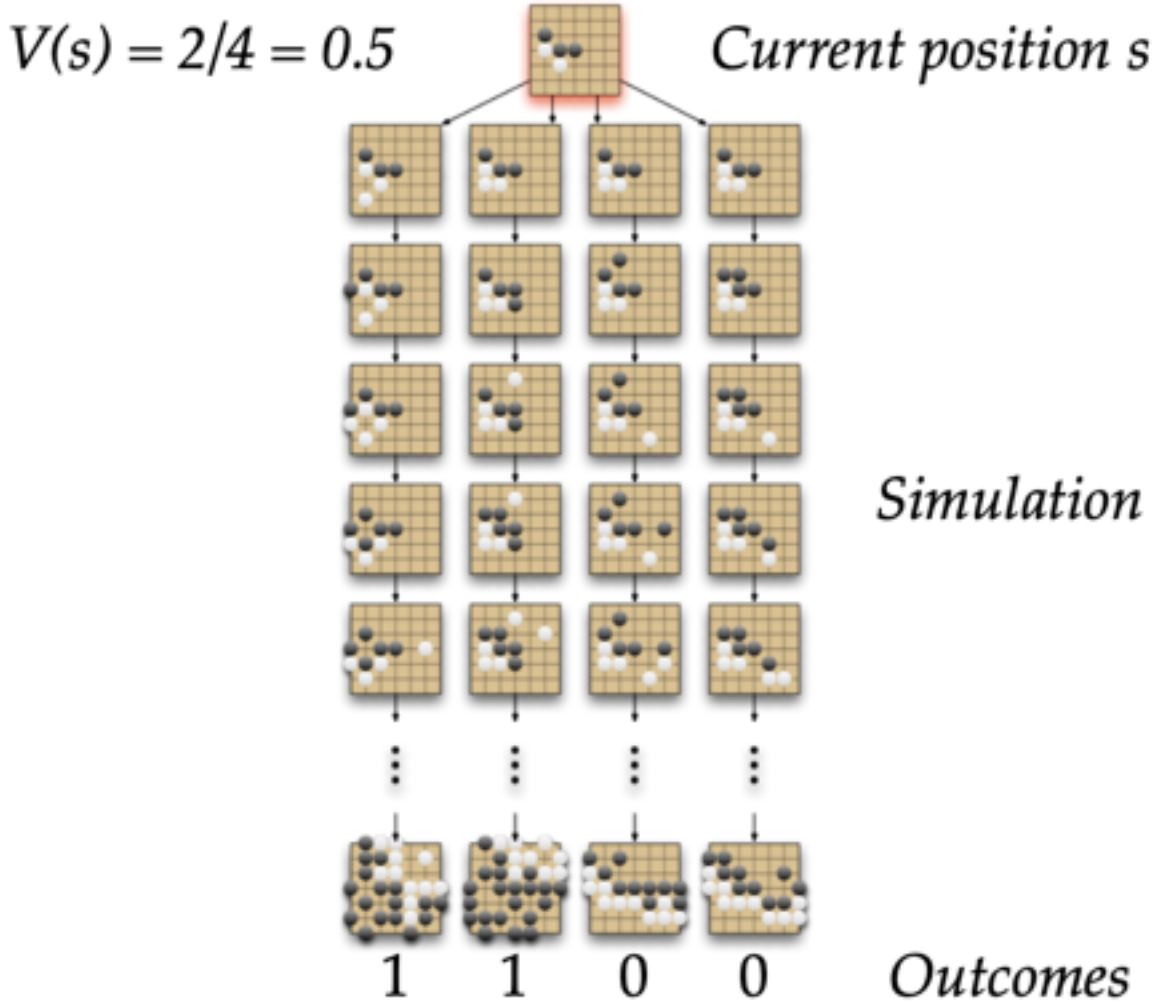
$$R_T = \begin{cases} 1 & \text{if Black wins} \\ 0 & \text{if White wins} \end{cases}$$

- Policy selects moves for both players: $\pi = \langle \pi_B, \pi_W \rangle$
- Value function (how good is position s)

$$v_\pi(s) = \mathbb{E}_\pi \left[R_T \mid S = s \right] = \mathbb{P} \left[\text{Black wins} \mid S = s \right]$$

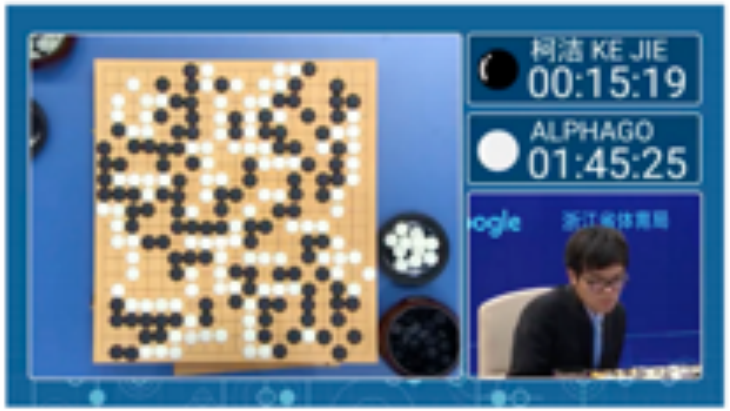
$$v_*(s) = \max_{\pi_B} \min_{\pi_W} v_\pi(s)$$

Monte-Carlo Evaluation in Go



AlphaGo Series

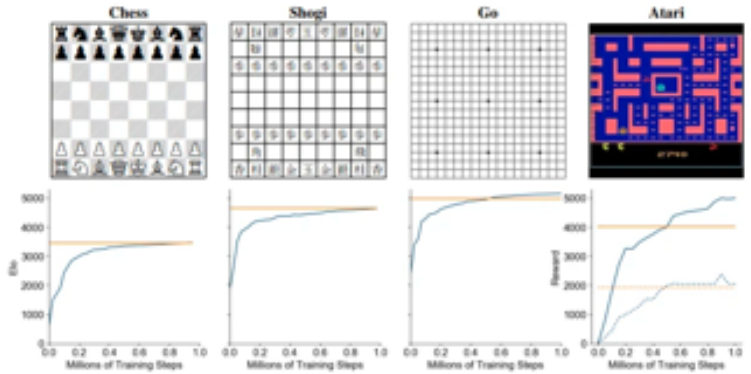
AlphaGo Lee
Mar. 2016



AlphaGo Master
May 2017

AlphaGo Zero
Oct. 2017

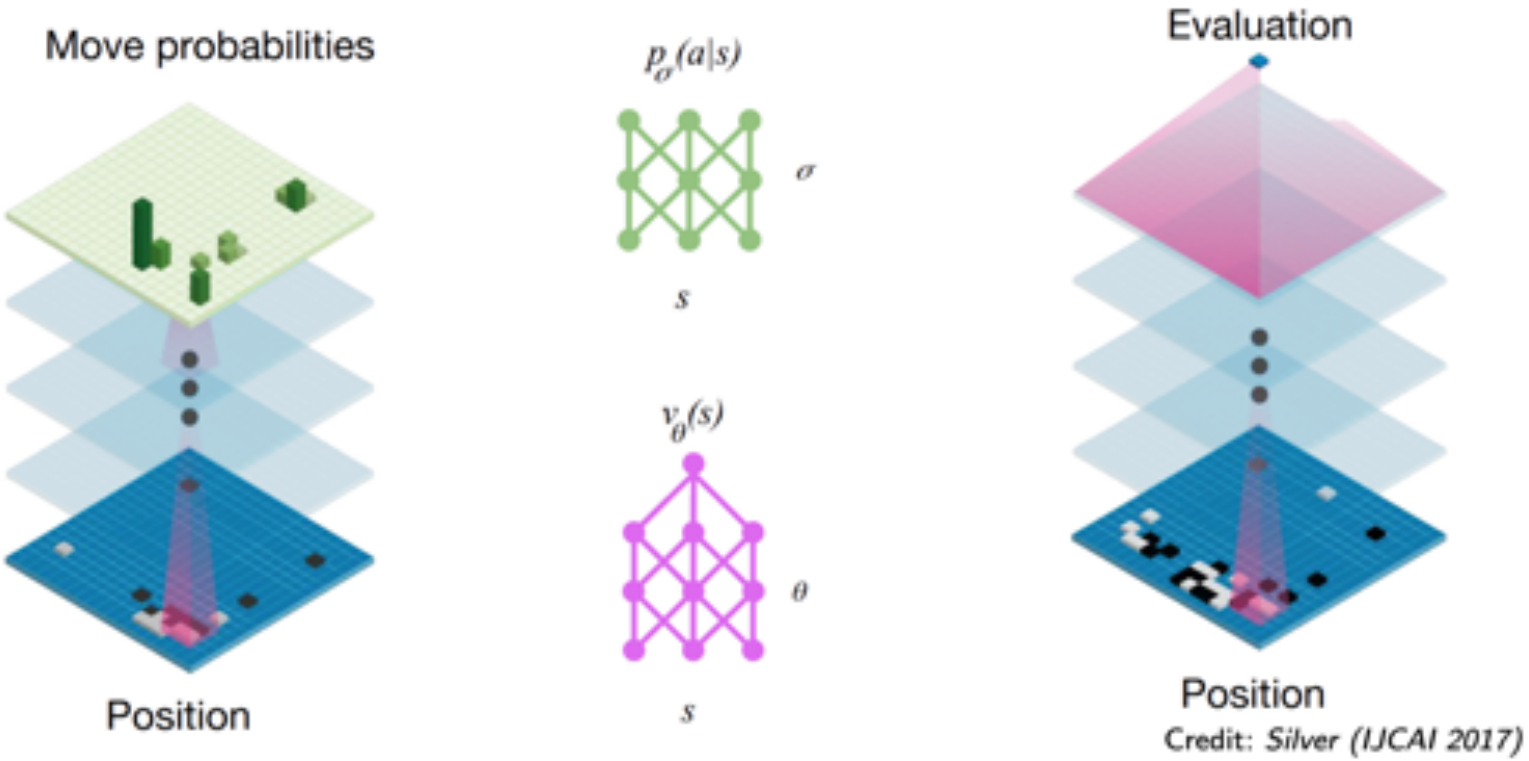
AlphaZero, Dec. 2017



MuZero, Nov. 2019

Reinforcement Learning by AlphaGo

- MCTS with RL self-play
- Separate policy and value networks



Reinforcement Learning by AlphaGo

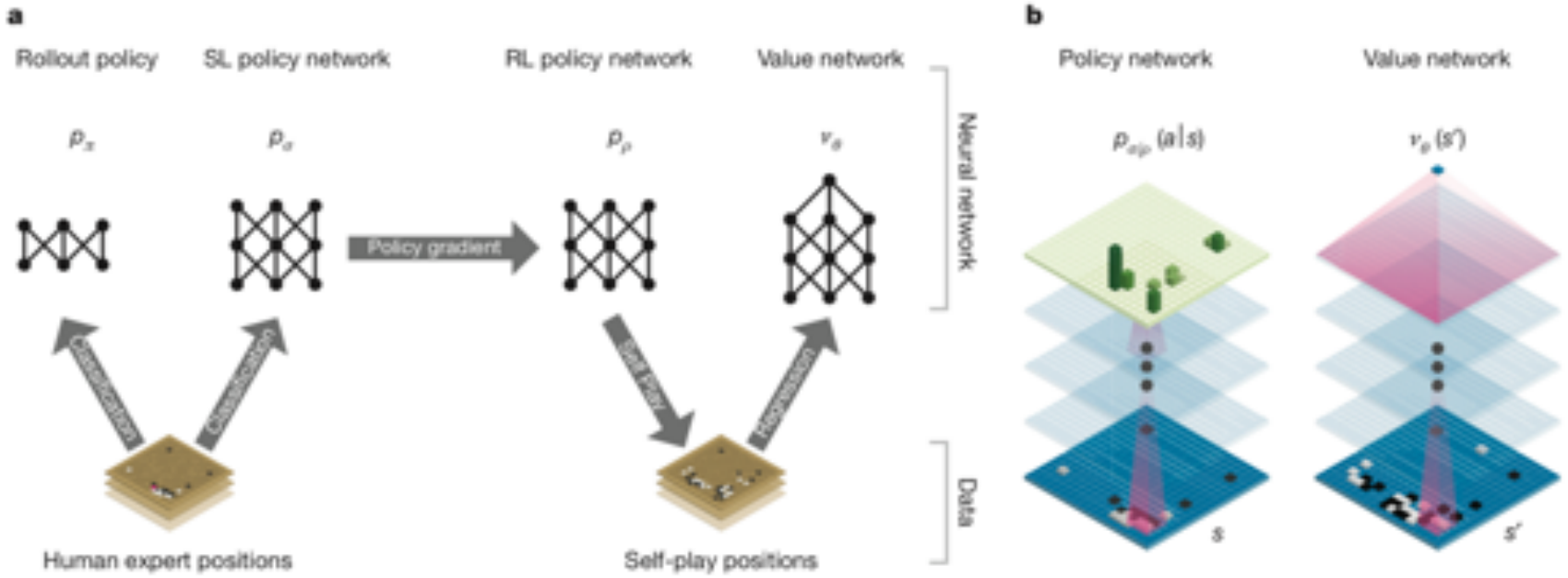
- 19x19x48 input features to the networks

Extended Data Table 2 | Input features for neural networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

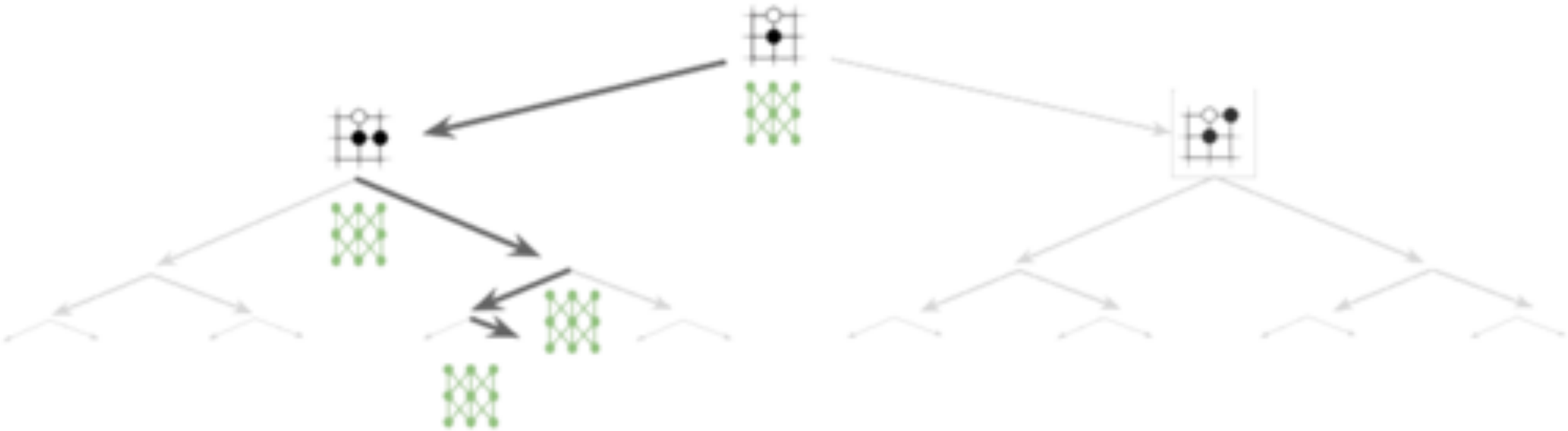
SL-RL Training Pipeline

- Policy networks (fast rollout and SL policy networks) initially trained to predict human expert moves
- Then fine-tuned through games of RL self-play



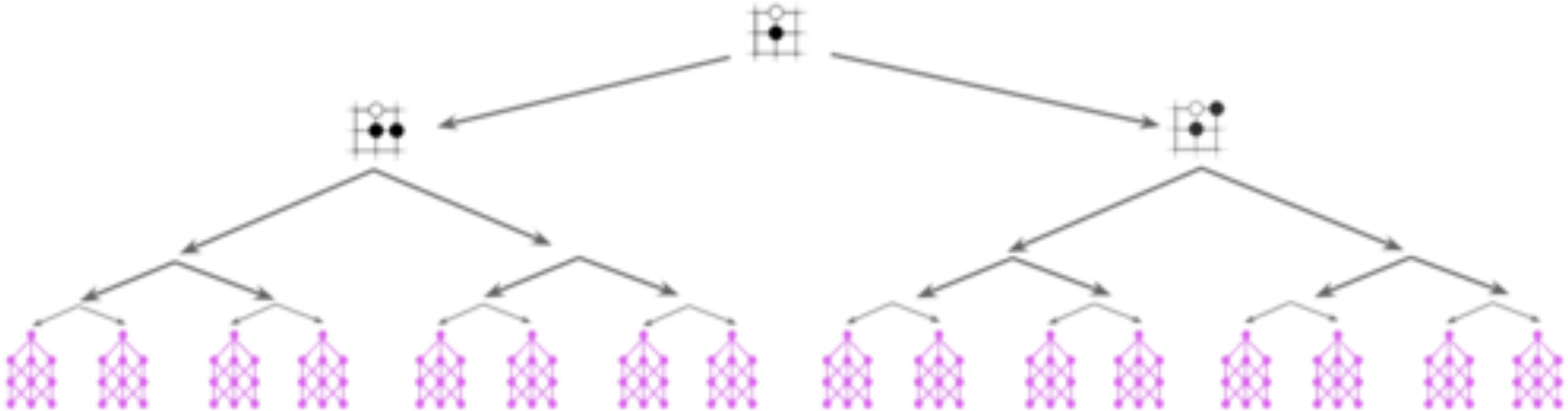
Policy Network and MCTS Search Breadth

- Approximate leaf values in MCTS using **rollout policy network** instead of MC random rollouts
- Reduce the search breadth in MCTS



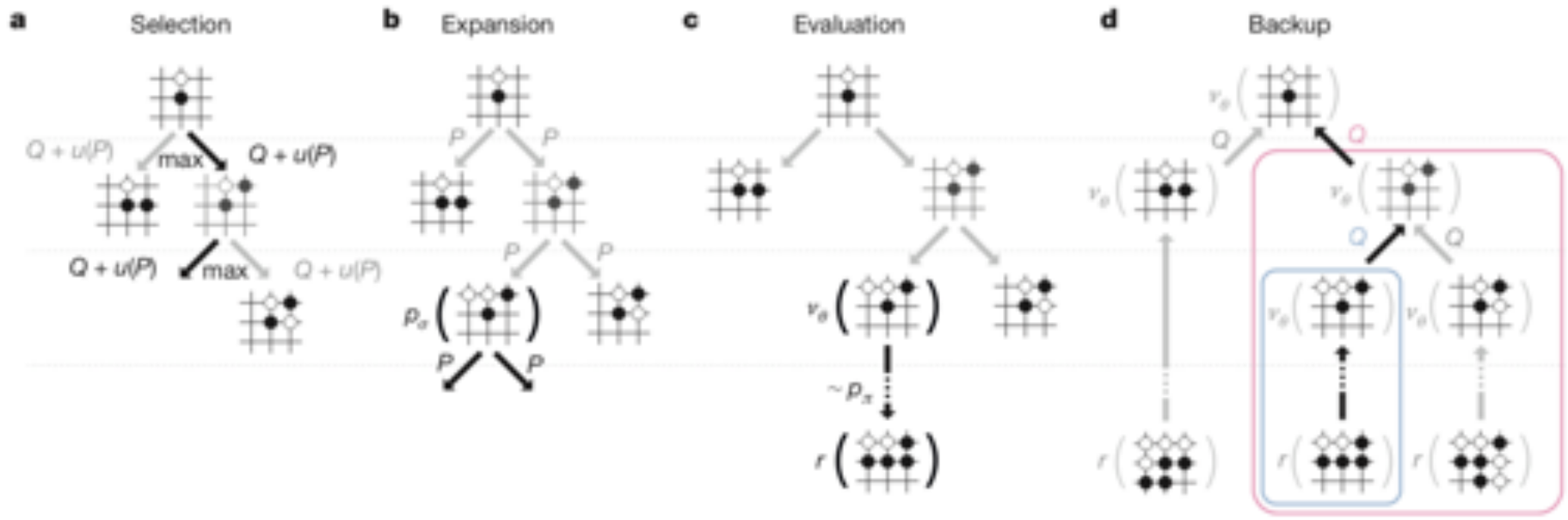
Value Network and MCTS Search Depth

- Approximate leaf values in MCTS using a value network instead of MC rollouts
- Reduce the search depth in MCTS



MCTS in AlphaGo

- Selection: Q-value and prior weight proportional to prior policy network evaluation
- Expansion: Expand tree to children of selected node, assign policy network prior
- Evaluation: MC rollouts using fast policy network and value network prediction
- Backup: Update Q-values and number of visits up to root based on evaluations

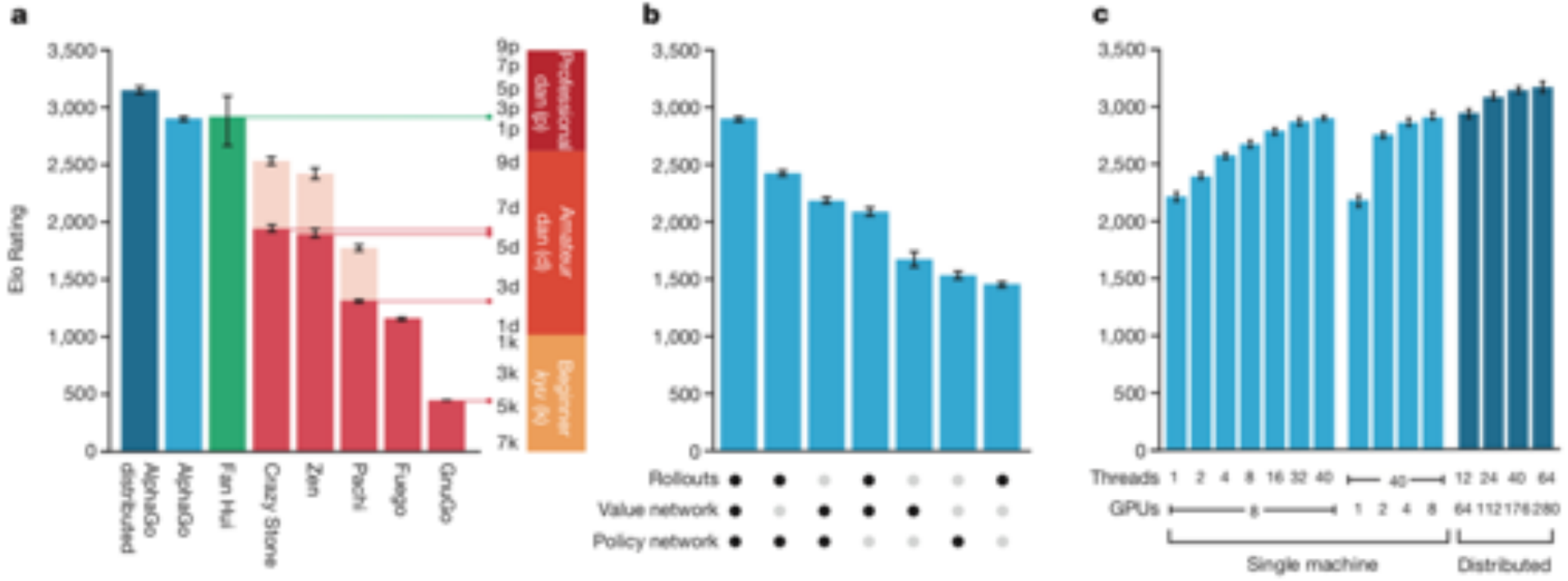


What AlphaGo Improves

- Prioritize searches and better evaluate positions
- Value network to evaluate positions
- Thus SL policy network is used for exploration
- Value network and game results are for exploitation

Results

- AlphaGo Fan defeated European Go champion Fan Hui 5-0
- AlphaGo Lee defeated world Go champion Lee Sedol 4-1
- AlphaGo Master defeated top Go players worldwide 60-0



Cost of AlphaGo

- AlphaGo: several months on 1,920 CPUs and 280 GPUs
- Cost: Estimated $35 Million in computing power

- DeepMind lost $162 million according to their financials (negative asset to Google) that year

From AlphaGo to AlphaGo Zero

Domain specific heuristics in AlphaGo (Inductive Bias)

- Pretraining on human data
- Use of handcrafted Go-specific features
- Exploitation of certain symmetries in state space

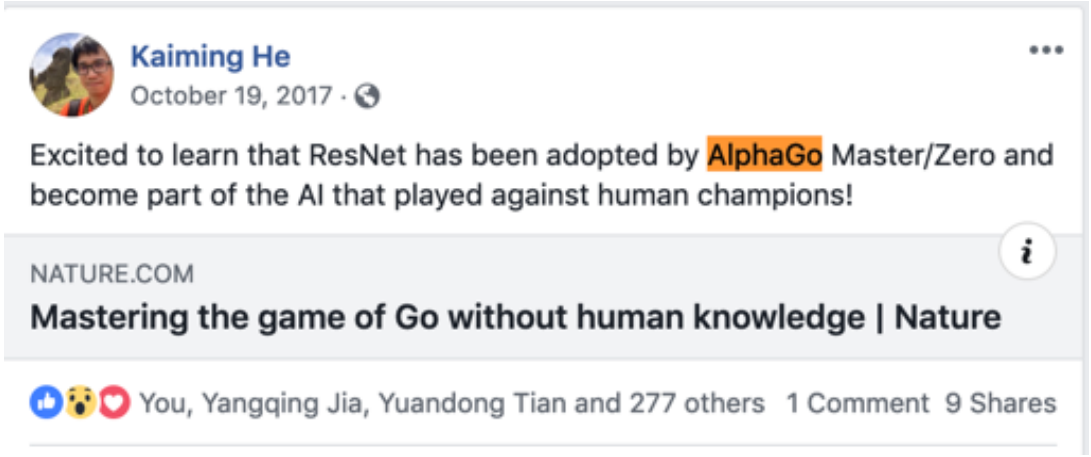
AlphaGo Zero and AlphaZero

- purer approach with stronger Go agents
- Could master other games such as Shogi and Chess

D Silver. Nature 2017, Mastering the Game of Go without Human Knowledge

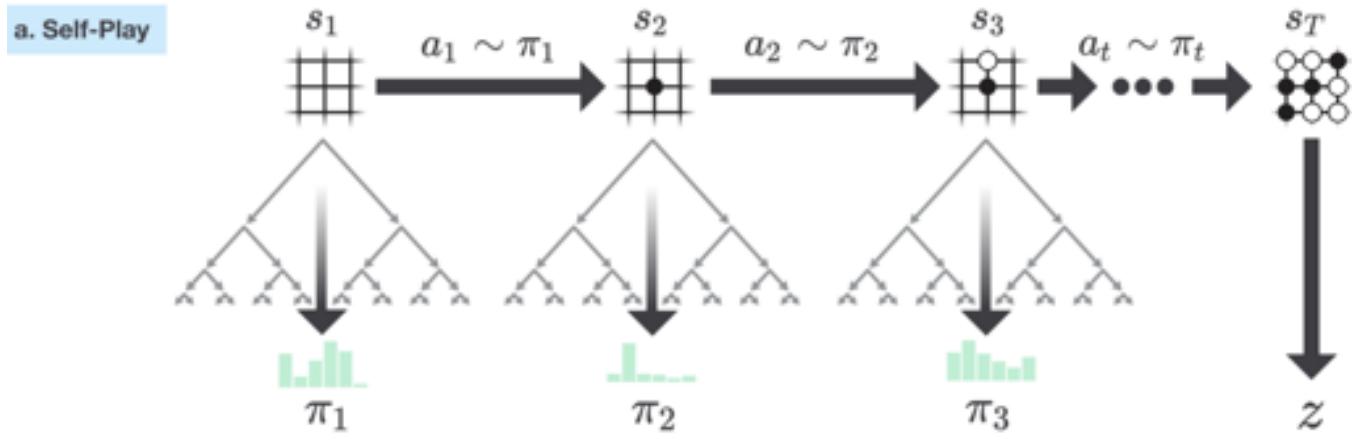
Difference of AlphaGo Zero

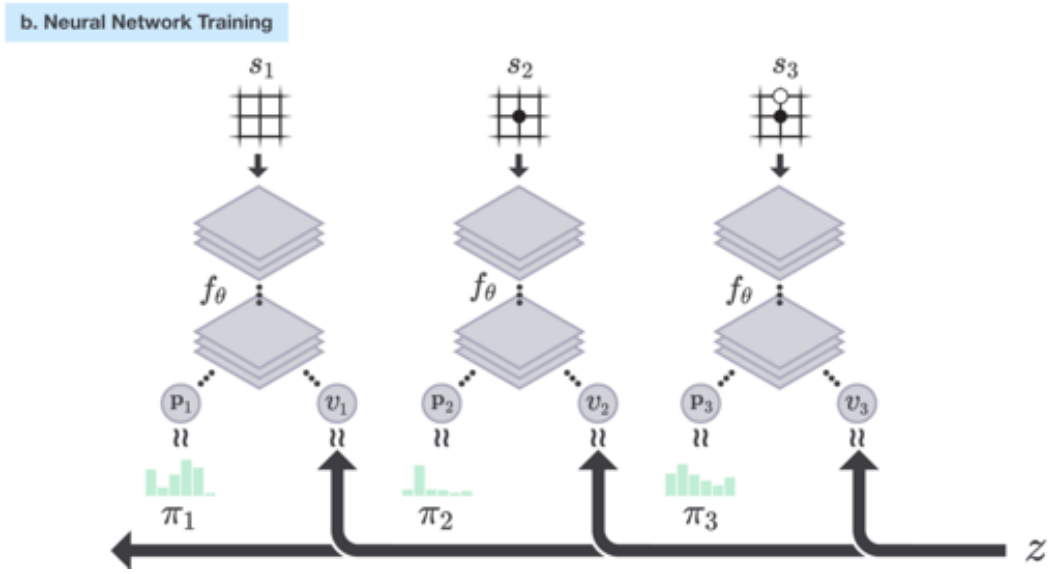
- Trained only via self-play
- One deep network for both value and policy in parallel
- MCTS evaluation via value function evaluation (no MC rollouts)
- Residual Network (ResNet) architecture



MCTS in AlphaGo Zero

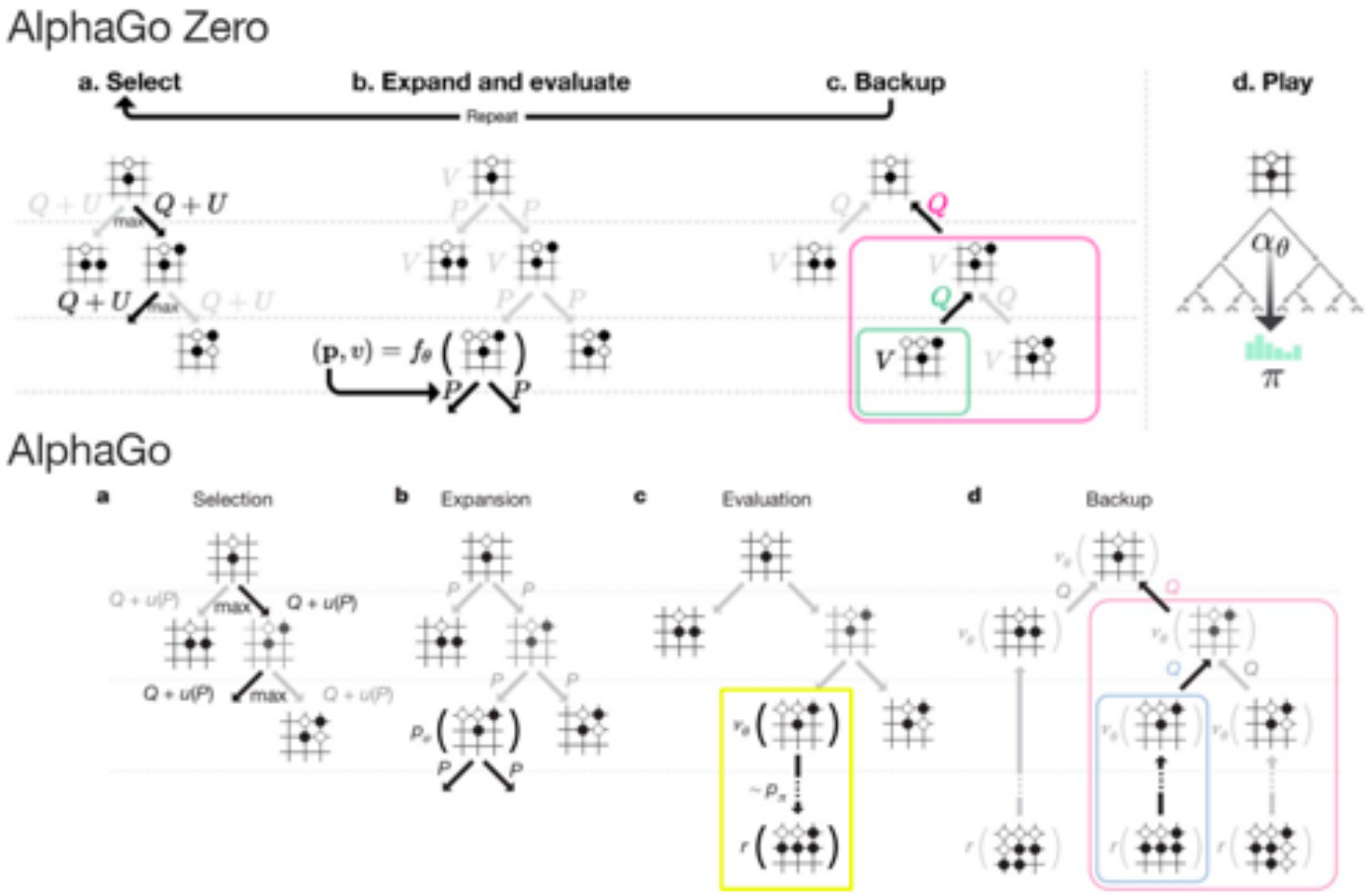
- Self-play via MCTS as search-based policy iteration



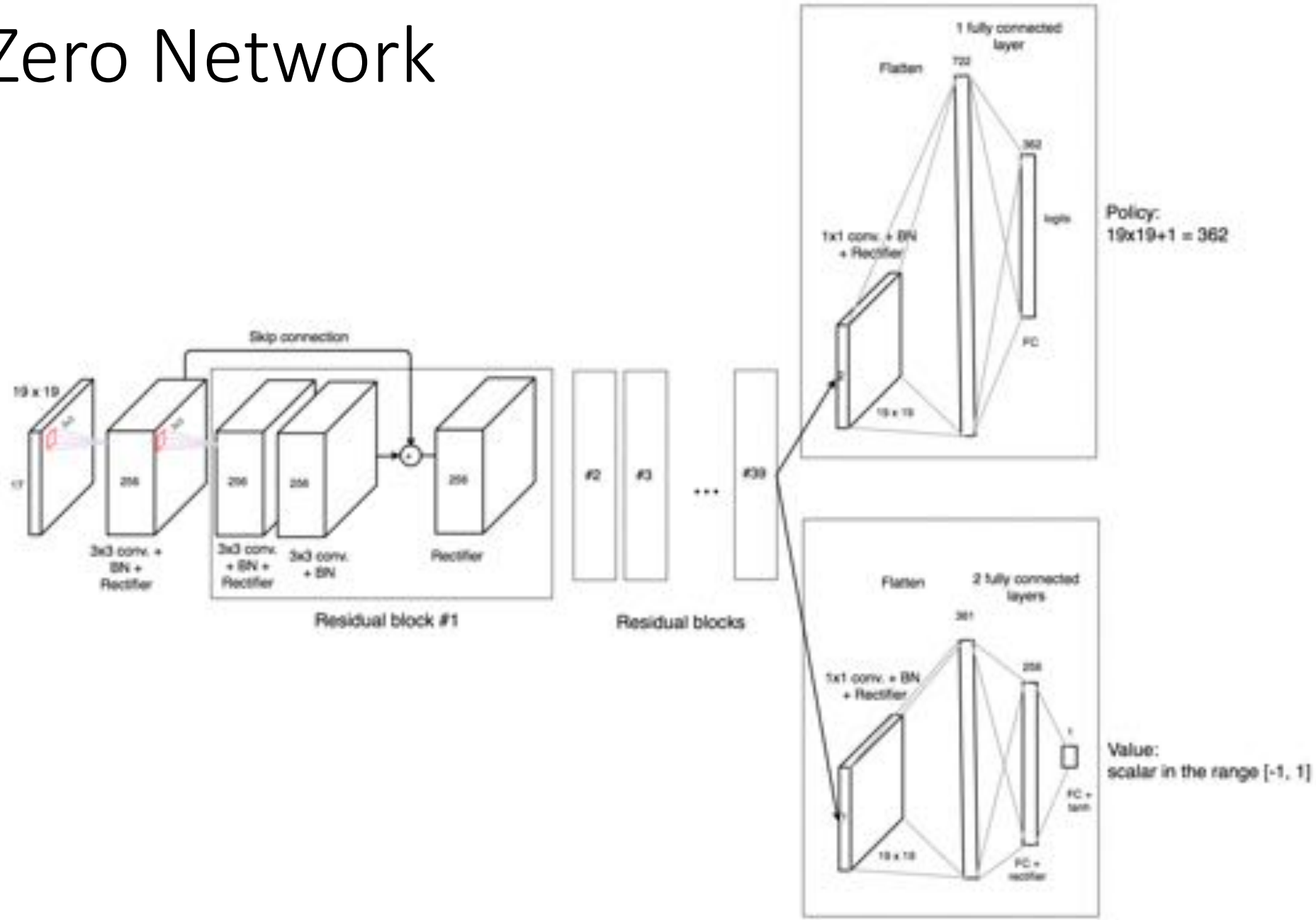


Difference between AlphaGo and AlphaGo Zero Search

- AlphaGo Zero does not use the MC rollout (in yellow)

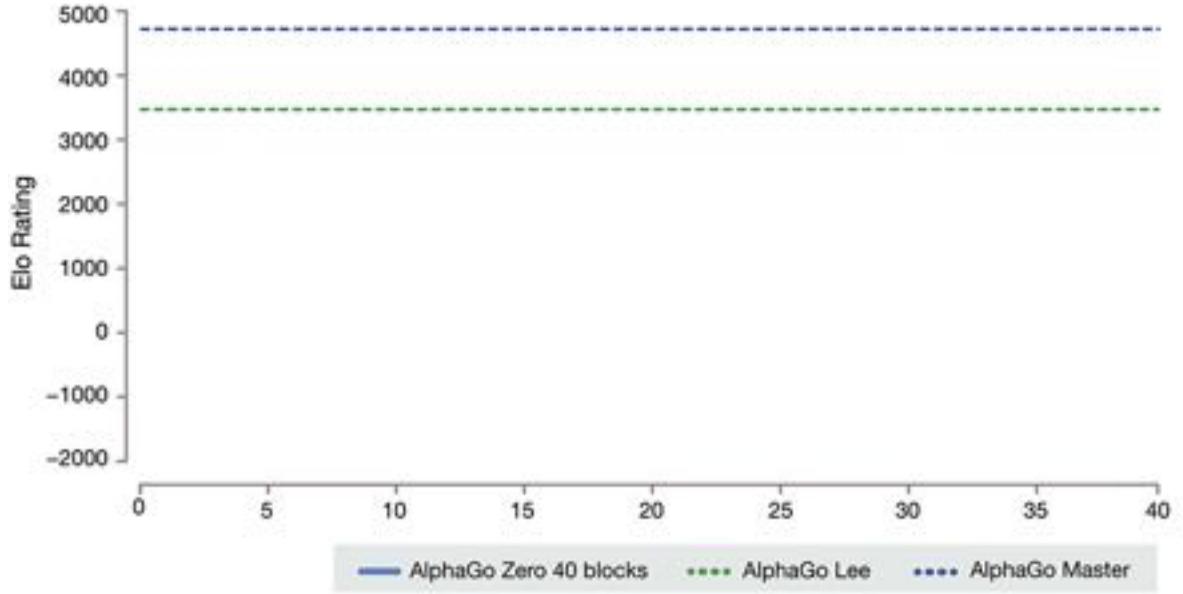


AlphaGo Zero Network



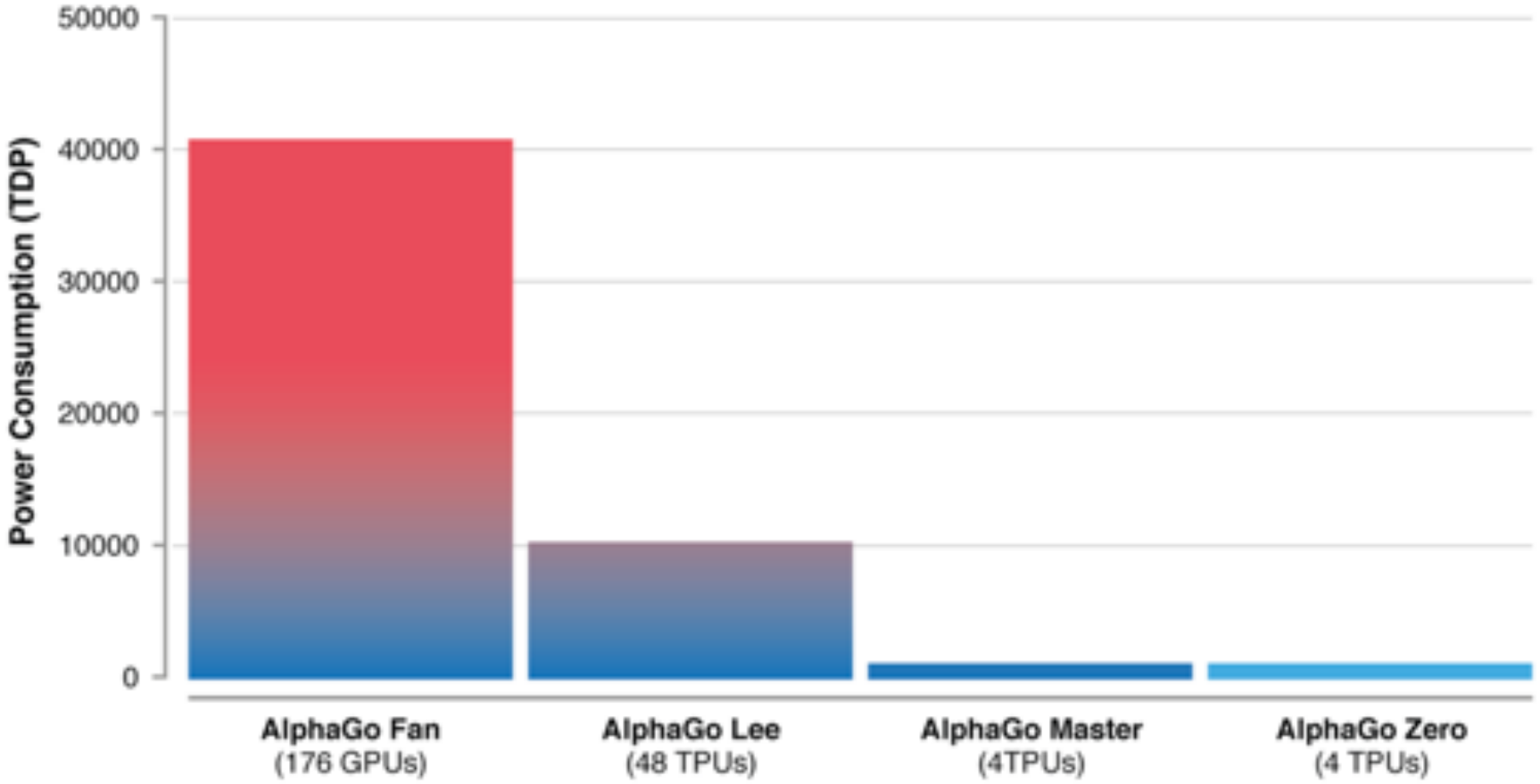
Results of AlphaGo Zero

- Passed AlphaGo Lee after just 3 days of self-play
- Passed AlphaGo Master after about 21 days
- Strongest Go player in the world after 40 days of self-play starting only from first principles



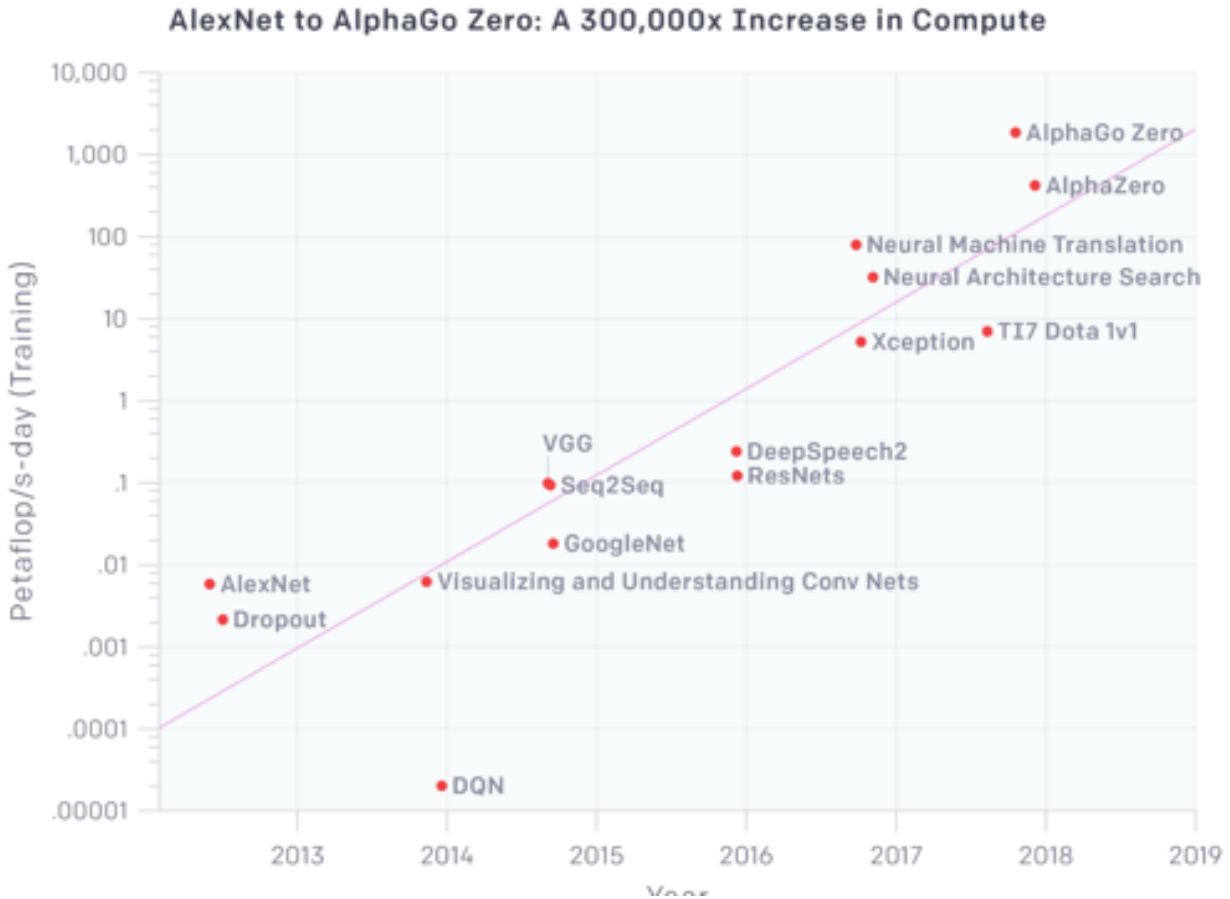
Increase in Computation

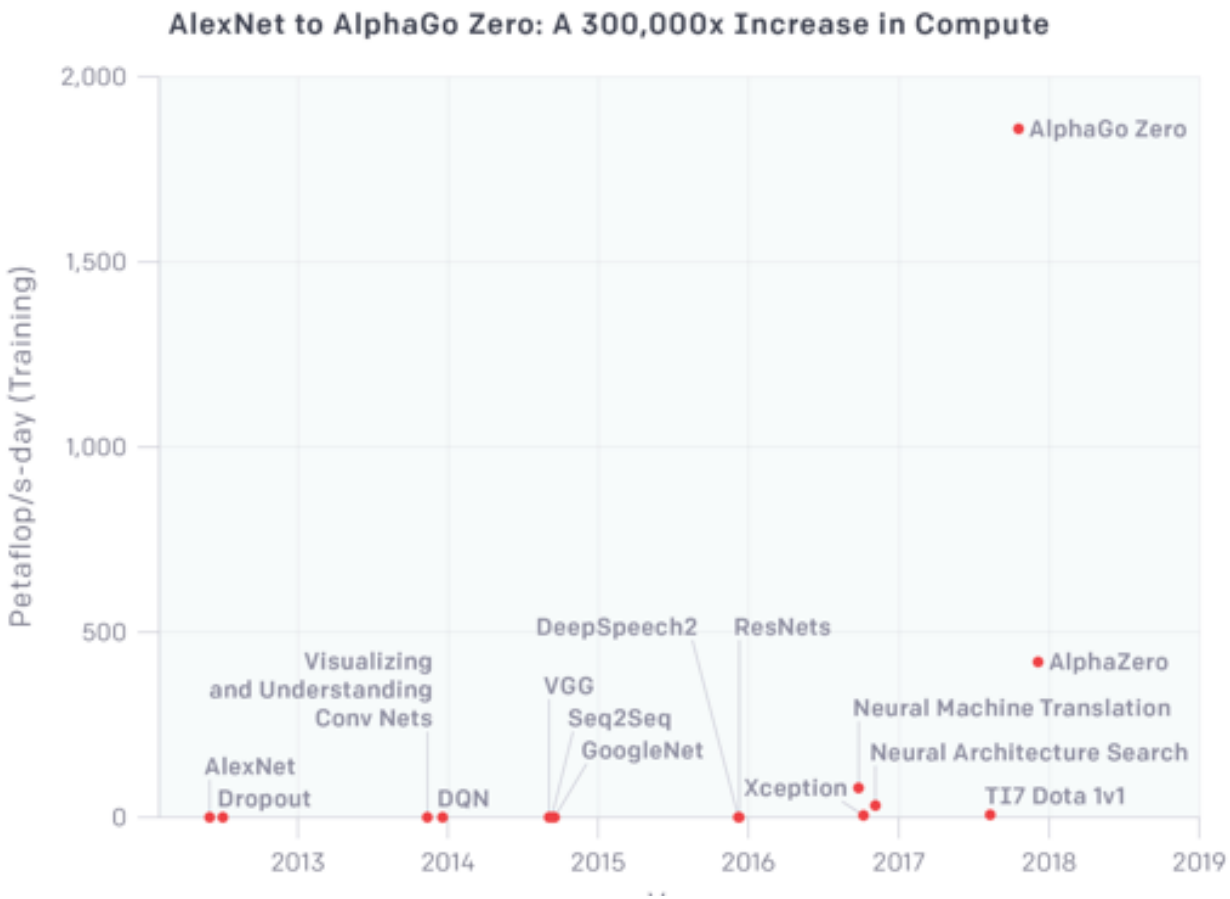
- Train the system in 40 days on four TPUs



Increase in Computation

- Train the system in 40 days on four TPUs





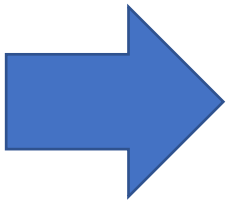
Historic Moment in Go Game (or game AI)

- AlphaGo beat Lee Sedol with 4:1 on March 8, 2016

https://www.youtube.com/watch?v=yCALyQRN3hw&t=13450s

Historic Moment in Go Game (or game AI)

- Beat Ke Jie (柯洁) at Go Summit in Wuzhen, Zhejiang. May 25, 2017

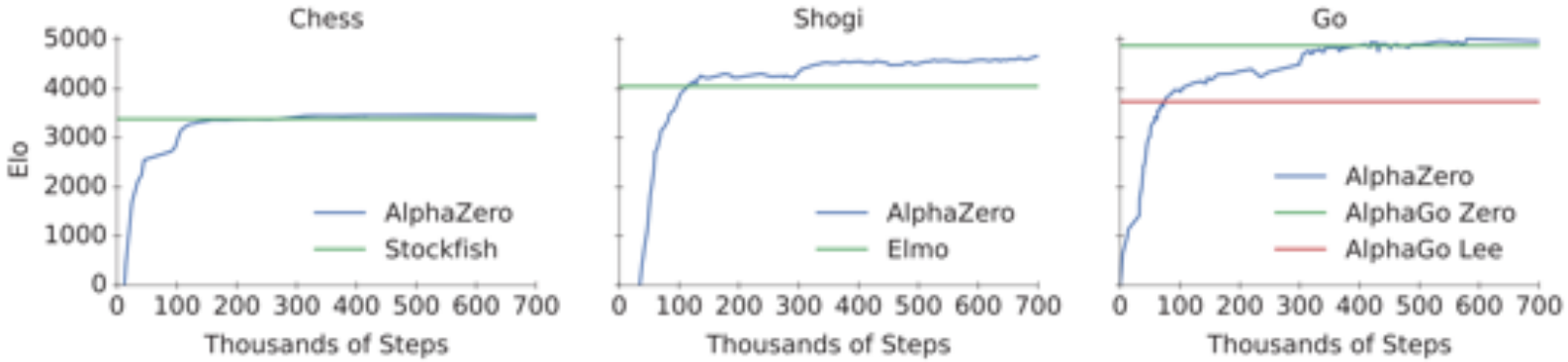


AlphaGo Movie

https://www.alphagomovie.com/ https://www.youtube.com/watch?v=WXuK6gekU1Y

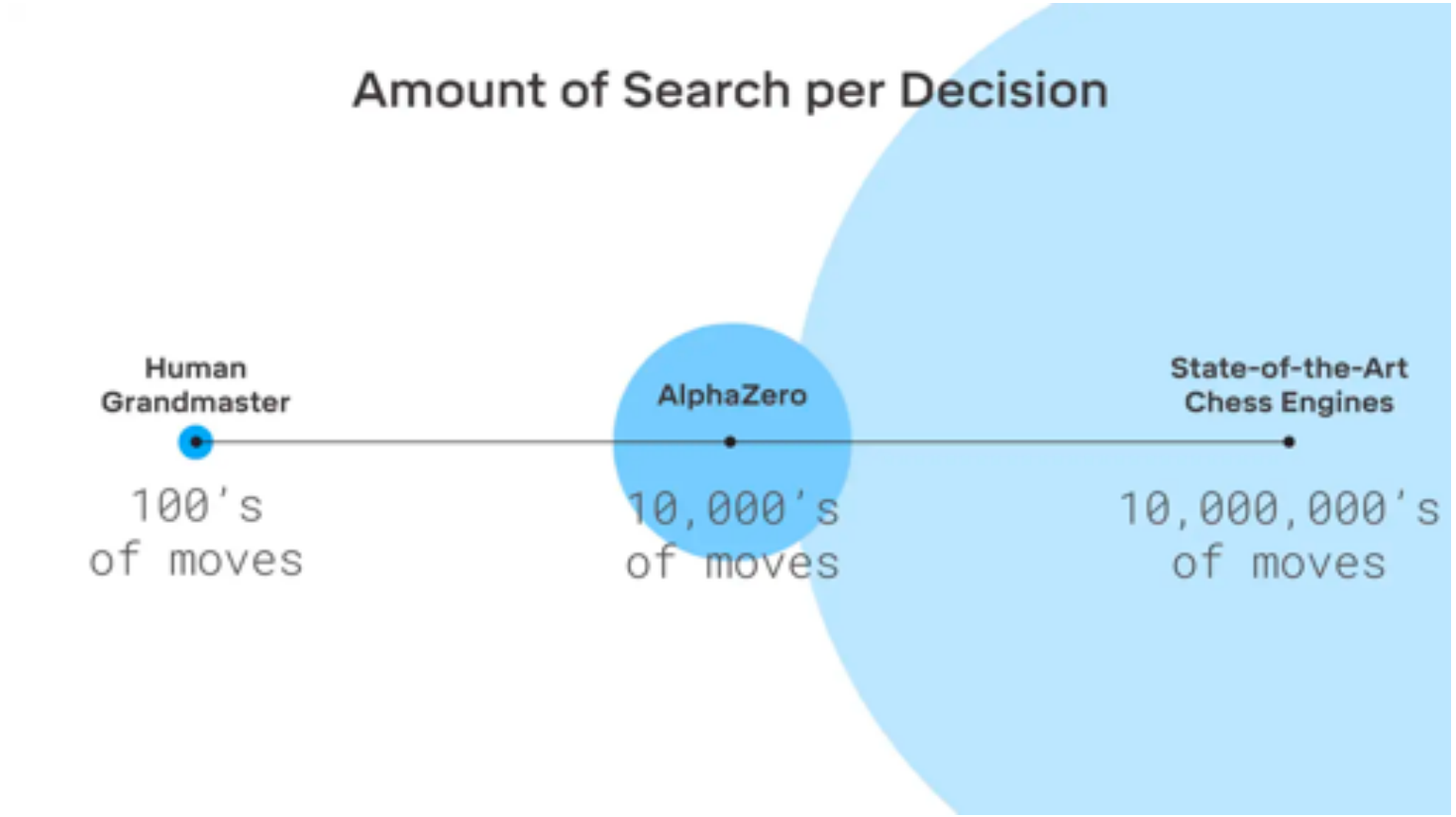
AlphaZero (Dec'17)

A general self-play RL algorithm mastering chess, shogi, and Go through self-play



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

AlphaZero (Dec'17)



Stockfish: strongest hand-crafted engines for chess

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

Difference between AlphaGo Zero and AlphaZero

More general and less domain knowledge:

- AZ has hard-coded rules for setting search hyperparameters.
- The neural network is now updated continually.
- Go (unlike chess) is symmetric under certain reflections and rotations; AlphaGo Zero was programmed to take advantage of these symmetries. AlphaZero is not.
- Chess can end in a draw unlike Go; therefore AlphaZero can take into account the possibility of a drawn game.

Knowledge Discovery in AlphaGo

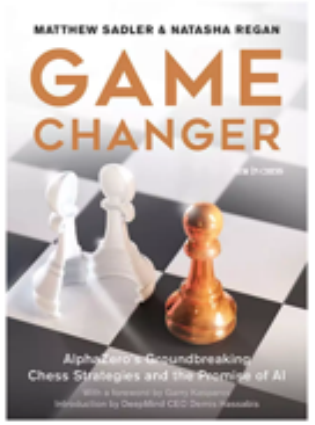
- Training from scratch by self-play without any human knowledge
- Amazing: able to create knowledge itself!

Move 37 in Match 2
“That’s a surprising move. I think it was a mistake”

> Some of its moves, such as moving the King to the centre of the board, go against shogi theory and - from a human perspective - seem to put AlphaZero in a perilous position. But incredibly it remains in control of the board. Its unique playing style shows us that there are new possibilities for the game."
>
> Yoshiharu Habu, 9-dan professional, only player in history to hold all seven major shogi titles

Knowledge Discovery in Machine Learning

- Superhuman performance leads to new knowledge
- Machine Human Learning: Learn from machines to improve human capability

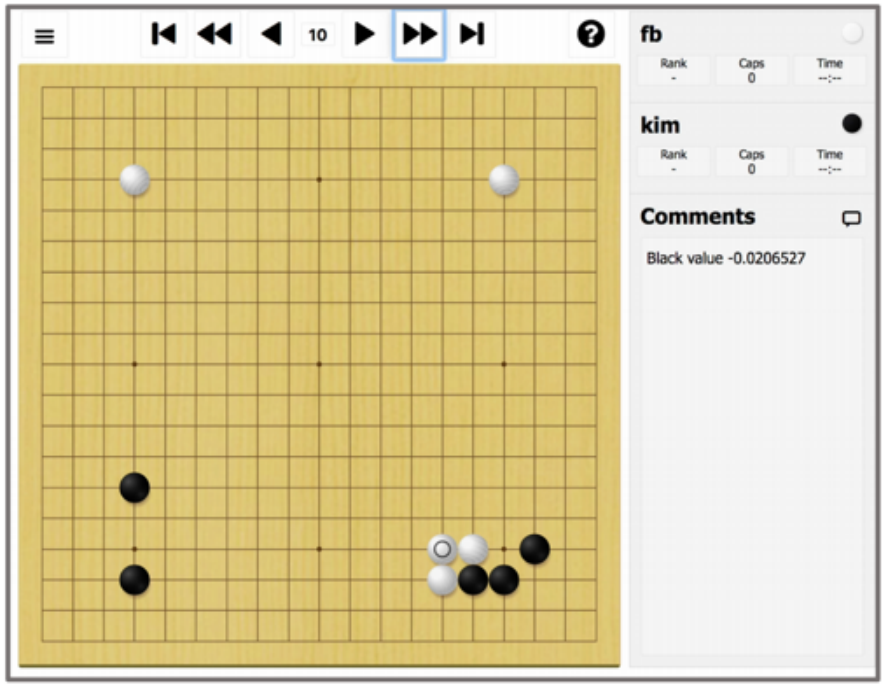


Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI
by Matthew Sadler , Natasha Regan , et al. | Feb 15, 2019
★★★★½ 14
Paperback
$16.34 $24.95
Get it as soon as **Wed, Mar 27**
FREE Shipping on orders over $25
shipped by Amazon
More Buying Choices
$14.99 (91 used & new offers)

2-time British Chess Champion Matthew Sadler analyzes the game played by AlphaZero

OpenGo from Facebook AI Research

- Reproducing AlphaZero
- ELF OpenGo: Trained with 2000 GPUs in 2 weeks

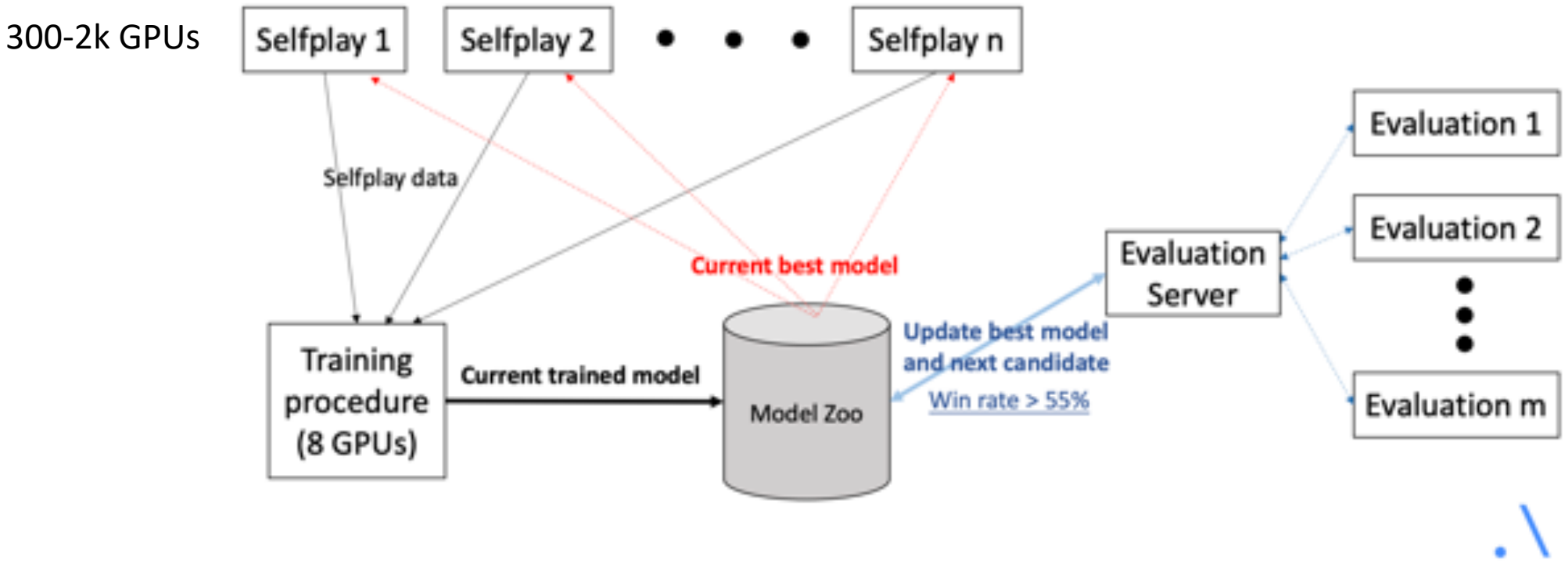


https://github.com/pytorch/ELF

https://research.fb.com/facebook-open-sources-elf-opengo/

OpenGo from Facebook AI Research

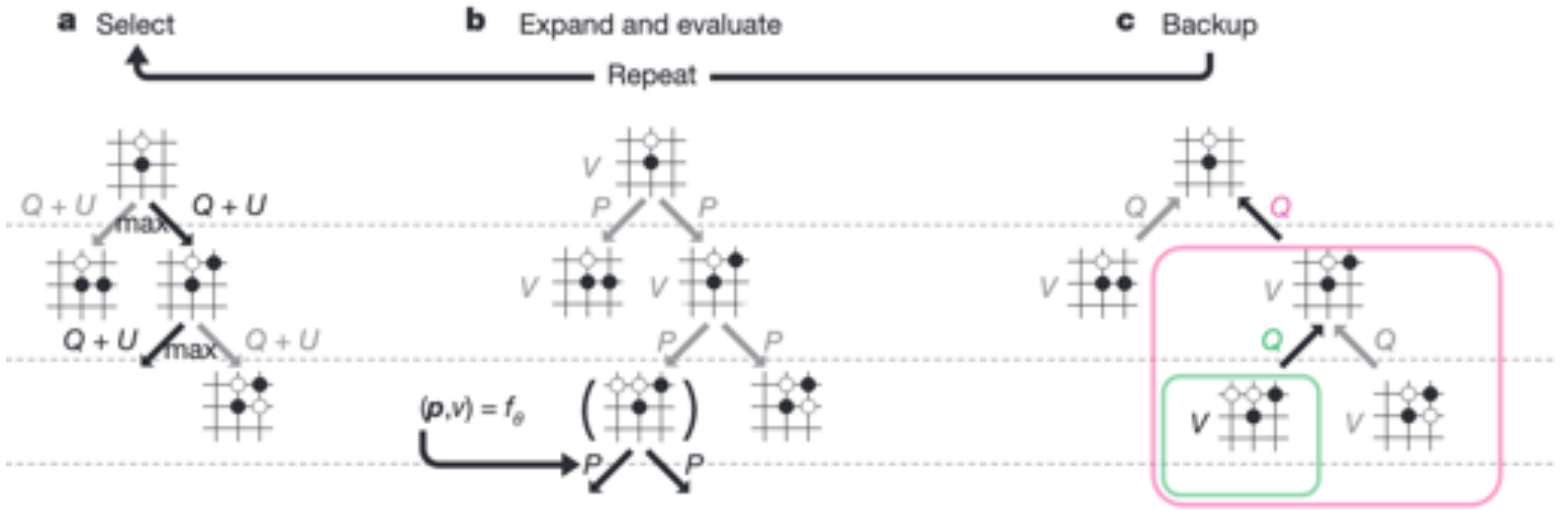
- Distributed training platform ELF



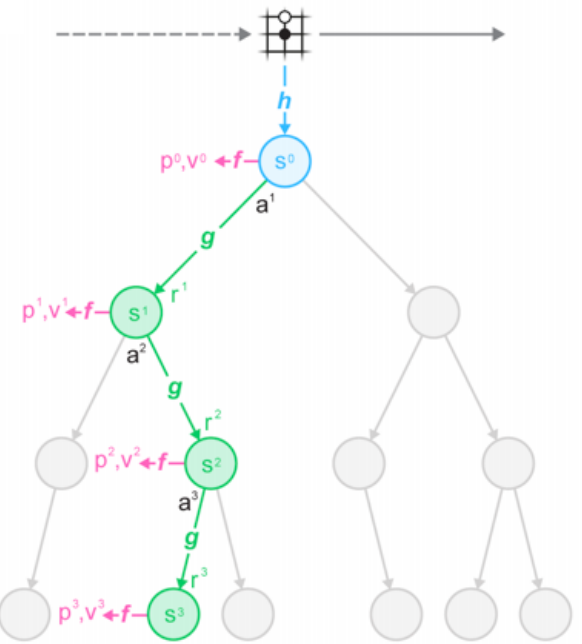
https://github.com/pytorch/ELF https://yuandong-tian.com/reproducibility.pdf

MuZero: Model-based Monte Carlo Tree Search

MCTS in AlphaGo Zero

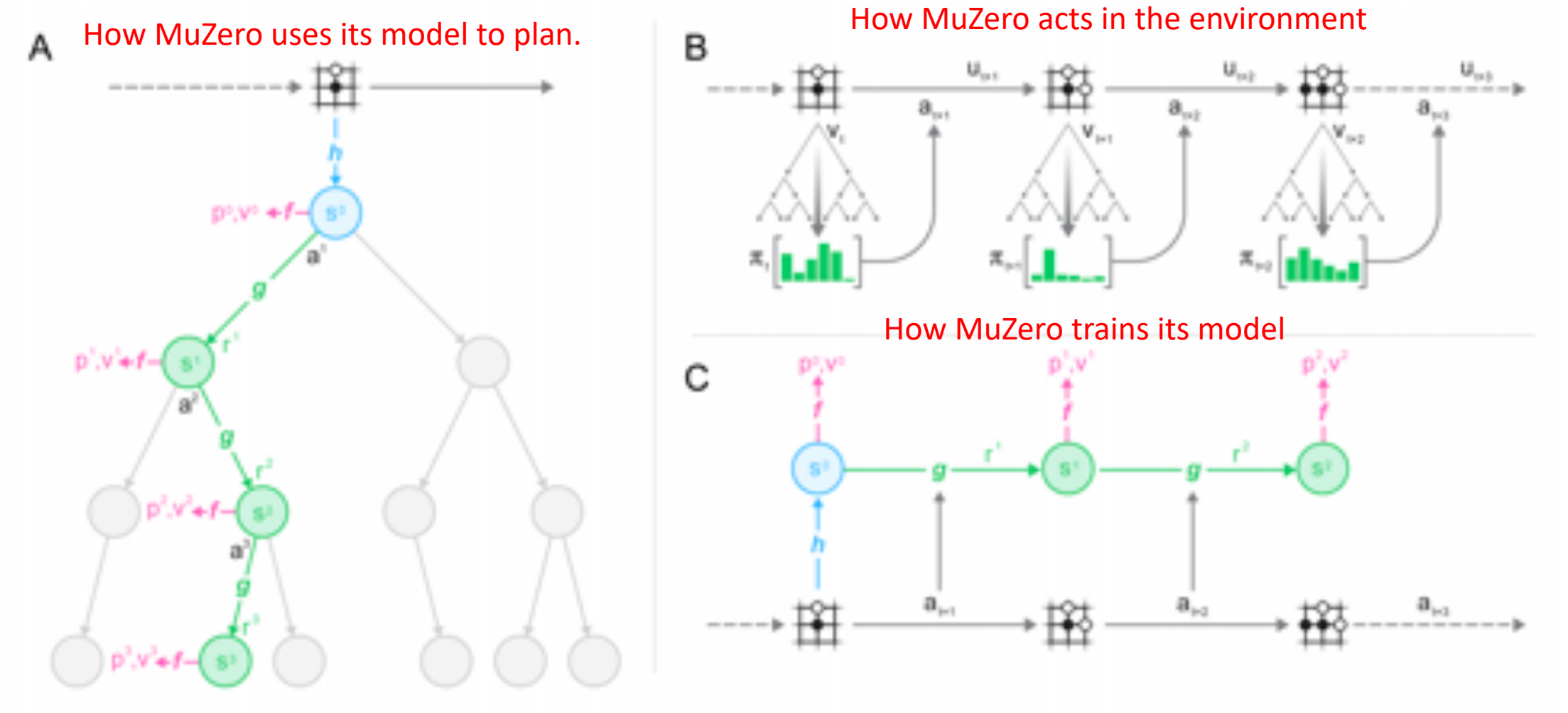


MCTS in MuZero

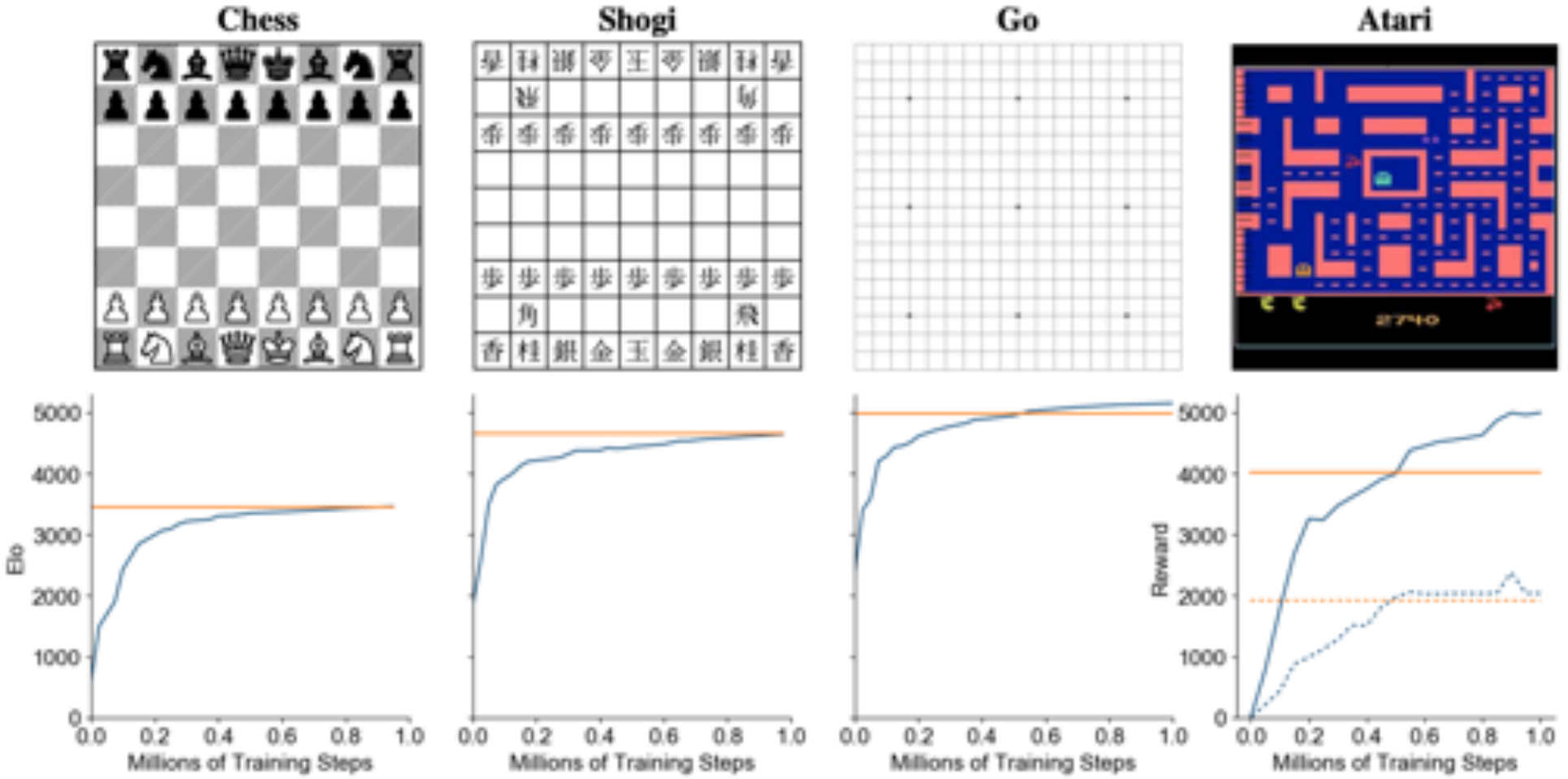


Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. https://arxiv.org/abs/1911.08265

MuZero: Model-based Monte Carlo Tree Search



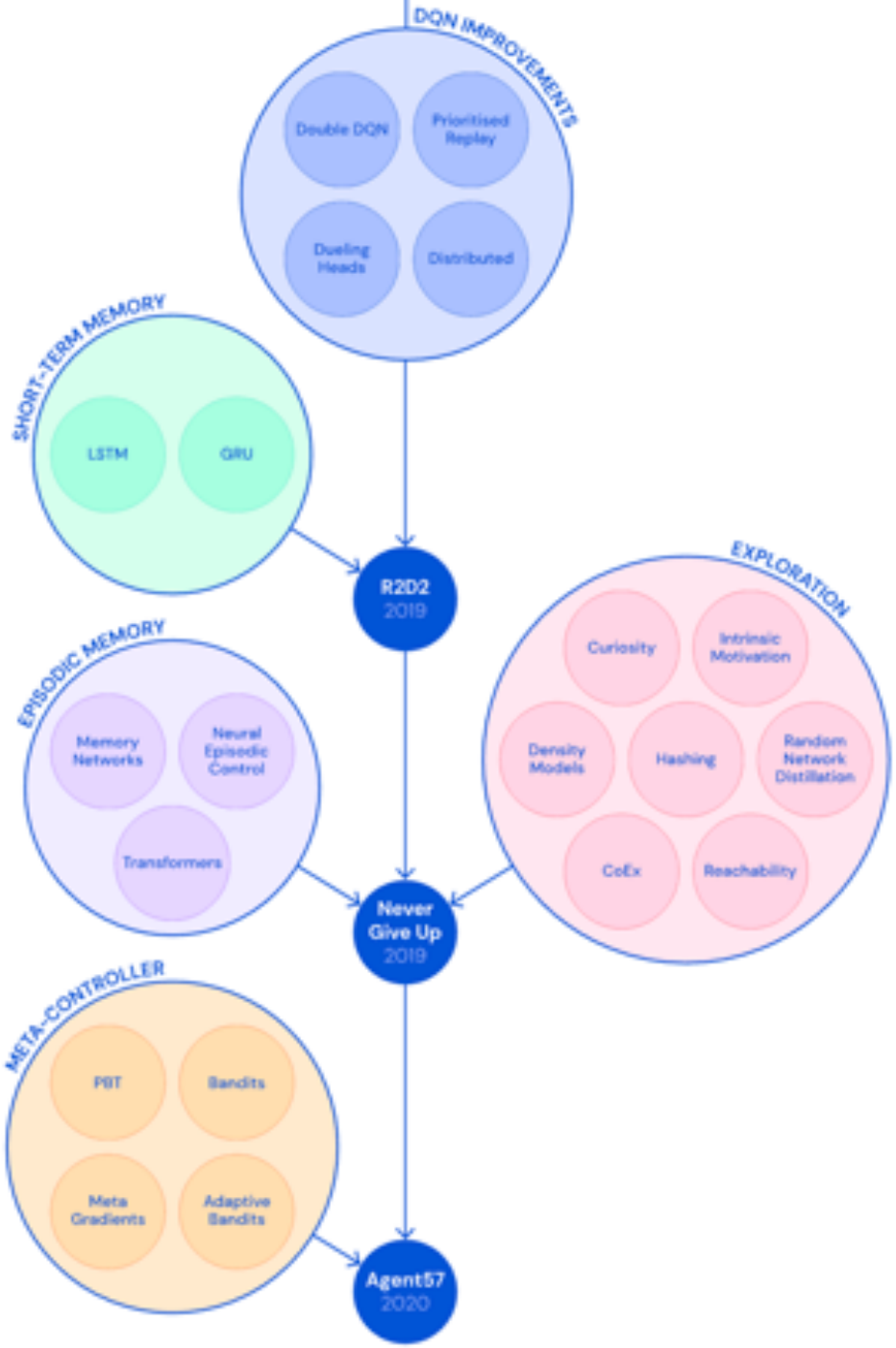
MuZero: Model-based Monte Carlo Tree Search



Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model
https://arxiv.org/abs/1911.08265

Conclusion

- DeepMind Alpha series is the miracle of engineering
- It is the milestone of AI history, starting the hyper of the AI revolution
- Opensource: https://github.com/pytorch/elf
- https://ai.facebook.com/blog/open-sourcing-new-elf-opengo-bot-and-go-research/



Latest (March 31, 2020): Agent57

- Outperforming the human Atari benchmark
- https://deepmind.com/blog/article/Agent57-Outperforming-the-human-Atari-benchmark