Multi-Agent Reinforcement Learning

Northwestern | McCORMICK SCHOOL OF ENGINEERING

Omkar Ranadive

Autocurriculum: The Hypothesis

- In a multi-agent system, the competition and cooperation between agents leads to emergence of innovation
- Social interaction leads to naturally emergent curriculum called as autocurriculum

The problem problem

- Intelligence is the ability to adapt to diverse set of environments
- So, for single agents the “cleverness” is bounded by the complexity of the environment

Exploration by exploitation

- Structure learning by changing the underlying environment
- Such a change is called as a challenge
- Challenges cause agent to explore new states by exploiting known information

Achieving this using autocurriculum

- Use multi-agents - No environment engineering needed
- Social interaction between agents give rise to challenges
- Challenges are generated by system, hence called autocurricula

Exogenous challenges

- Exogenous challenge originates outside the adaptive unit
- Example - Agent 1 changes its strategy if Agent 2 changes its strategy
- May not always work. Example - Rock paper scissors

Self-play

- Play against an older version of yourself - neither too weak neither too strong
- Agent will learn to exploit its own errors
- Approaches Nash equilibrium for small environments

Endogeneous challenges

- Challenges faced by cooperating agents
- Agents must learn to find socially beneficial outcomes

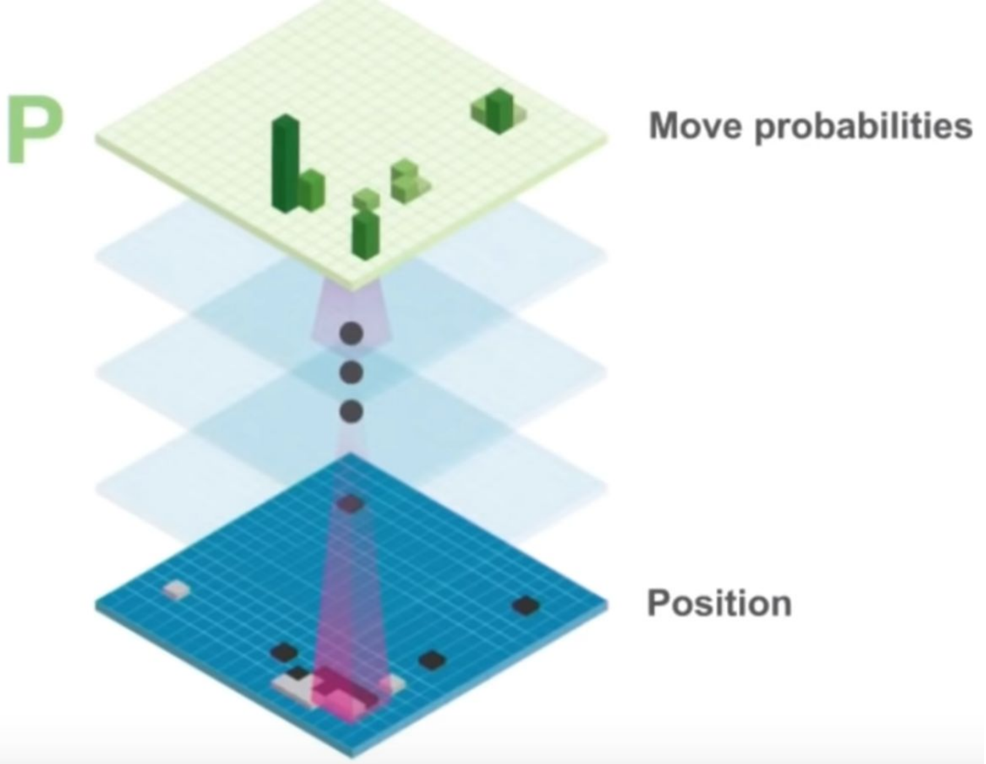
Hide and Seek: Rules

- Hiders avoid line of sight, seekers bring hiders in line of sight
- Preparation phase - Hiders prepare
- Team based reward - +1 if **all** hiders are hidden, -1 if any is seen b seeker
- Agents can grab objects, lock objects and unlock objects
- **No explicit incentive to interact with the objects**

Hide and Seek: Demo

Emergent Tool Use from Multi-Agent Interaction

Alpha Go: Policy and Value networks



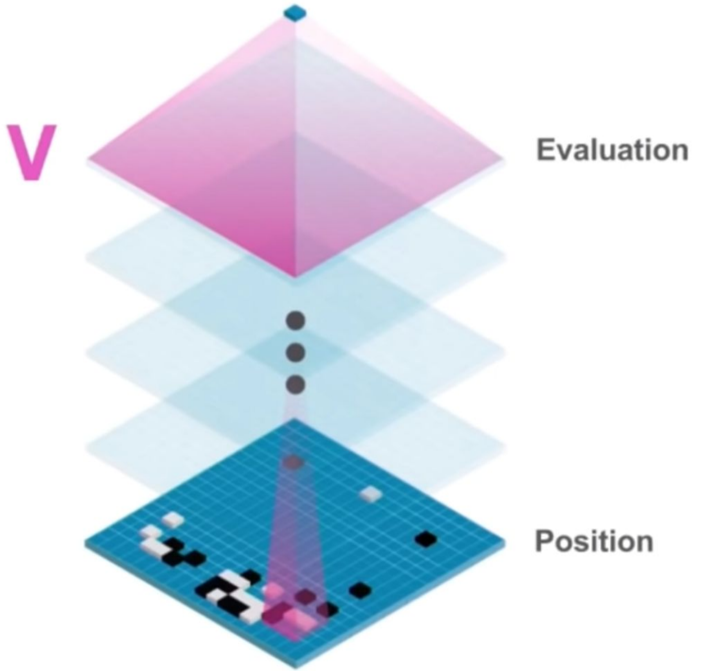


Image courtesy of Deepmind, NIPS 2017

Alpha Go: Training process

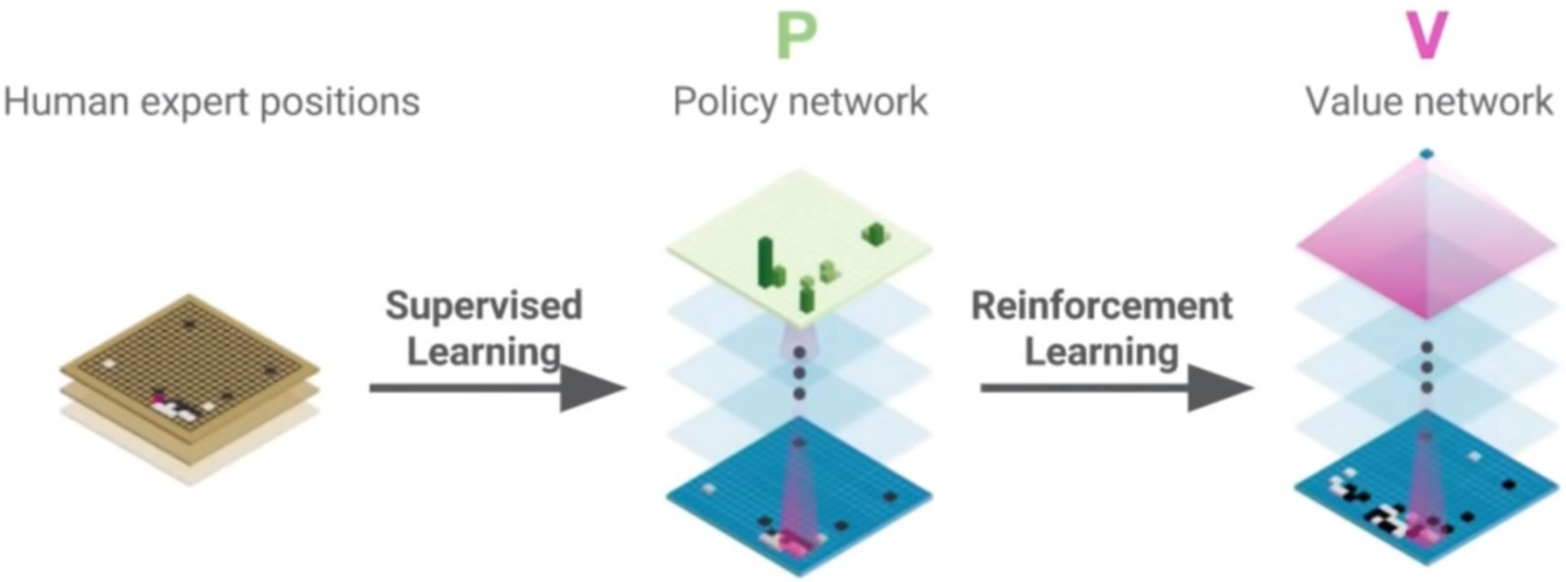


Image courtesy of Deepmind, NIPS 2017

Alpha Go: Reducing breadth

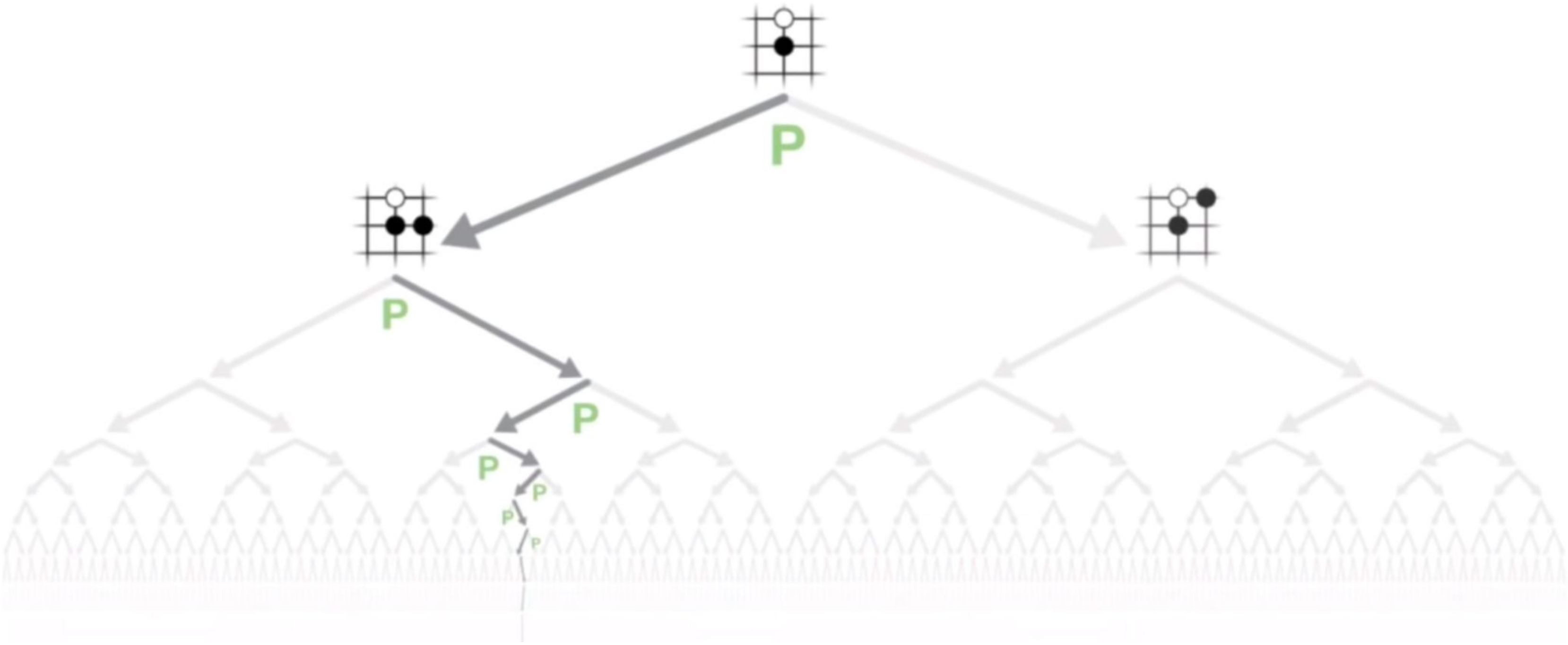


Image courtesy of Deepmind, NIPS 2017

Alpha Go: Reducing depth

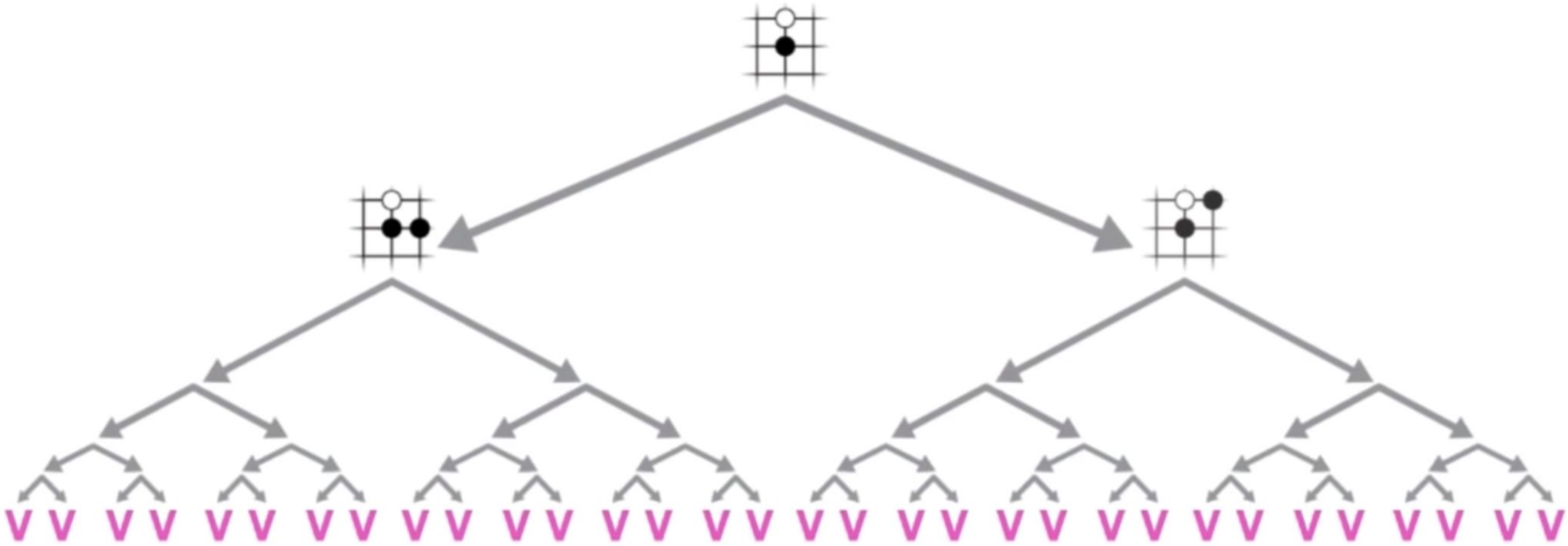


Image courtesy of Deepmind, NIPS 2017

Monte Carlo Tree Search (MCTS)

- Simulate k episodes from current state s_t using simulation policy π

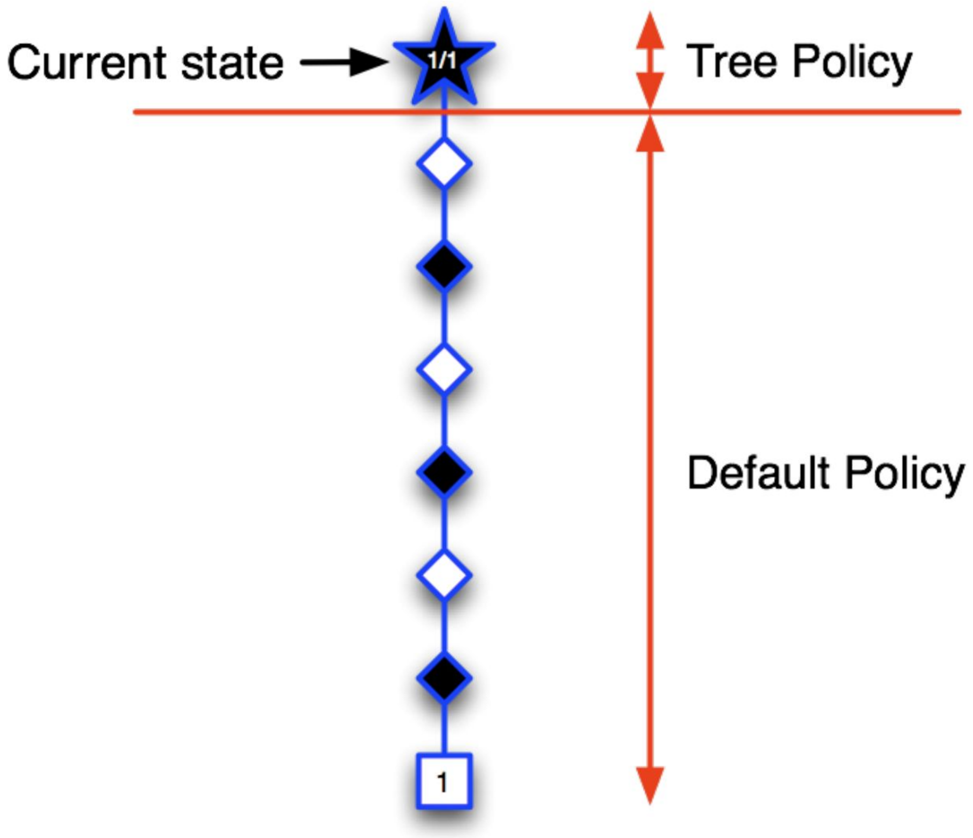
$$\{s_t, A_t^k, R_{t+1}^k, S_{t+1}^k, \ldots, S_T^k\}_{k=1}^K \sim \mathcal{M}_\nu, \pi$$

- Build a search tree of visited states and actions
- Evaluate states as mean return of episodes

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

MCTS

- Each simulation consists of two phases (in-tree, out-of-tree)
  - Tree policy (improves): pick actions to maximise Q(S;A)
  - Default policy (fixed): pick actions randomly
- Keep repeating such simulations
- Main idea: We visit the most beneficial states more often


Current state
1/1
Tree Policy
Default Policy
1

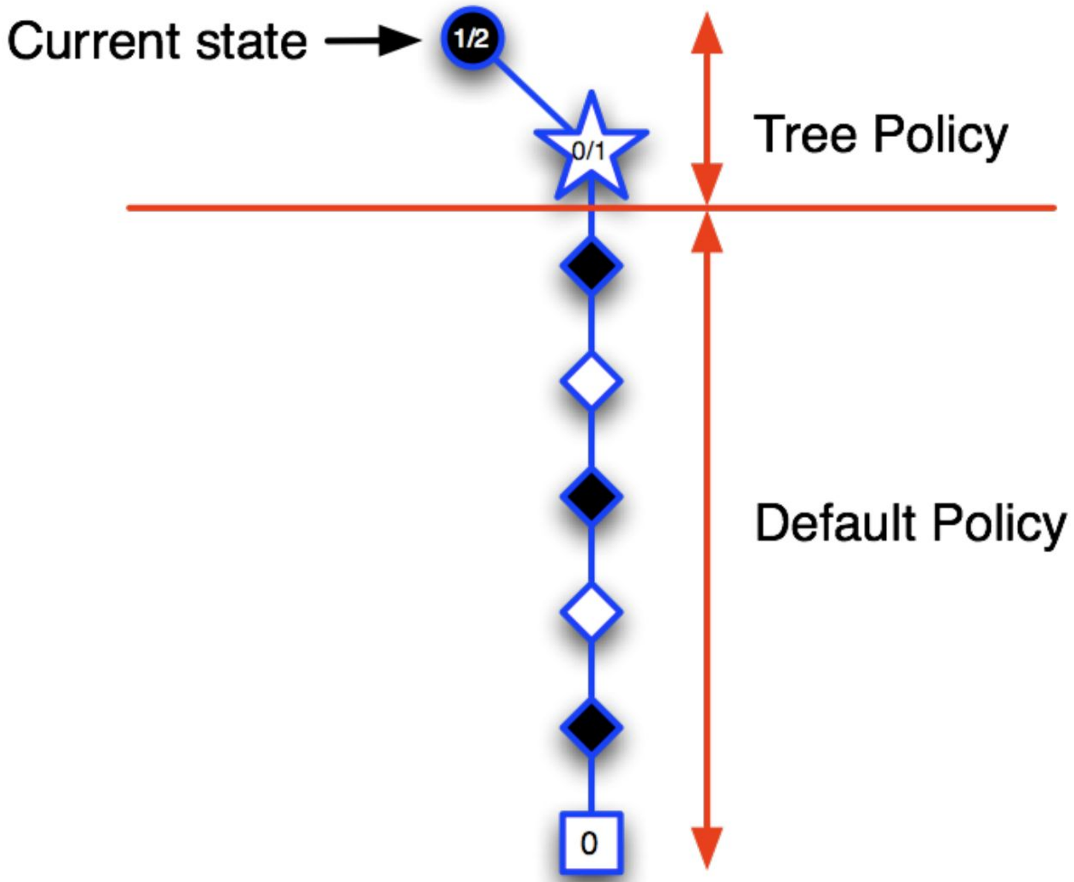
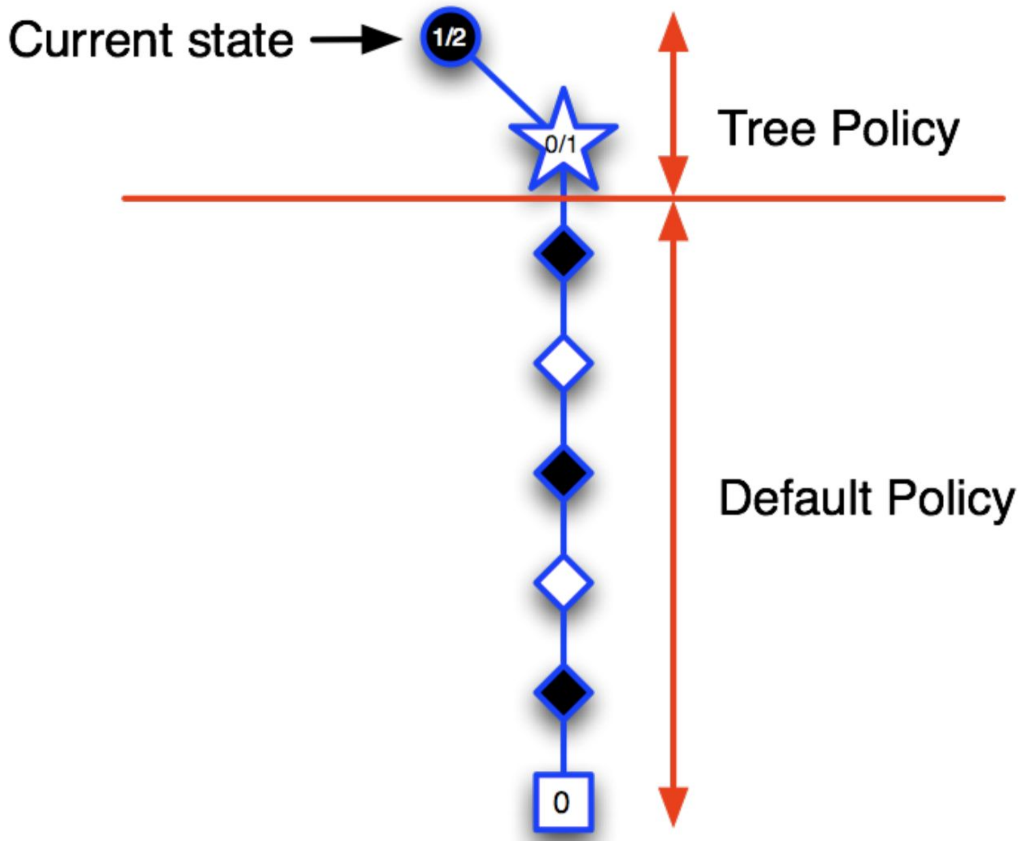


Image courtesy of David Silver


Current state
1/2
0/1
Tree Policy
Default Policy
0

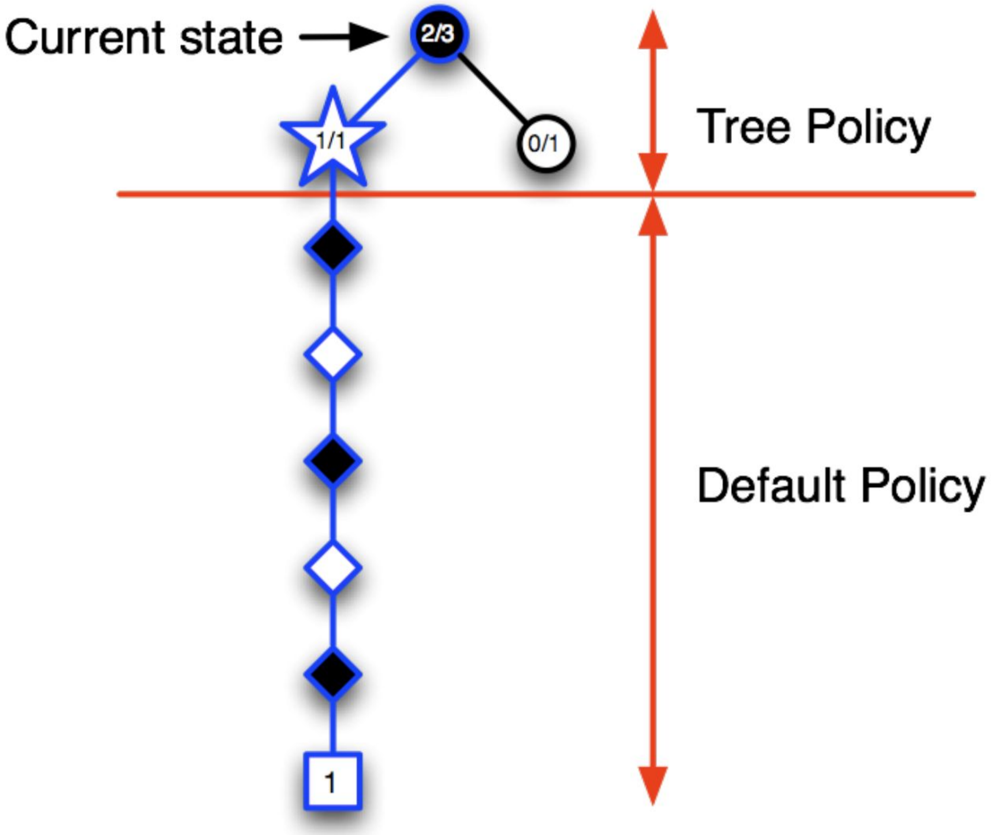


Image courtesy of David Silver

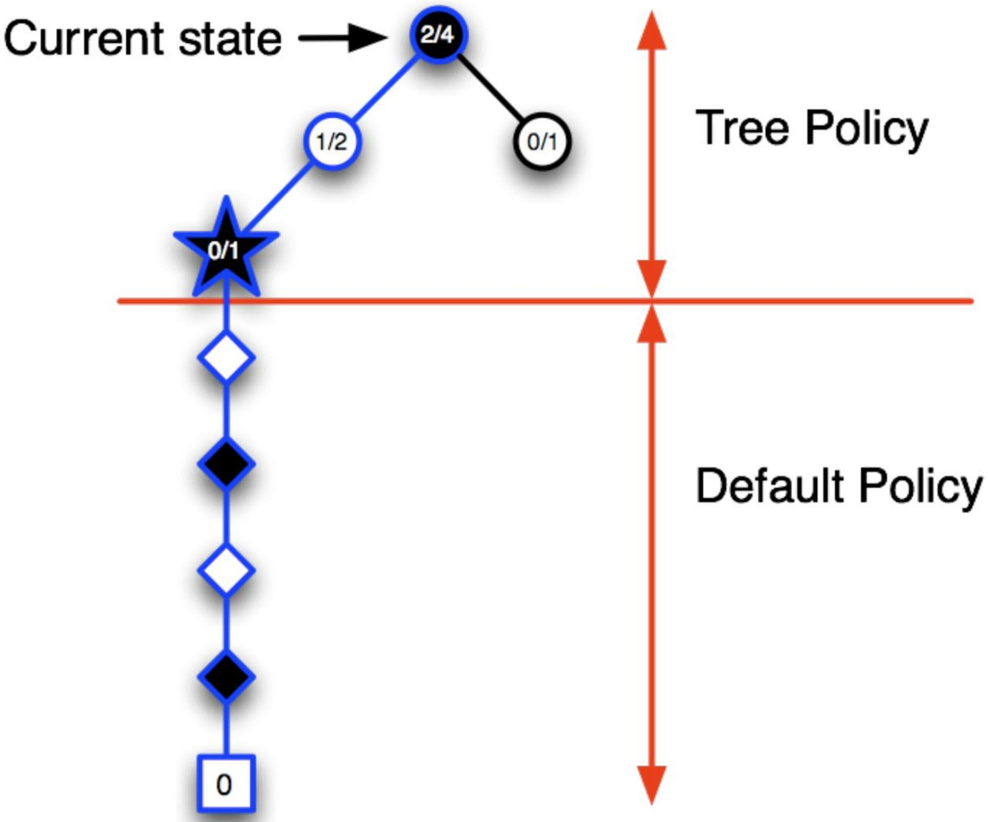
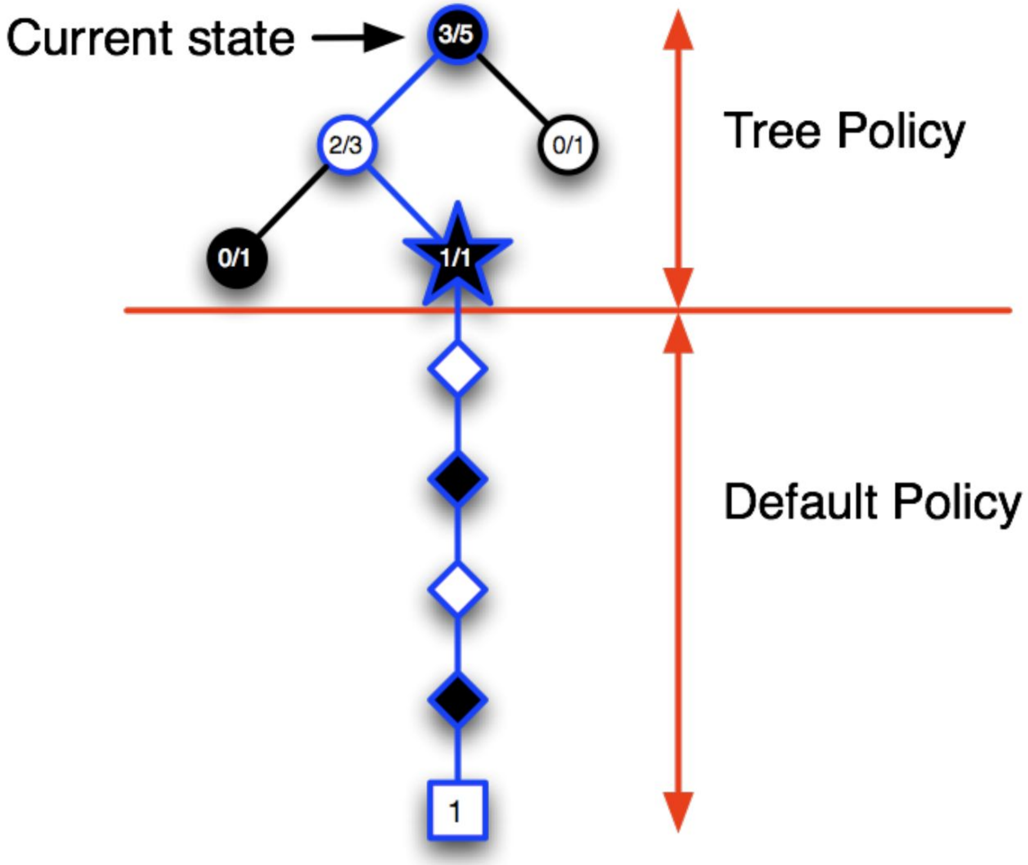


Image courtesy of David Silver

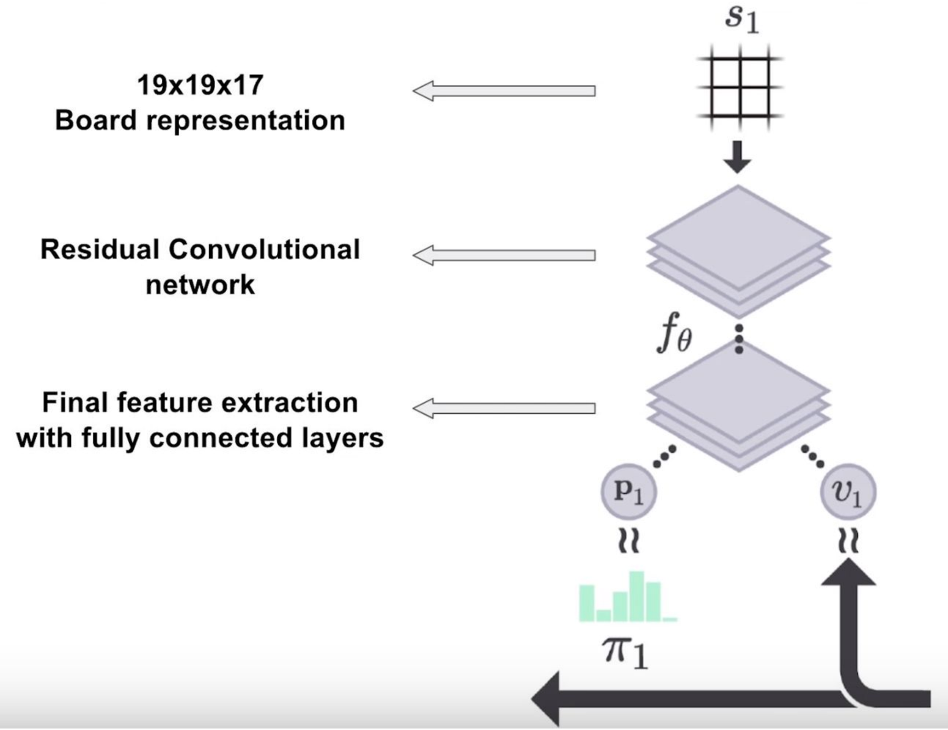

Current state
3/5
2/3
0/1
0/1
1/1
1
Tree Policy
Default Policy

Alpha Go: Zero

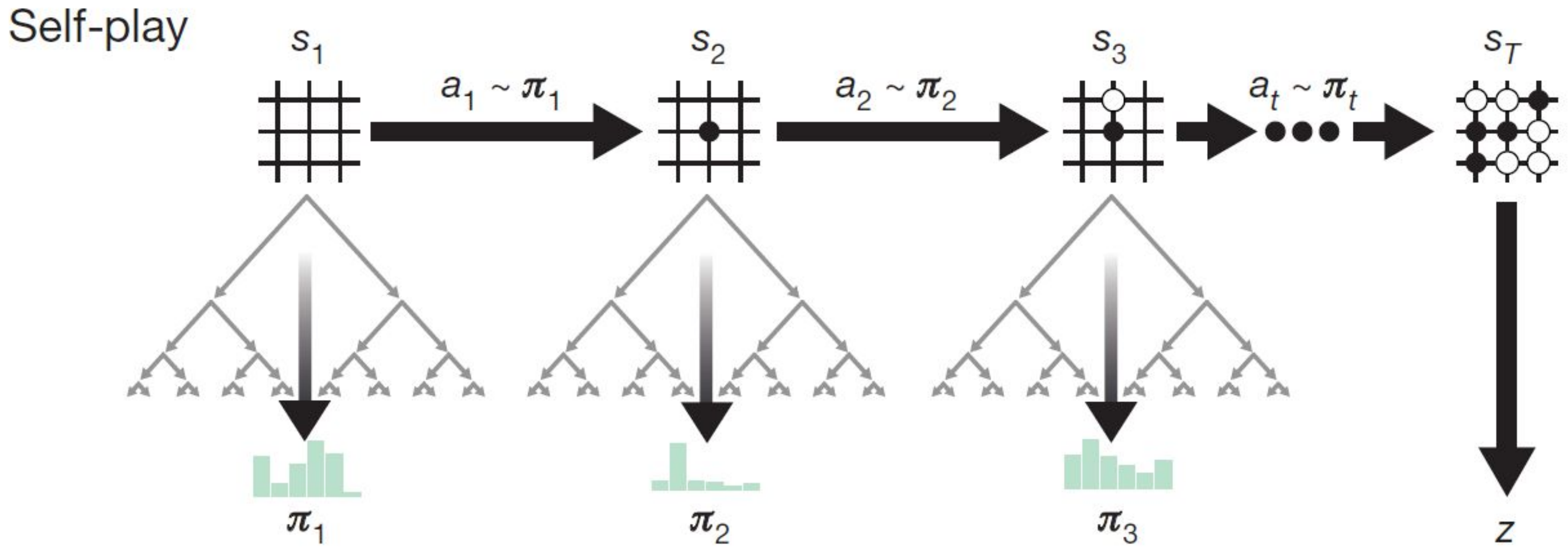
- No human data - only self-play
- No human features - only raw board image
- Single resnet instead of two separate networks
- No randomized Monte-Carlo rollouts

Alpha Go Zero: Architecture

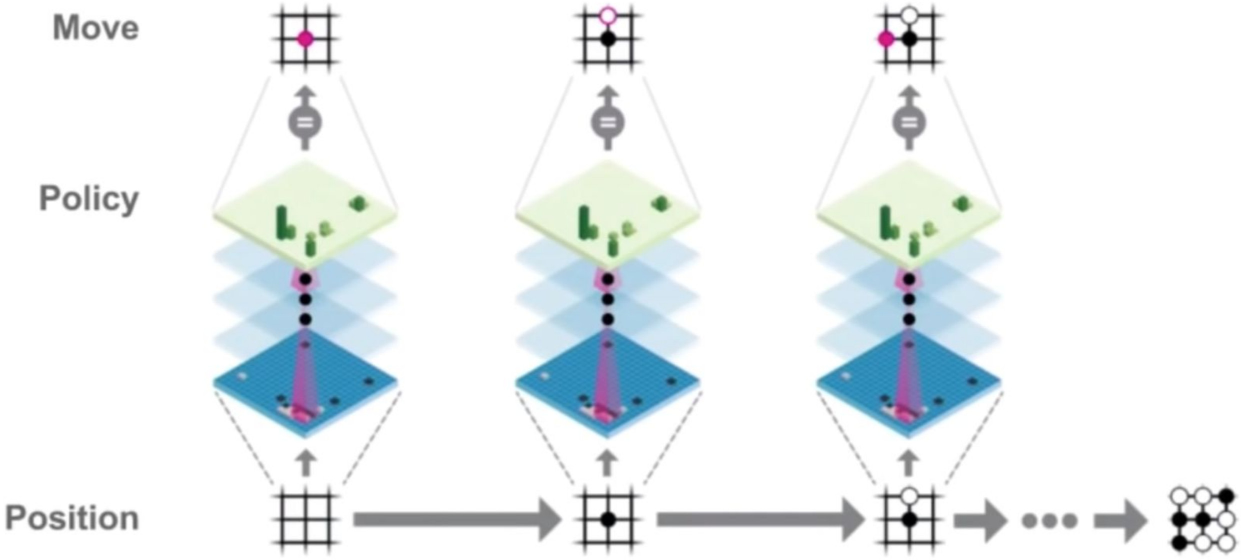
- 19x19 board
- 8 feature maps for white (with history)
- 8 feature maps for black (with history)
- 1 feature map for turn indication



Alpha Go Zero: Self Play

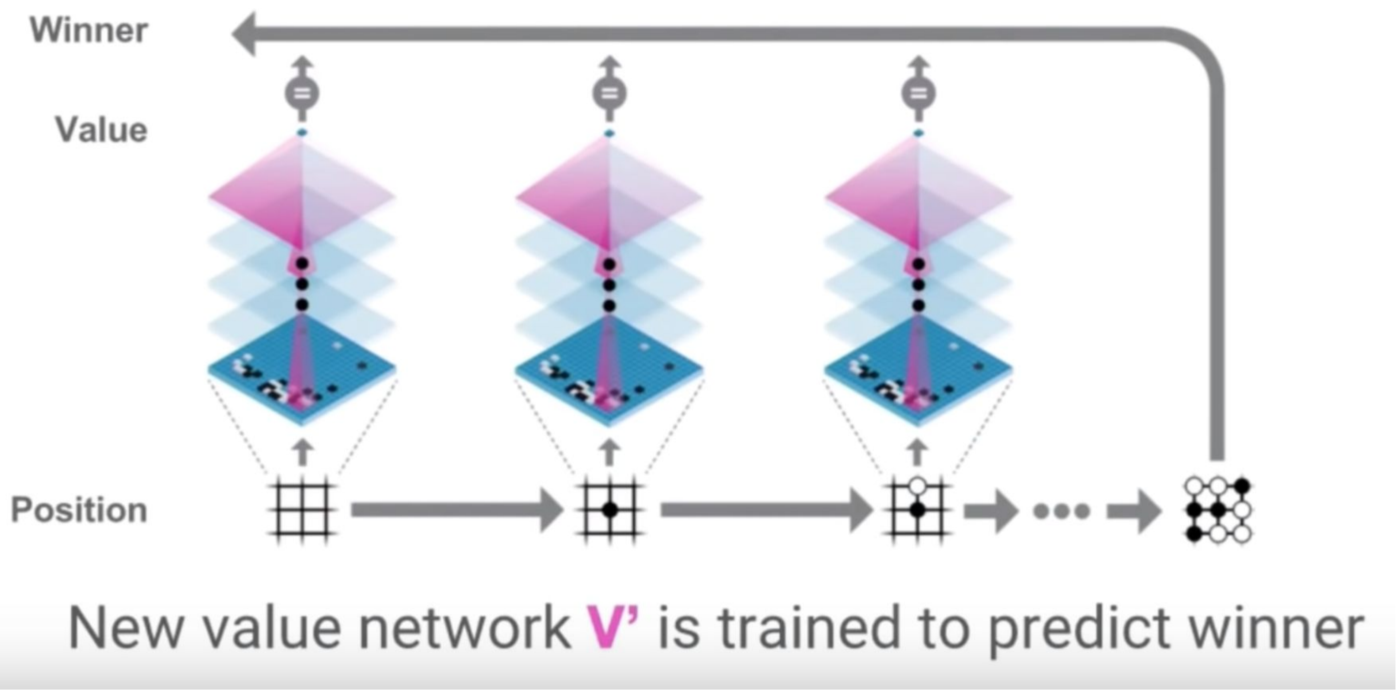


Alpha Go Zero: Update



New policy network P' is trained to predict AlphaGo's moves

Alpha Go Zero: Update



Performance

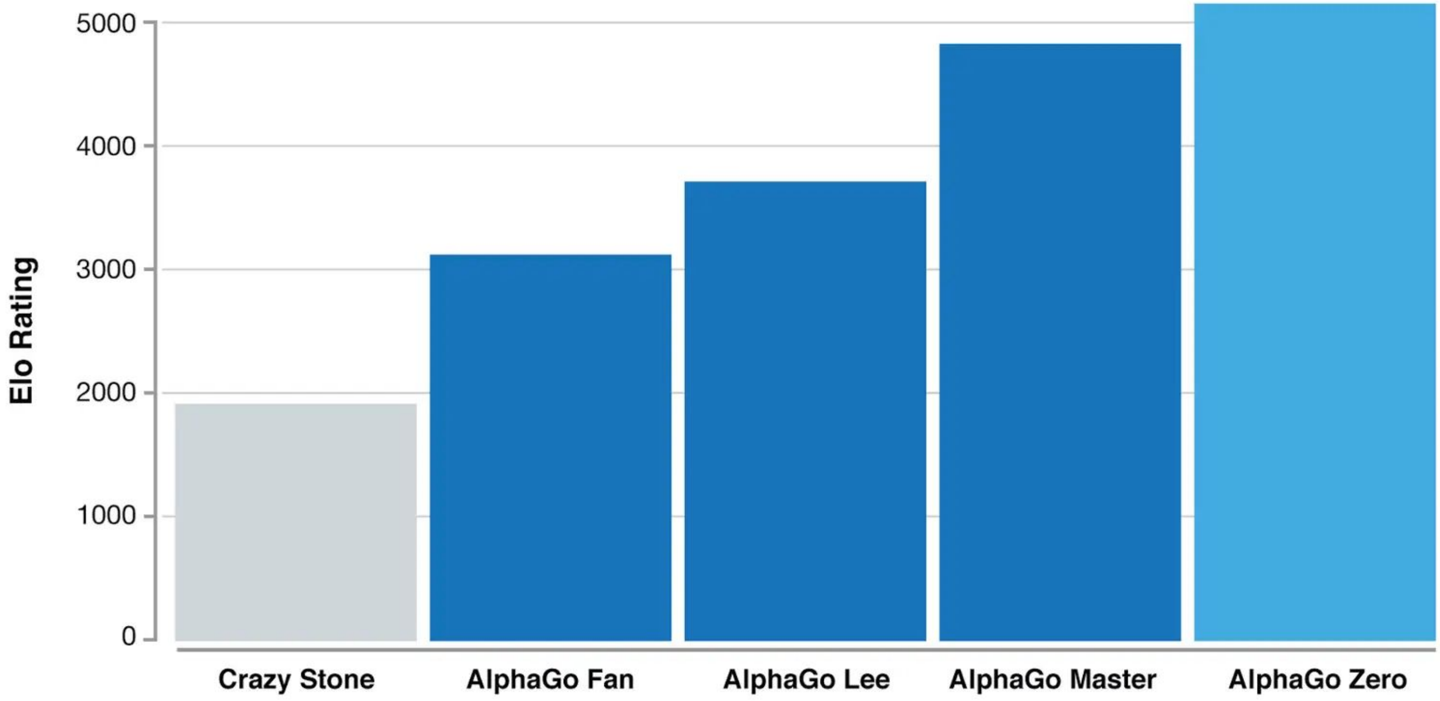


Image courtesy of Deepmind, NIPS 2017