CME 213

SPRING 2017

Eric Darve

Stanford University

MPI SUMMARY

- Point-to-point and collective communications
- Process mapping: across nodes and within a node (socket, NUMA domain, core, hardware thread)
- MPI buffers and deadlocks
- Blocking and non-blocking communications
- Linear algebra applications
- Groups, communicators, topologies
- Speed-up, efficiency, iso-efficiency

AlphaGo

AlphaGo

AlphaGo

- **We have seen previously the two main NNs used by AlphaGo:**
  - **Policy Network: used to predict the next move. The NN outputs a probability for each position that can be played at the next turn.**
  - **Value Network: estimate the strength of the current position, that is who is likely to win from this position.**
- **AlphaGo builds on neural networks but also uses other ideas, such as Monte-Carlo Tree Search.**

LEARNING

- Two types of learning mechanisms were used.
- **Supervised learning:** the NN for the policy network is trained to reproduce expert moves from the KGS Go server database.
- **Reinforcement learning.**
- AlphaGo plays against older versions of himself to improve the policy network.

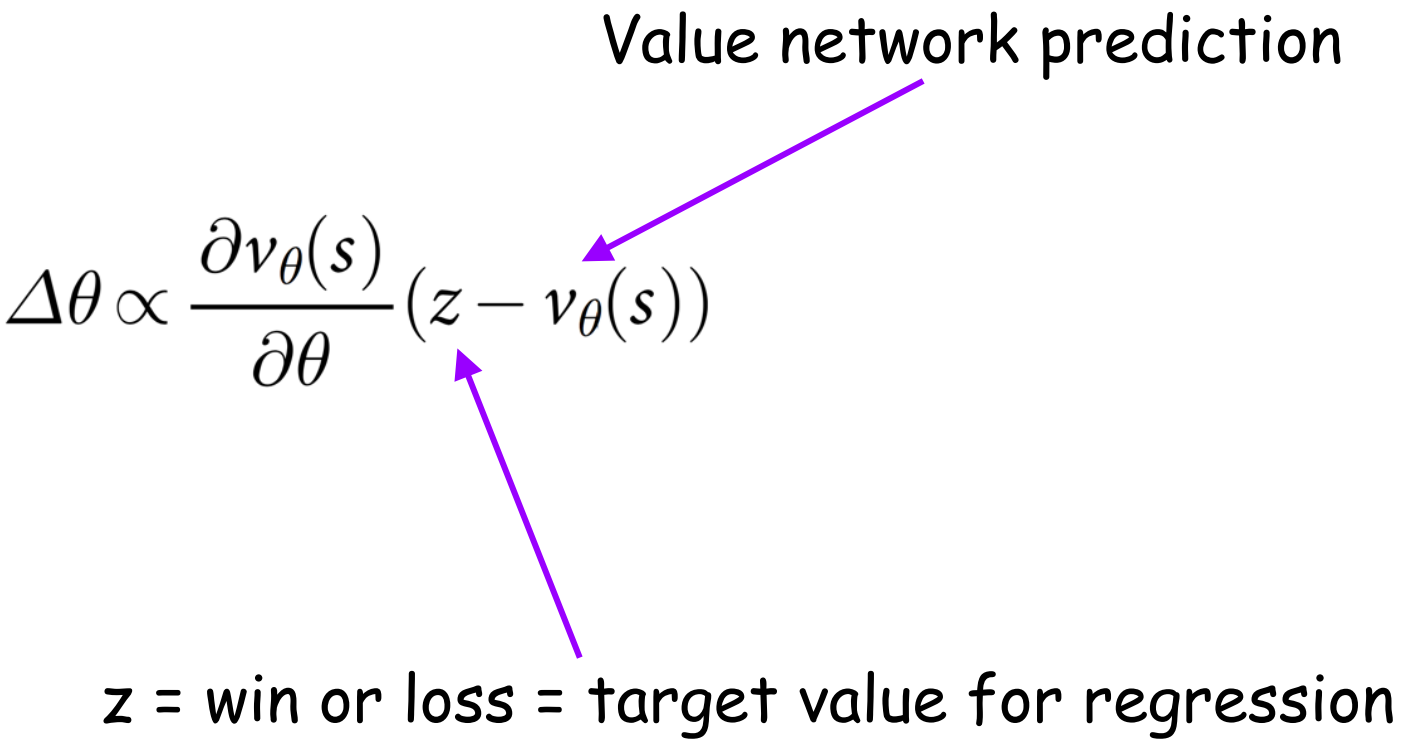
$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t \mid s_t)}{\partial \rho} z_t$$

Increase probability of move

+1 if win, -1 if lose

VALUE NETWORK

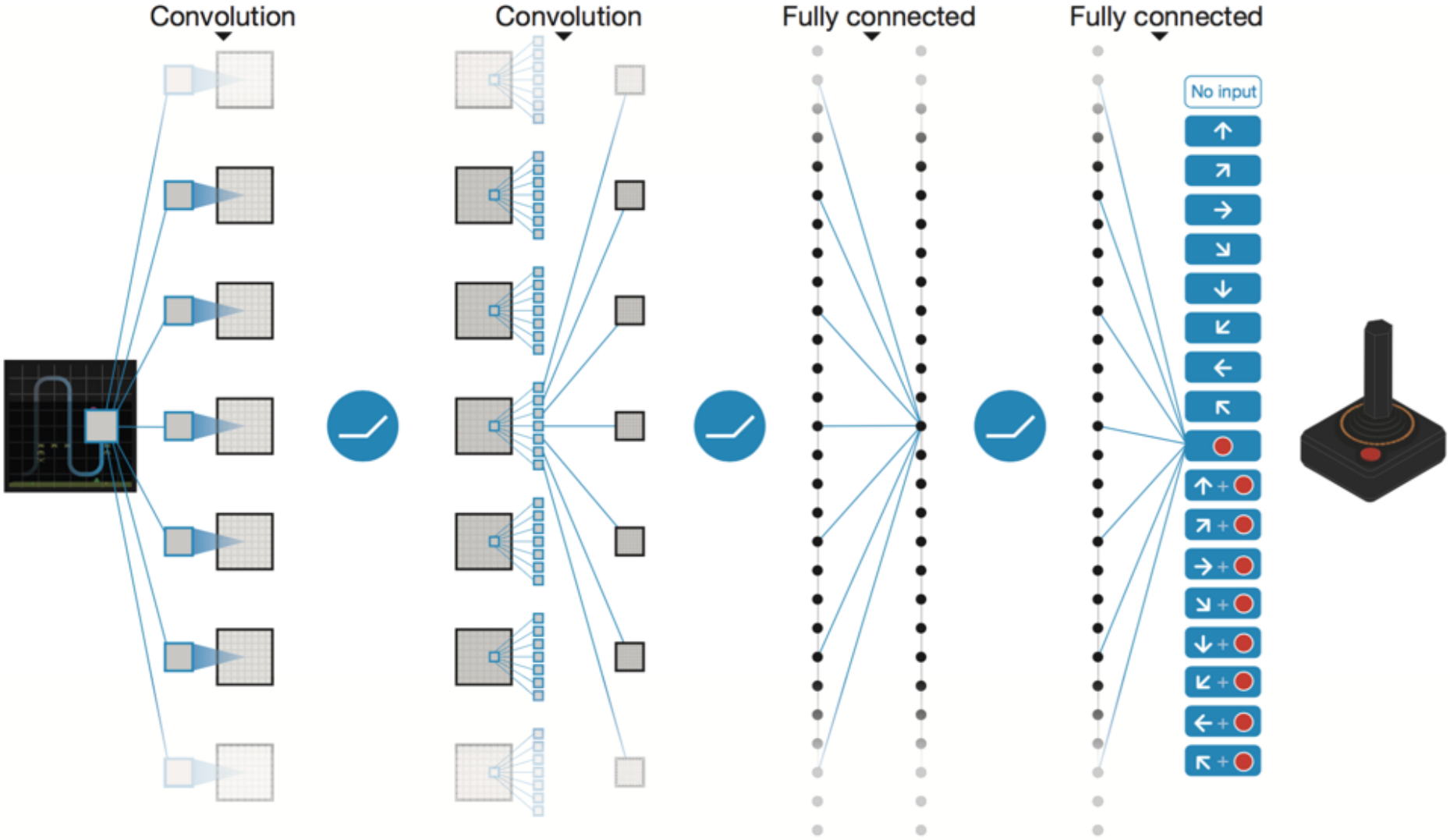
The value network is similarly updated using a regression technique.



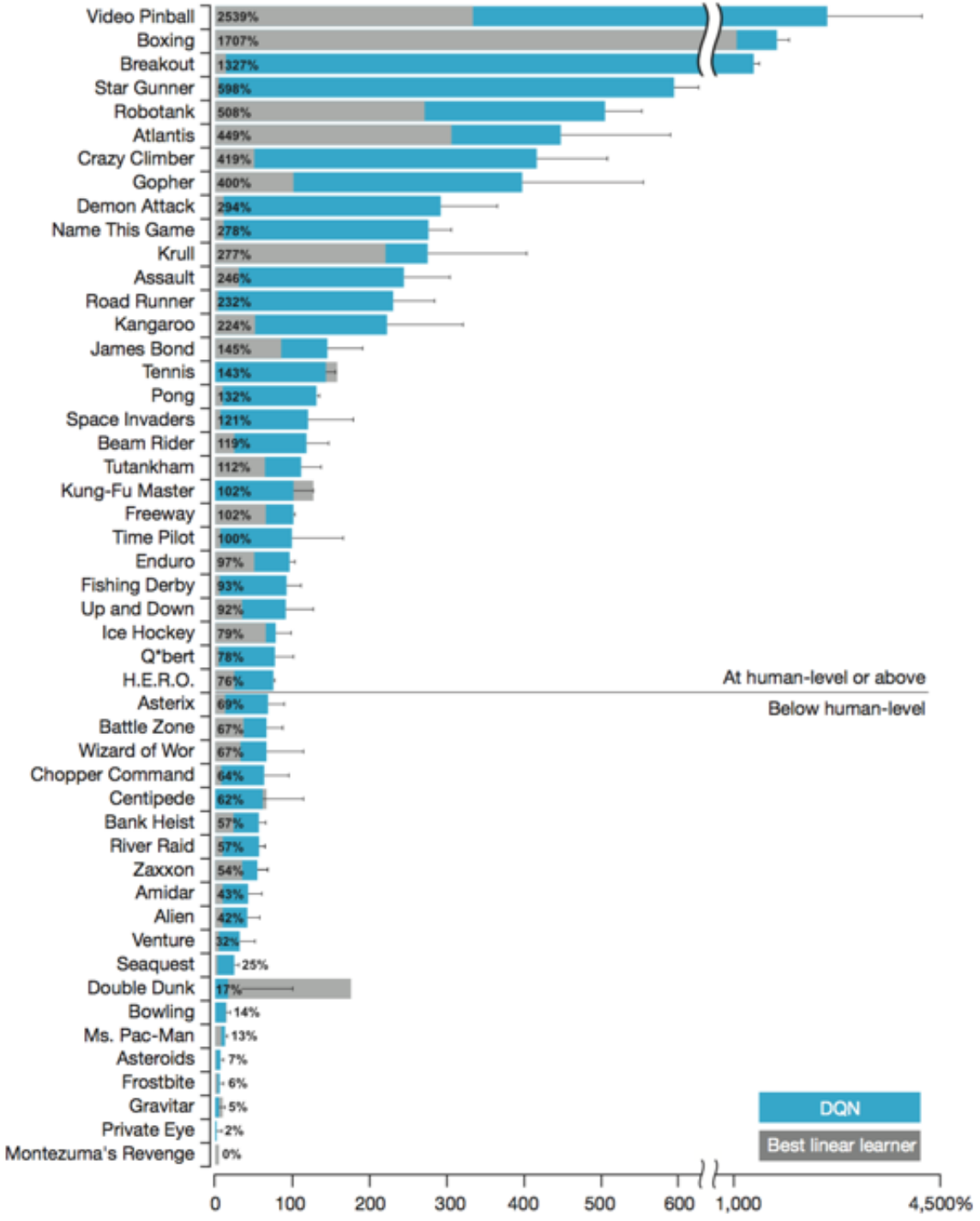
Other examples of ML for games

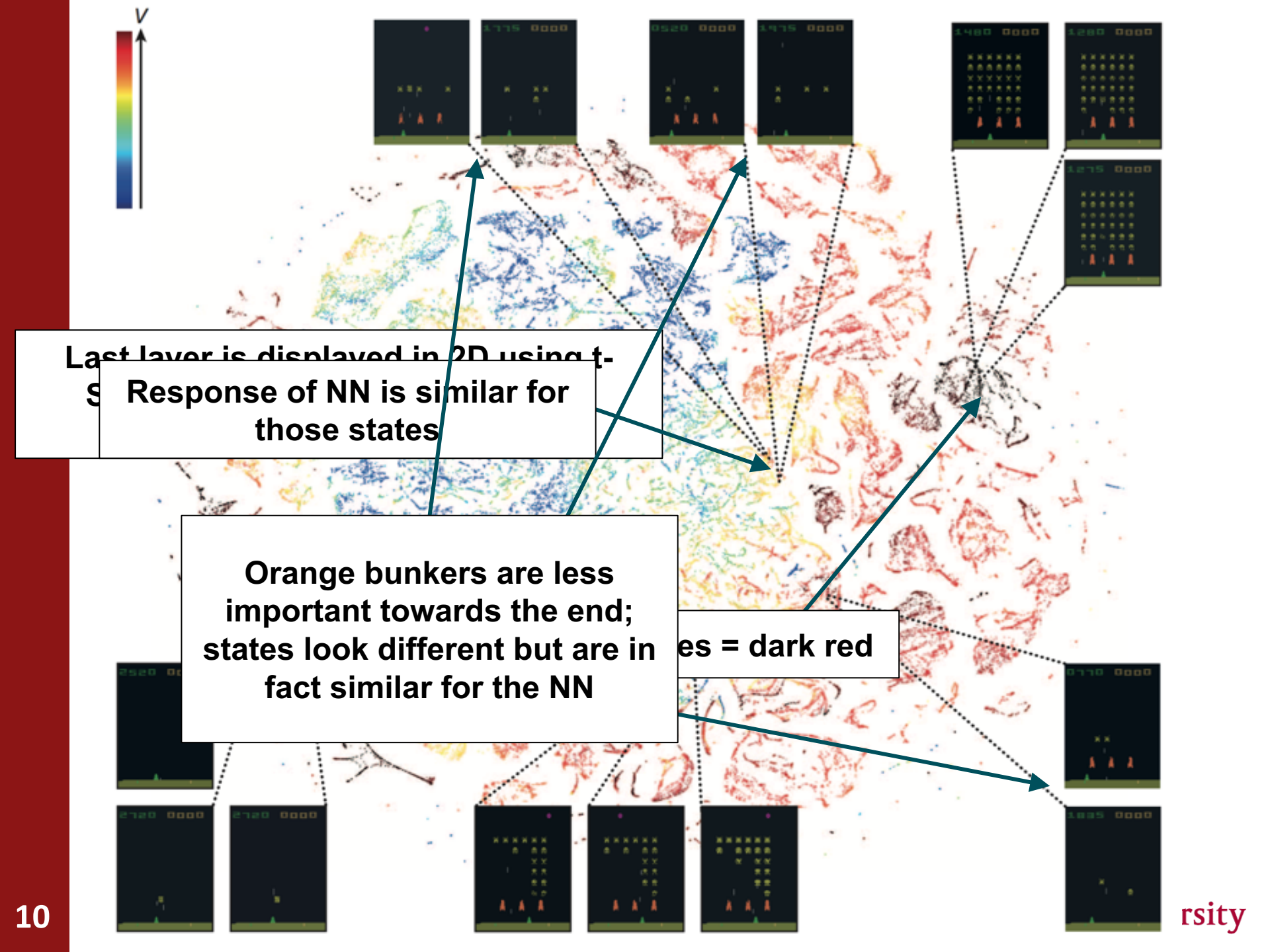
- NNs can be used for a variety of board and computer games.
- DeepRL = deep NNs with Reinforcement Learning
- DeepRL was used to master Atari 2600 games.
- DQN: Deep Q-network (Google DeepMind)
- Superhuman level was achieved using only the raw pixels and score as inputs.

SCHEMATIC VIEW OF NN



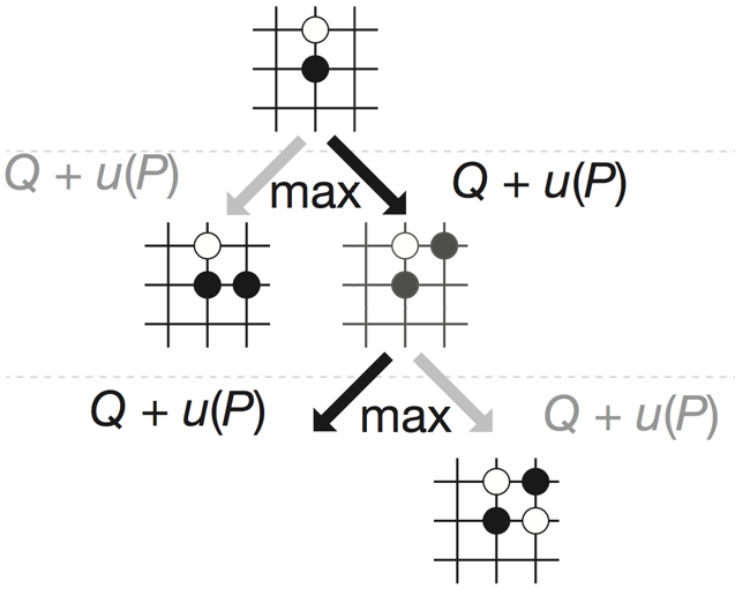
PERFORMANCE COMPARED TO HUMAN PLAYERS




V
Last layer is displayed in 2D using t-
S
Response of NN is similar for those states
Orange bunkers are less important towards the end; states look different but are in fact similar for the NN
es = dark red

NN IS NOT ENOUGH FOR GO; TREE SEARCH IS REQUIRED

- By itself, the policy network is not sufficient to achieve an accurate prediction.
- So the NN technique is coupled with a tree search algorithm.
- Tree search = search among all possible moves from the current position.
- The problem is that there are too many moves.
- So a Monte-Carlo strategy is used.



MONTE-CARLO TREE SEARCH

- **The idea is to explore the node that has the highest probability of winning.**
- **However, a strategy is in place that allows exploring other nodes as well.**
- **It's sometimes called *exploitation* and *exploration*.**
  - ***exploitation:* for a few moves with high average win rate, explore their sub-tree extensively**
  - ***exploration:* quick traversal of other nodes to make sure there is a sufficient variety of root branches being explored.**

is it better to switch from explo to exploitation?

Early

In the middle

Towards the end

Start the presentation to activate live content

If you see this message in presentation mode, install the add-in or get help at PollEv.com/app

Total Results: 0

TREE TRAVERSAL

- **The next move in the tree is picked by using**

$$a_t = \underset{a}{\operatorname{argmax}}\left(Q(s_t, a) + u(s_t, a)\right)$$

- **Initially, Q is 0. At convergence, Q becomes the probability that a is the best move.**

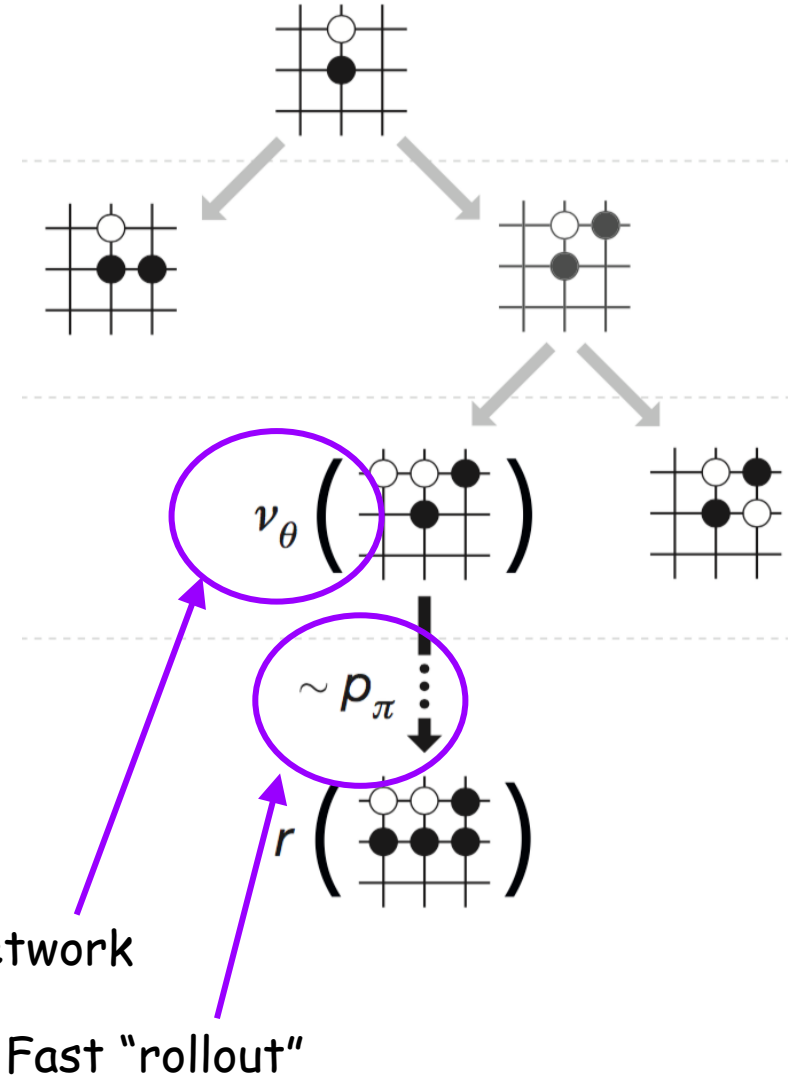
$$u(s,a) = c_{\text{puct}} P(s,a) \frac{\sqrt{\sum_b N_r(s,b)}}{1 + N_r(s,a)}$$

Number of visits

- **For nodes that are unvisited, u increases and can become large. This encourages the MCTS to visit these nodes (PUCT strategy).**
- **If they lead to a small Q, the node is ignored again. If not, the tree is further explored.**

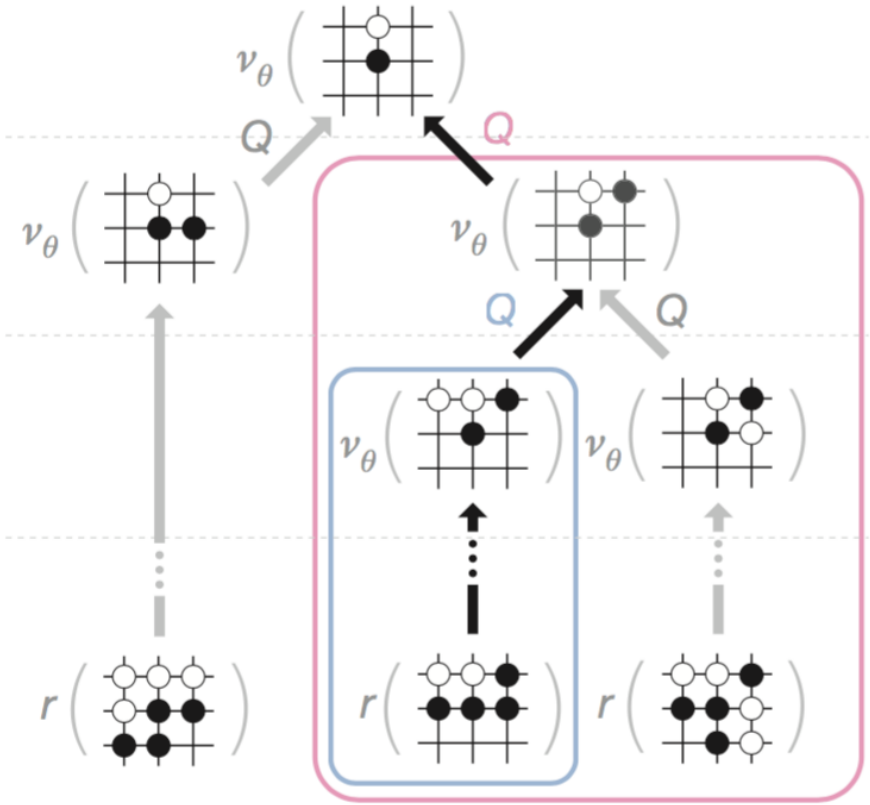
UPDATING NODE WEIGHTS

- The algorithm proceeds by visiting nodes according to the previous rule.
- When you reach a leaf, you estimate who the winner is going to be using the value network. There is also a fast procedure to run the game to the end that provides another estimate. Both are combined into a single number.



BACK UP

- **We update Q.**
- **Q is the probability that a move leads to winning the game. It is estimated using:**



Was edge (s,a) visited?

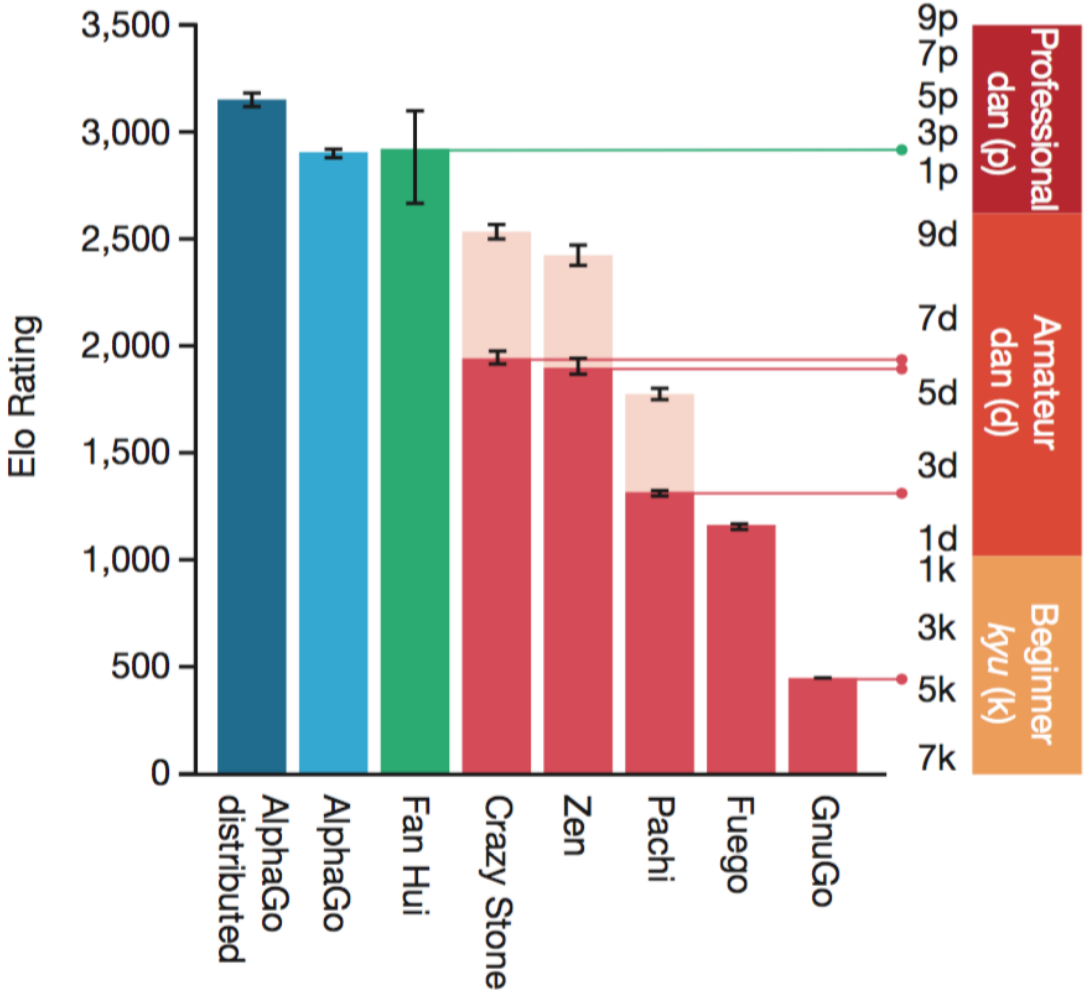
$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$

Did you win this game?

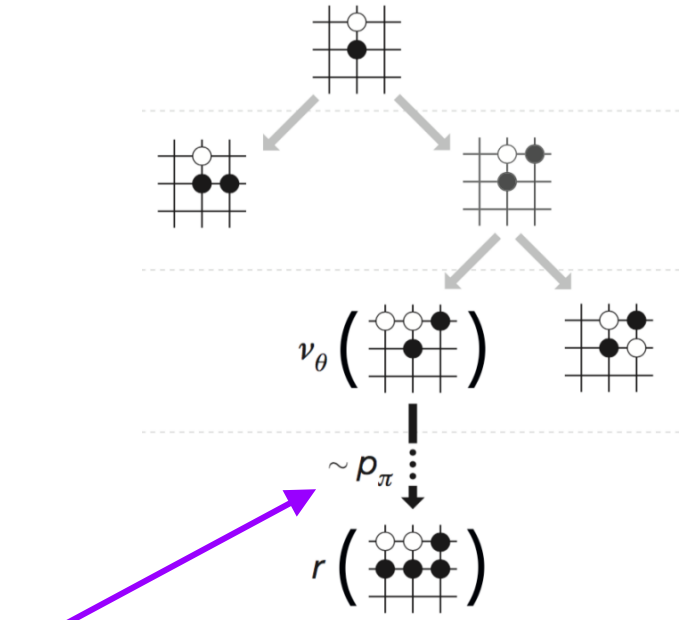
Normalization

How good is AlphaGo?



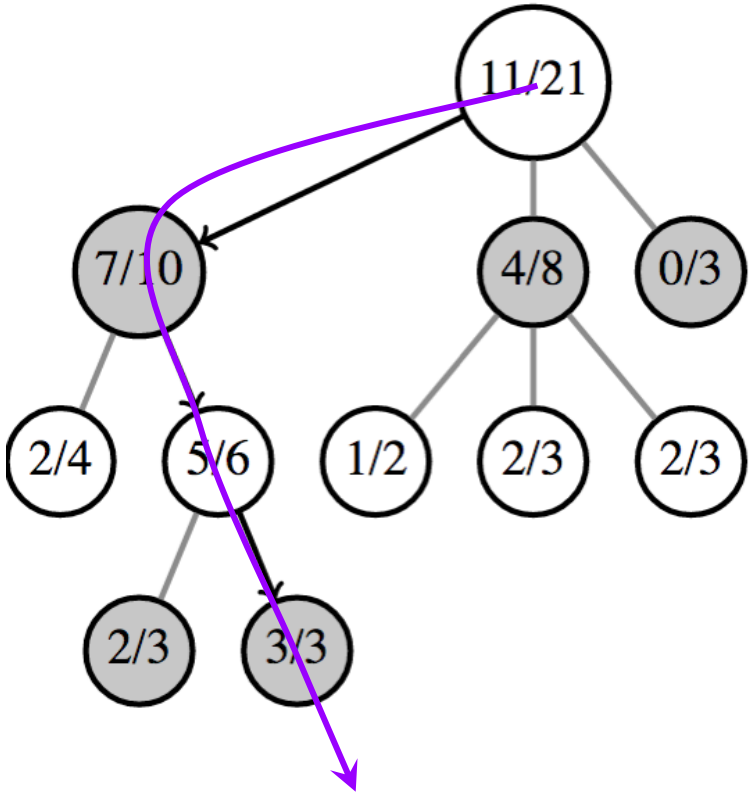
PARALLELIZATION OF MCTS

- We cover not just parallelization for AlphaGo but also general parallelization for Monte-Carlo Tree Search.
- In most MCTS, the game is played using a random strategy.
- AlphaGo uses a strategy that is only partially random: the fast random rollout to determine the winner from the current position.
- How do you distribute the work and collect in parallel the results from simulations, possibly asynchronously?



THE 4 BASIS MCTS OPERATIONS: SELECTION

- **Traverse the tree using some strategy to select the most promising moves.**
- **AlphaGo: uses argmax to find the best move.**



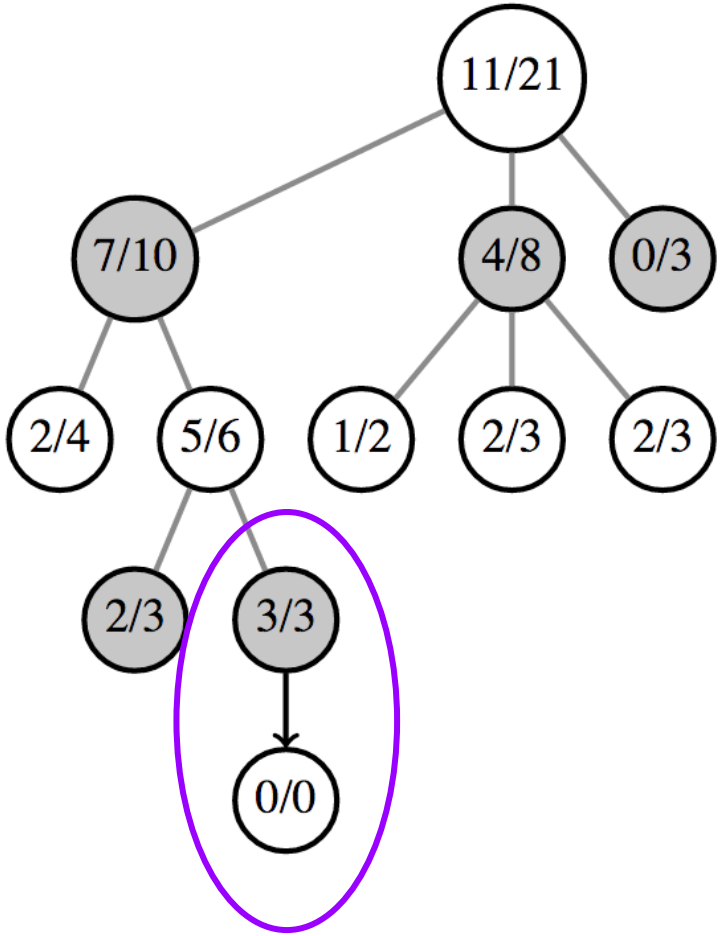
Operation 2: Expansion

Add new nodes in the tree to encourage exploration.

AlphaGo:

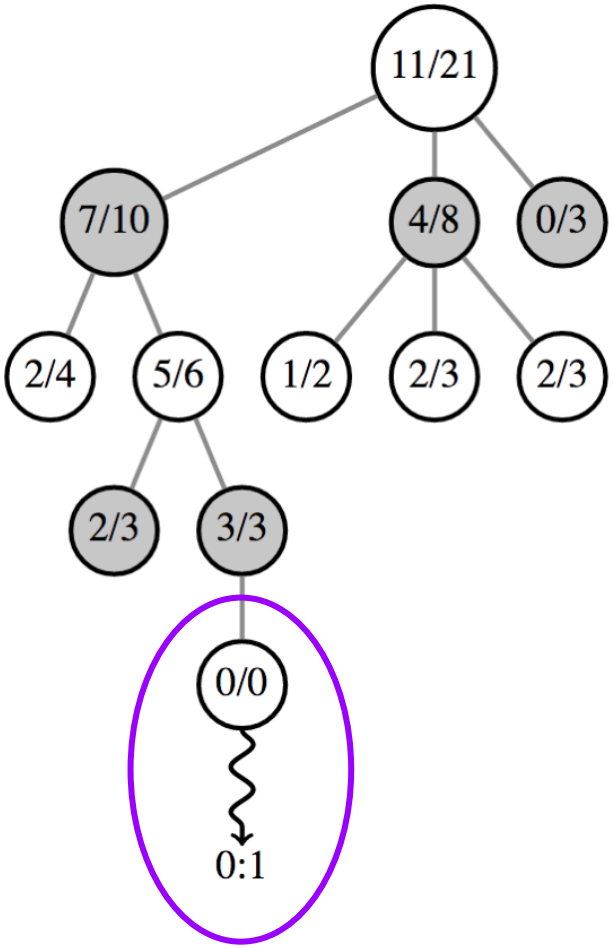
- **new nodes added depending on argmax value**
- **add a new sibling when an edge has been explored too often.**

This encourages the appearance of new nodes in the tree.



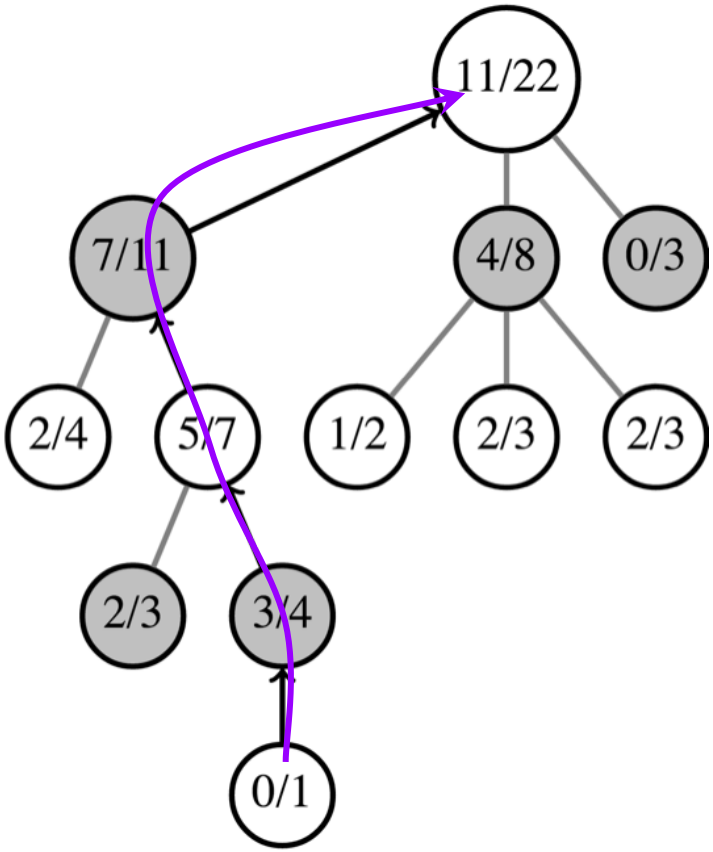
Operation 3: simulation from leaf node

- Evaluate a leaf position by playing the game from that node.
- Alphago: combination of value NN evaluation + random rollouts.



OPERATION 4: BACK PROPAGATION

Back propagate the result of the leaf simulation to update the tree.



PARALLEL STRATEGY 1: ROOT PARALLELIZATION

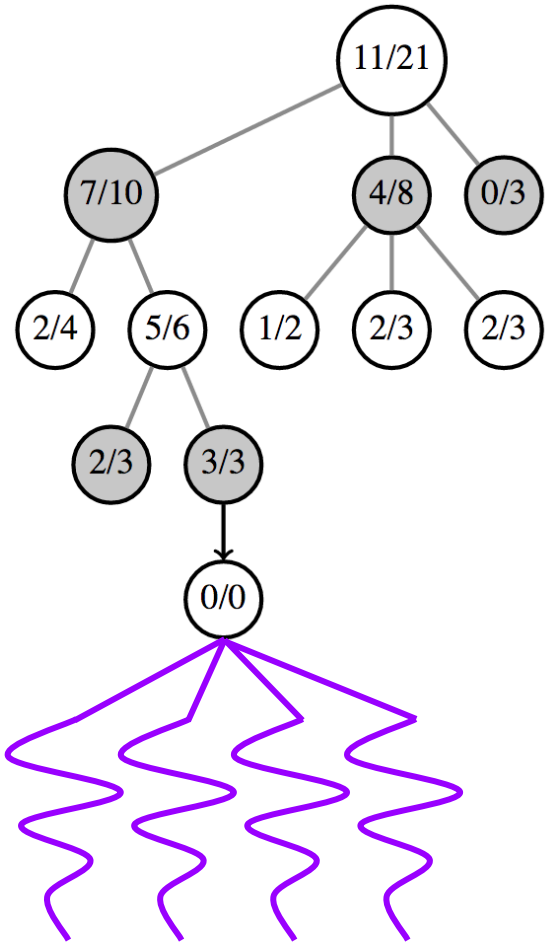
- The simplest!
- Each thread starts simulations from the root of the tree.
- Because the simulation is random, all threads traverse the tree slightly differently.
- When all simulations are complete, statistics are gathered for the root to determine the correct move.

PARALLEL STRATEGY 2: TREE PARALLELIZATION

- **The most complex!**
- **The 4 operations in MCTS are executed in parallel using mutexes.**
- **This allows accessing and updating the tree in a consistent manner.**
- **A virtual loss is sometimes used to avoid threads going down the same path.**
  - **If many threads traverse the tree at the same time, they may end up traversing the same nodes.**
  - **To avoid this, whenever a thread (call it Thread0) goes down the tree it records a fictitious loss (virtual loss).**
  - **As a result other threads tend to visit other nodes.**
  - **When Thread0 has determined the actual win or loss, it back propagates this information and corrects the virtual loss.**

PARALLEL STRATEGY 3: LEAF PARALLELIZATION

This is the closest one to AlphaGo.



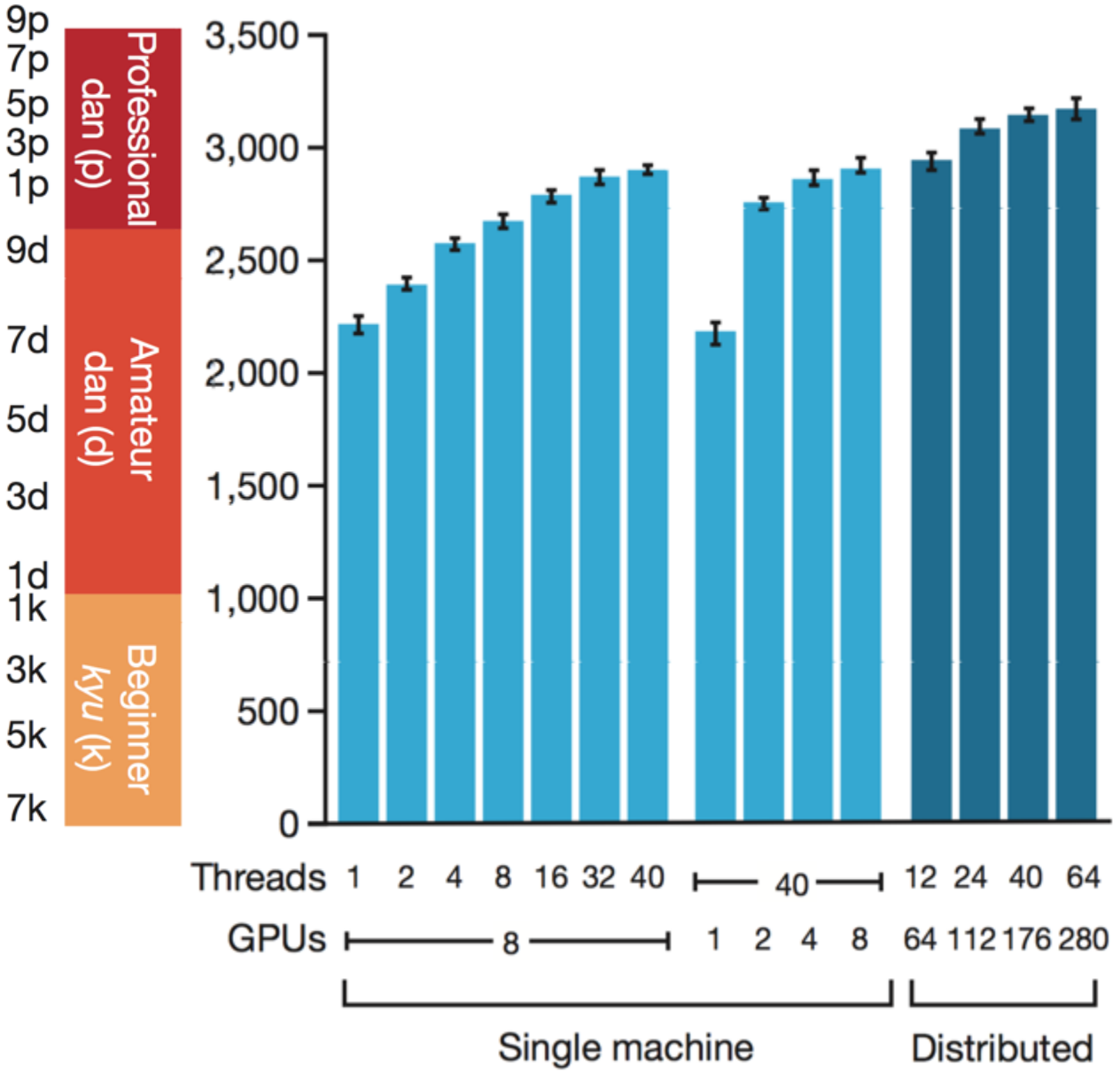
Spawn worker tasks at a leaf to evaluate the win/loss probability

AlphaGo parallelization

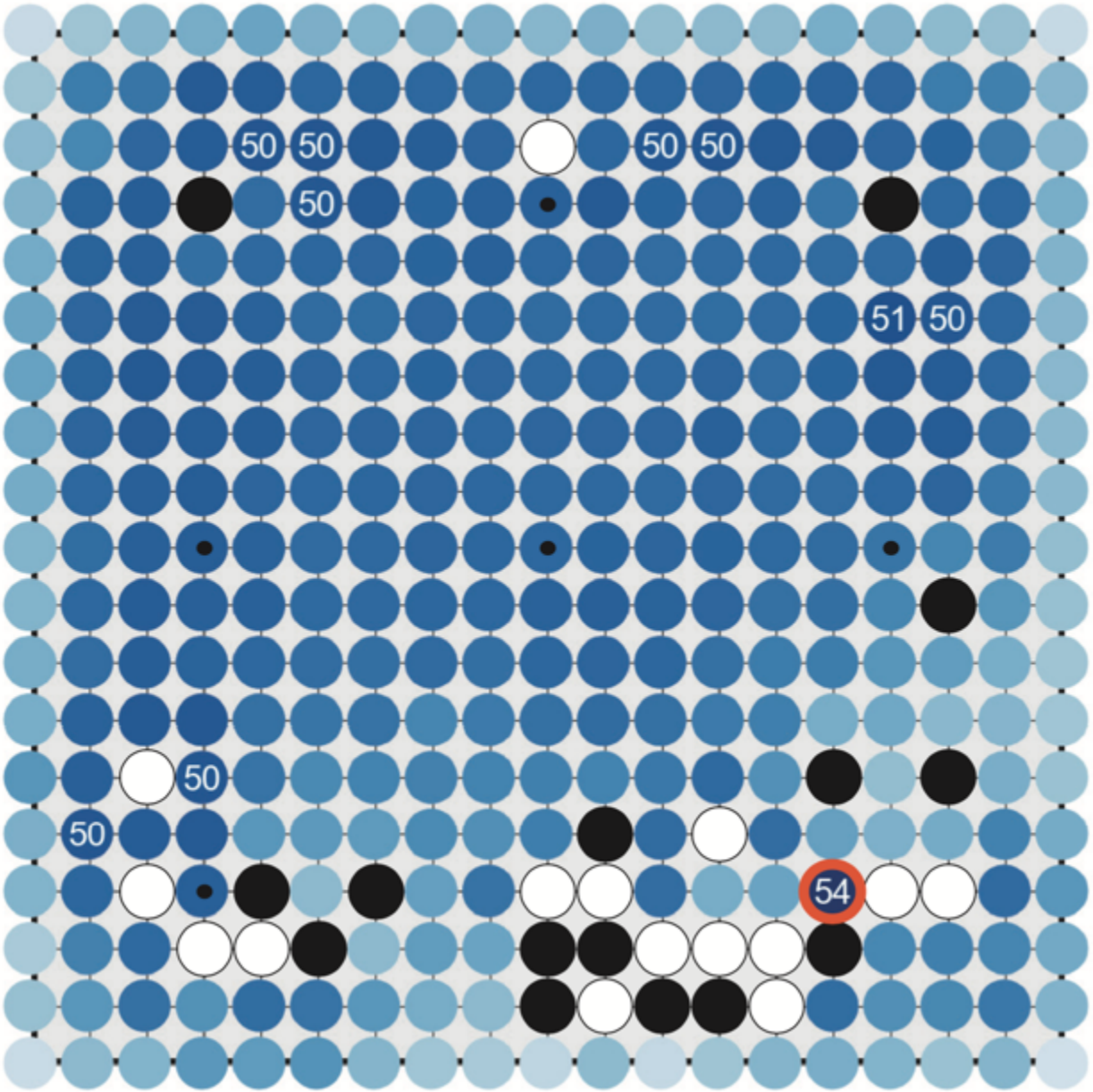
- It uses a combination of these techniques.
- A single search tree is stored on the master thread.
  - CPU: random rollouts simulations.
  - GPU: policy and value NN evaluations.
- GPUs were also used heavily to train the NNs.
- Asynchronous updates: virtual losses are used in the parallelization procedure.
  - The master thread assigns a virtual loss.
  - Then, it launches a worker task to compute the actual win/loss.
  - Once the result comes back, the tree is updated.
- This prevents too many threads from exploring the same path.

ELO OF ALPHAGO

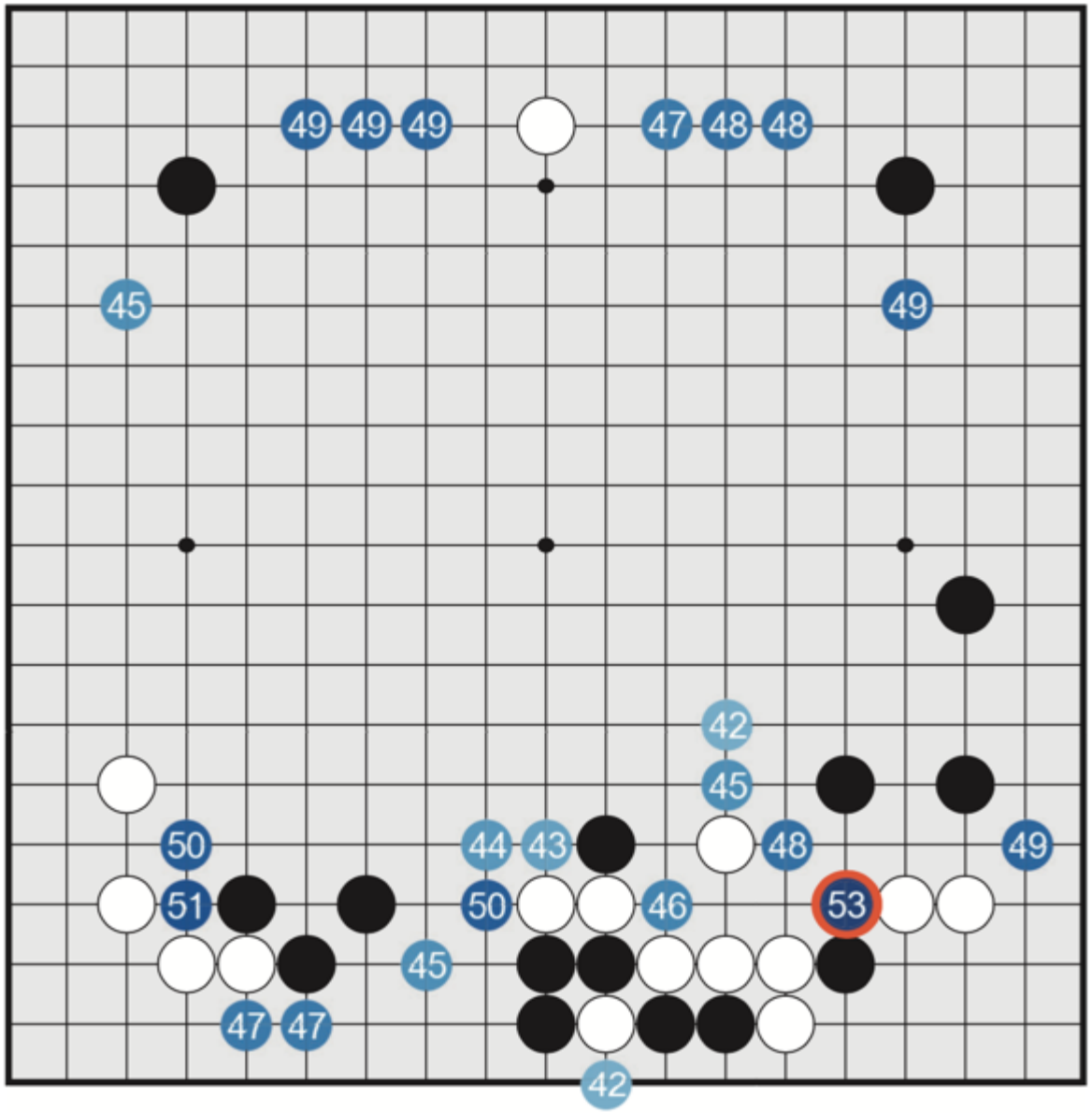
Elo rating as a function of the number of concurrent CPU cores and GPUs used.



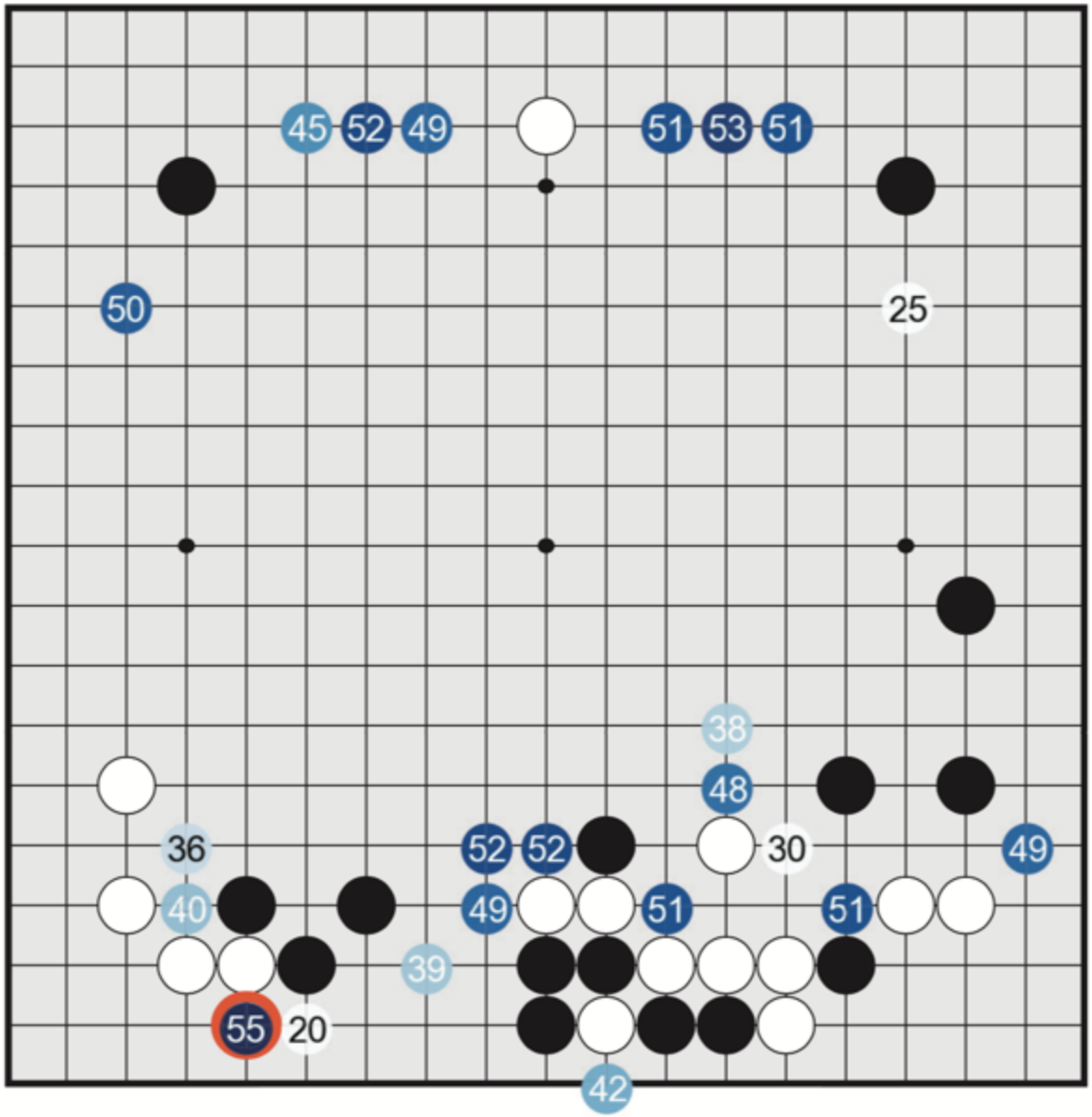
EXAMPLE OF PLAY – INFORMAL GAME AGAINST FAN HUI – VALUE NETWORK



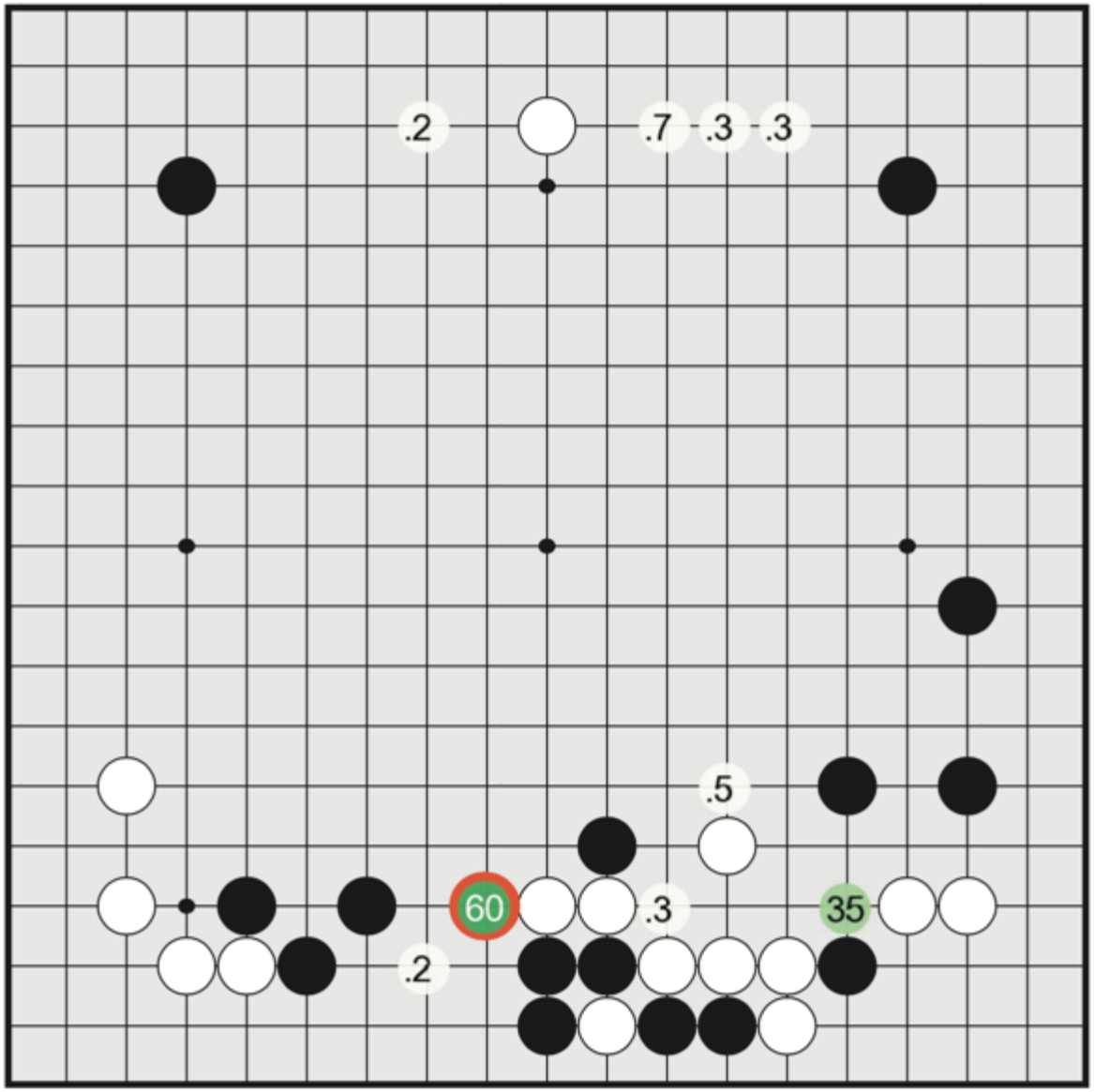
Evaluation using Tree and value NN only



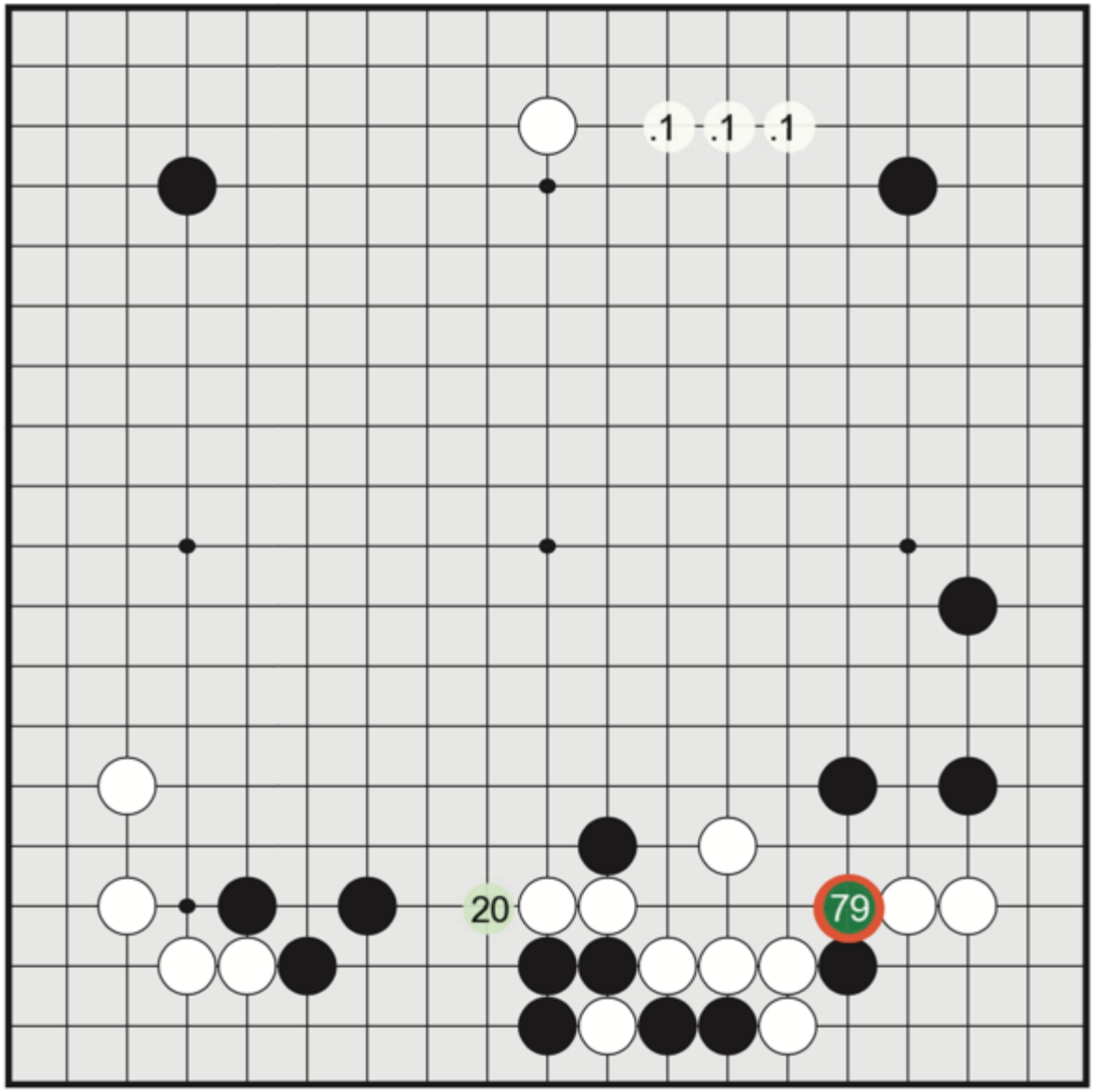
EVALUATION USING TREE AND RANDOM ROLLOUTS ONLY



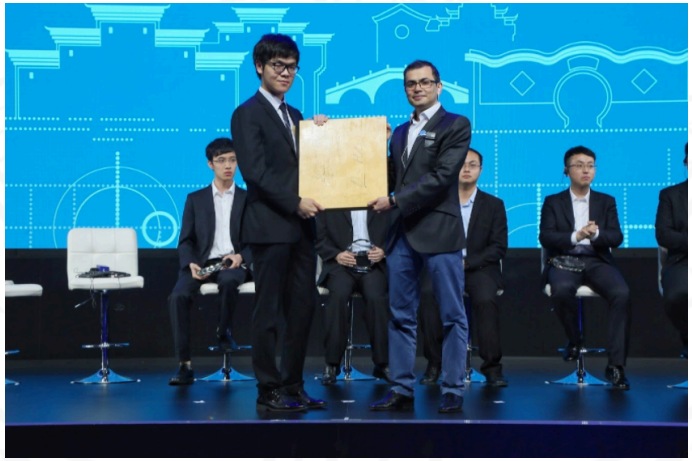
POLICY NN



NUMBER OF TIMES A MOVE WAS VISITED



"This Summit is one of the greatest matches that I've had. I believe, it's actually one of the greatest matches in history." — Ke Jie, 9 Dan Professional, during the post match wrap up



Have you enjoyed CME213?

Yes **A**

Yes **B**

Start the presentation to activate live content

If you see this message in presentation mode, install the add-in or get help at PollEv.com/app

Total Results: 0

"That's all Folks!"