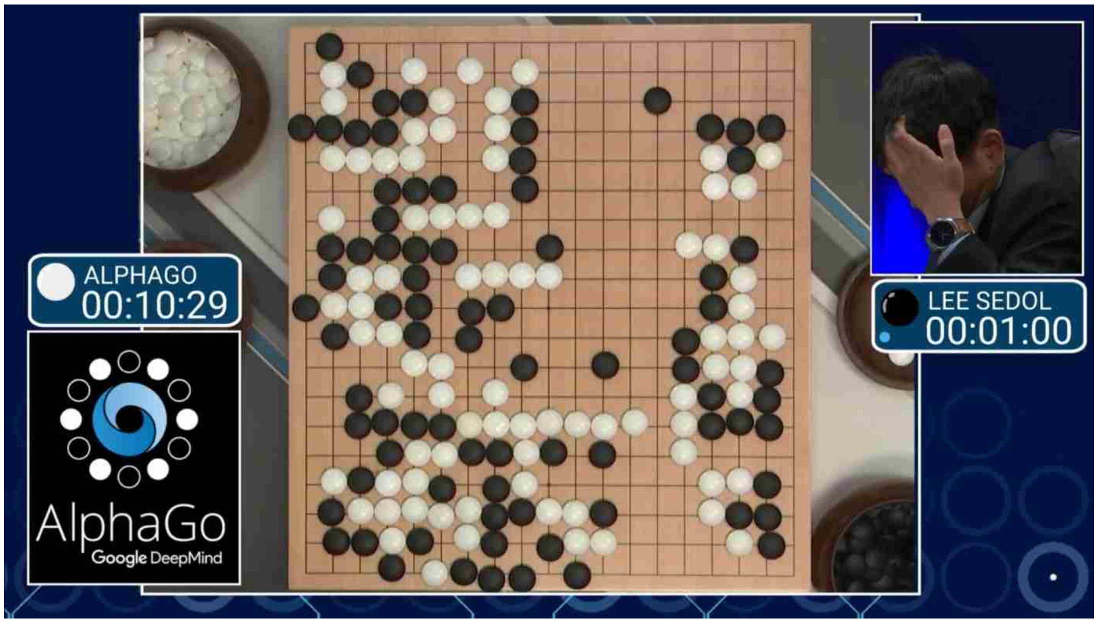
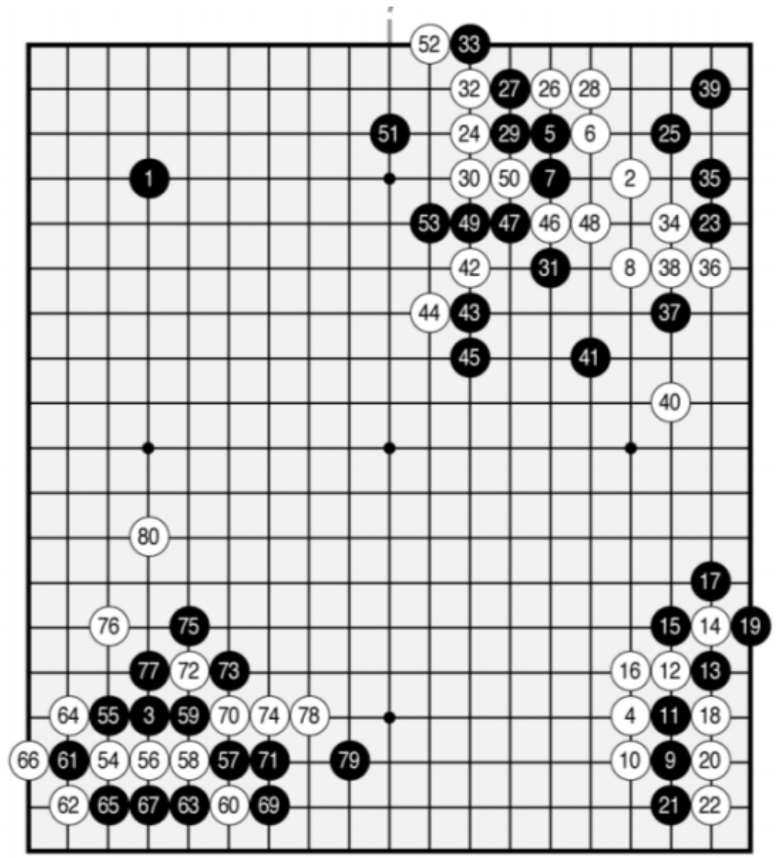
AlphaGo



Jonathan Hui

https://medium.com/@jonathan_hui/alphago-how-it-works-technically-26ddcc085319

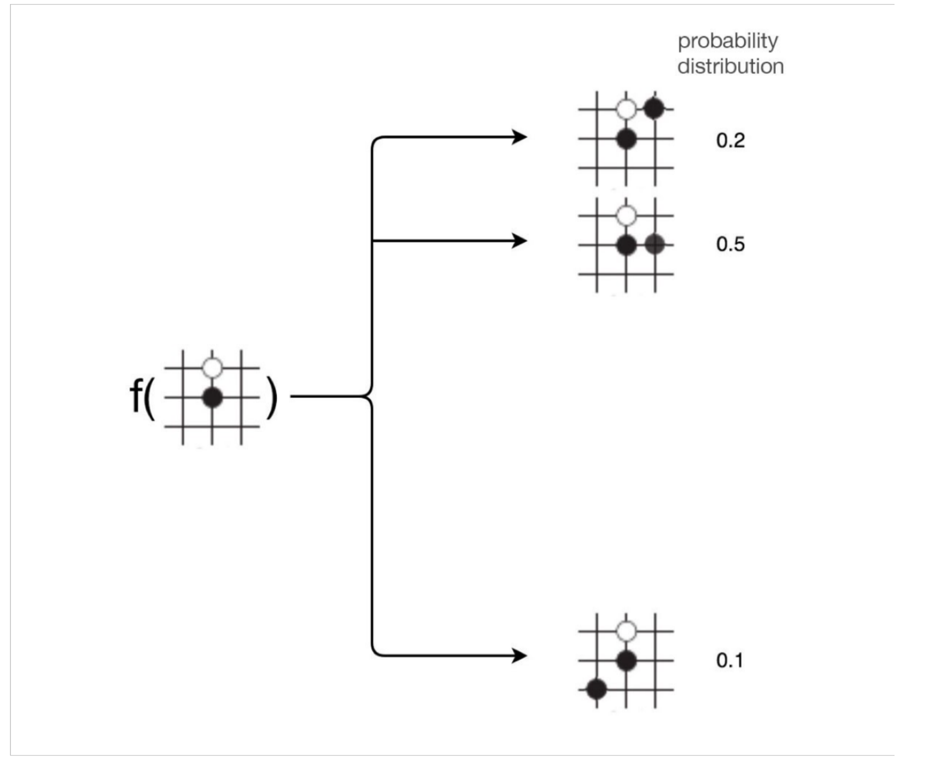
Go – surround territory on 19x19 board



68 at 61

Learn policy

- **a** = f(**s**)
- **a** = where to play (19*19)
- **s** = description of board



state ~ 19*19*48

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

Train “SL policy network”

- 13 layers of convolutional filters and rectifiers
  - softmax classifier
- **Train** using moves for 30 million board positions
- 50 GPUs for 3 weeks

Approximate using rollout policy net

- SL policy network (55.7% accuracy) **3 ms**
- Rollout policy network (24.2% accuracy) **2 μs**
  - Also trained on human expert positions

RL policy network

- **Initialize with SL policy network**
- **Train using self-play with older network**
- Let $z_t = 1$ if we win game or -1 if we lose

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

policy gradient RL

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a \mid s)}{\partial \sigma}$$

DL backprogagation

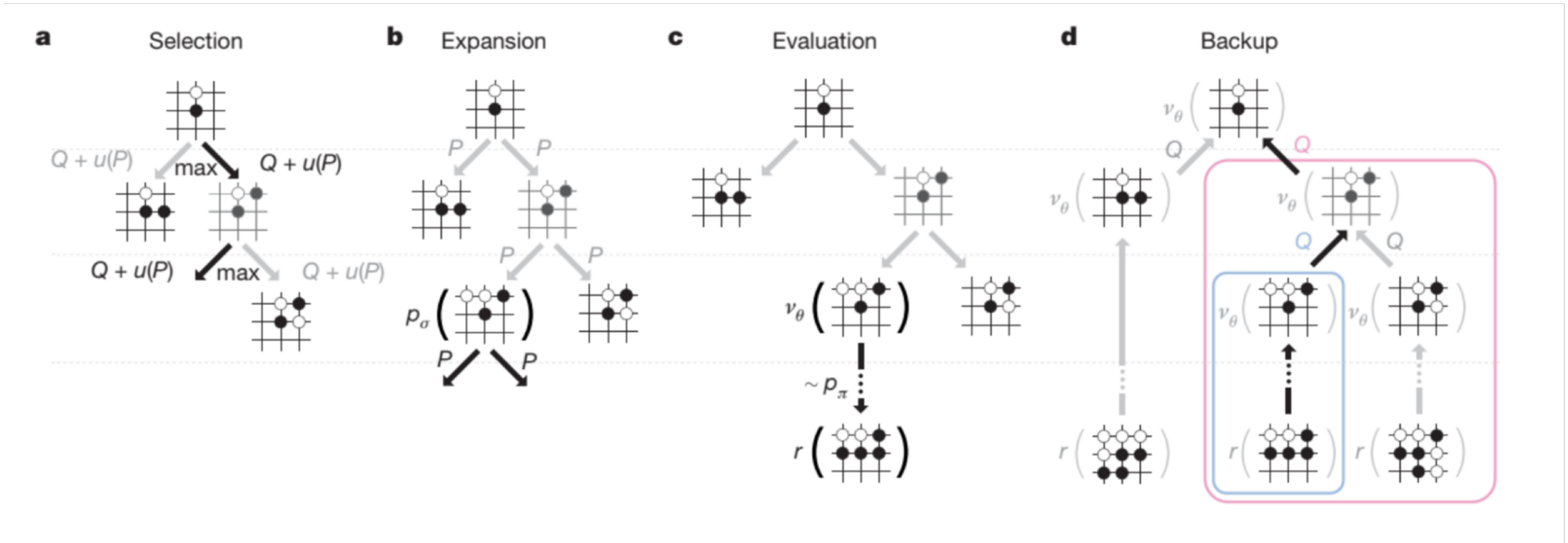
Value network

- **Estimate value of board state under the policy followed by the policy network**
- **Input:** one board from each self-play game
- **Output:** z (win/loss for that game)
- again, 50 GPUs for one week.

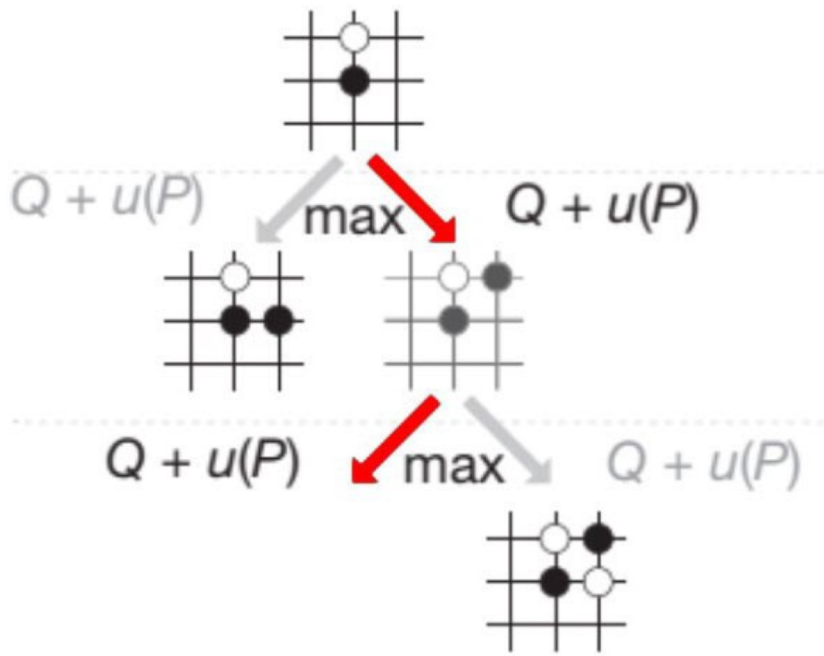
Monte Carlo Tree Search

- **Need to trade off exploration and exploitation**
  - But can’t afford to do a full search
- **Use**
  - predictions from the policy and value networks
  - how many times we have picked the move
  - simulated game results with wins.

Monte Carlo Tree Search



Selection



$$a_t = \underset{a}{\operatorname{argmax}}(\underbrace{Q(s_t, a)}_{\text{exploration}} + \underbrace{u(s_t, a)}_{\text{exploitation}})$$

$$u(s,a) \propto \frac{P(s,a)}{1 + N(s,a)}$$

$$P(s,a) = p_\sigma(a|s)$$ **SL policy net**

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

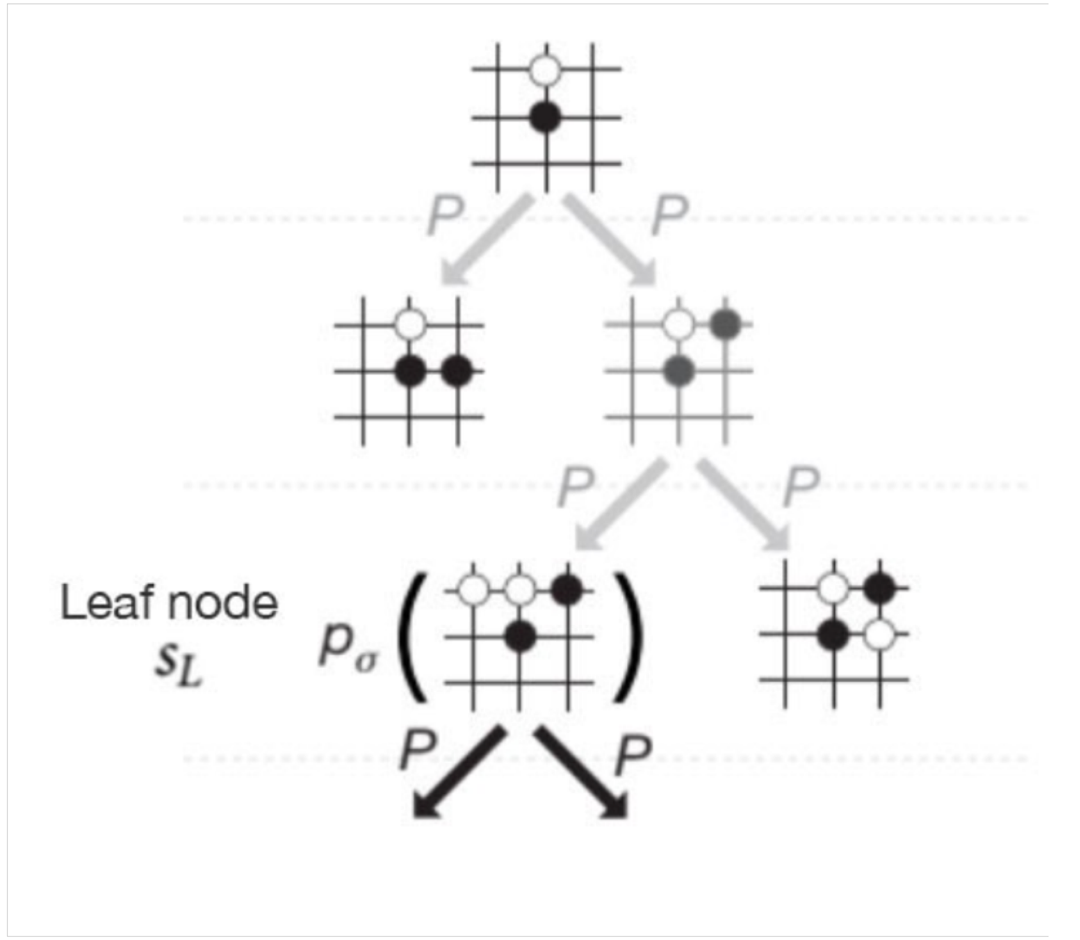
$$V(s_L) = (1-\lambda)\underbrace{v_\theta(s_L)}_{\text{value net}} + \lambda \underbrace{z_L}_{\text{previous game result}}$$

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

$p_\sigma(a|s)$ - From the policy network: how good to take action a.
$v_\theta(s_L)$ - From the value network: how good to be in positions s_L.
$N(s,a)$ - How many times have we select action a so far.
z_L - the previous simulated game result.

Expansion

- ***Q*** from RL value net
  - more accurate
  - use for exploitation
- ***P*** from SL policy network
  - more diverse
  - use for exploration (***u***)



Evaluation

- Simulate the rest of the game using **Monte Carlo Rollout** starting from the leaf node
- Sample moves using the rollout policy.
  - Use the fast (but inaccurate) rollout net
    - 1500x faster
- Predicts a win or a loss z_L

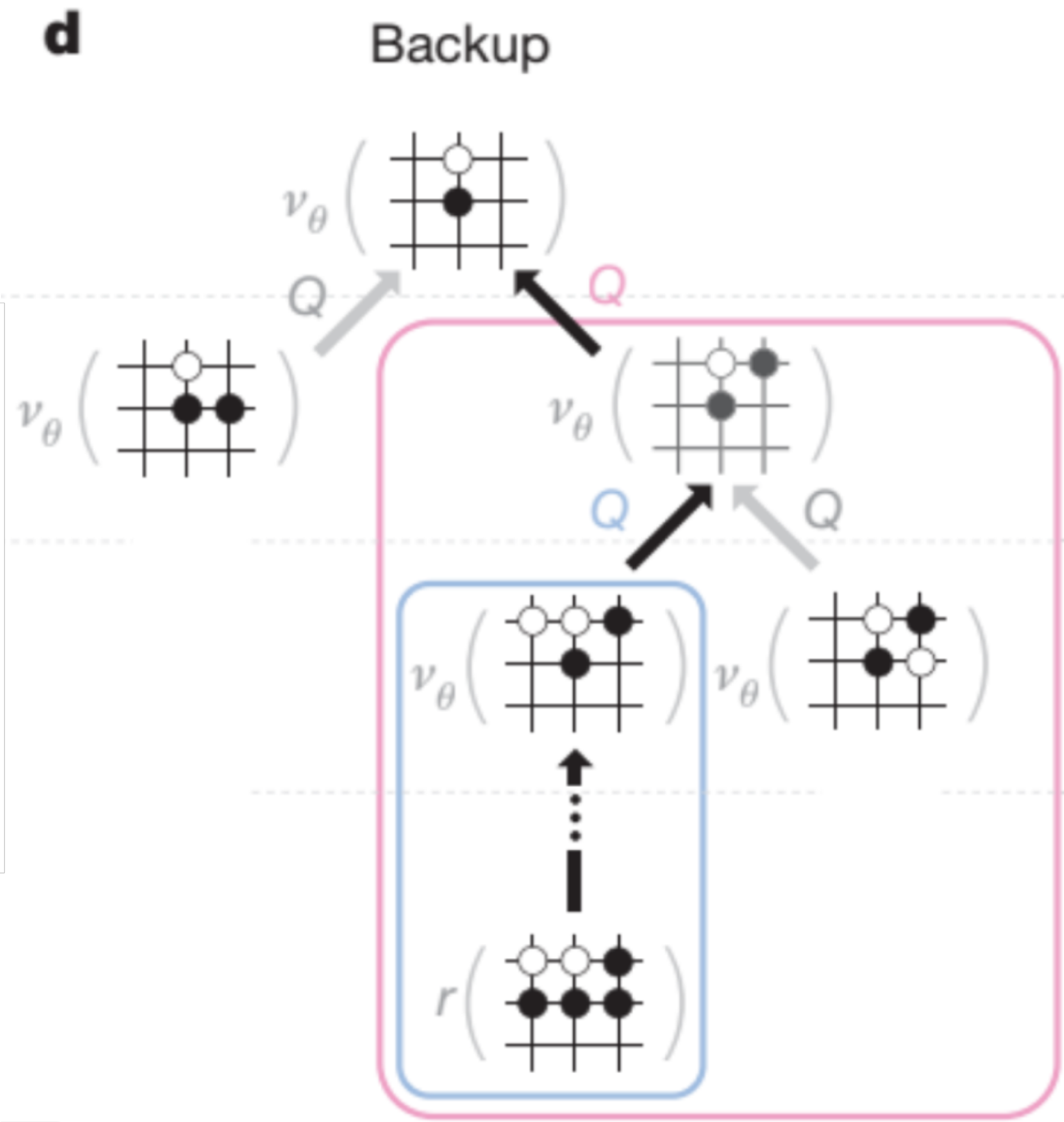
Backup

- Update $\boldsymbol{Q}$ with

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

$$V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$$

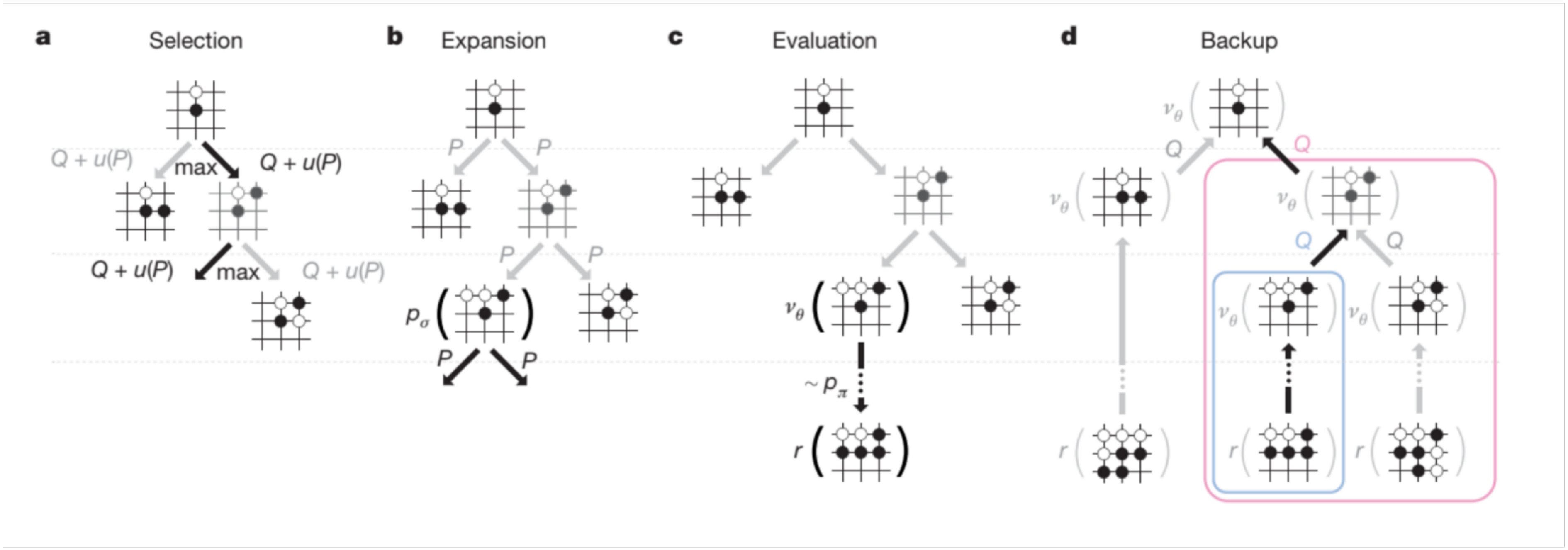
$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$$



Picking the next move

- Could use Q(s,a) but don't
- Use the move that was most often picked for the current board position
    - Leads to increasing exploitation over time

Monte Carlo Tree Search



AlphaGo take-aways

- **Boot-strap**
  - Start with policy learned from human play
- **Self-play**
- **Speed matters**
  - Rollout network
  - Monte Carlo search
- **It still helps to have fast computers**
  - 100 GPU weeks