Mastering the game of Go without human knowledge

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel , Demis Hassabis

DeepMind London

Nature / Presenter: Ji Gao

Outline

1. Overview
   - AlphaGo
   - Rules of go
2. Method Overview
   - Play
   - Train
3. Method
   - Network
   - Monte Carlo Tree Search
   - Reinforcement Learning
4. Experiment

Alphago

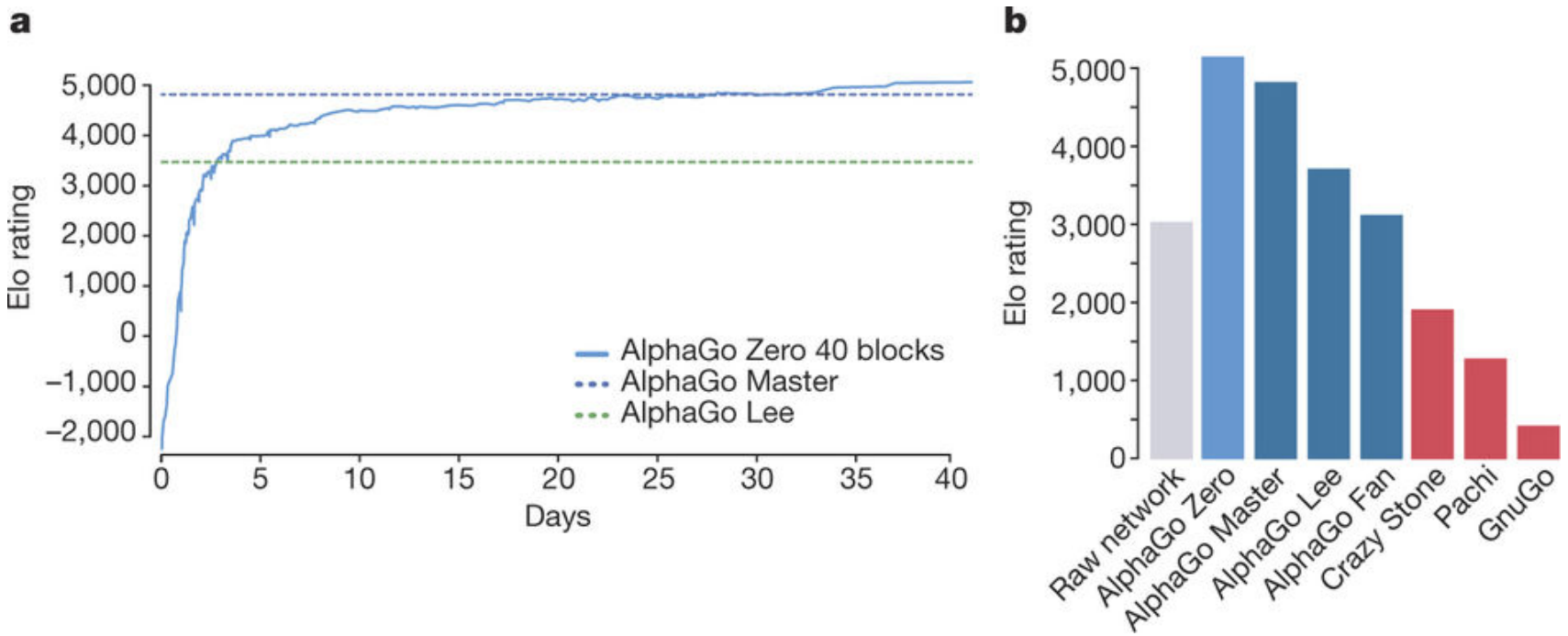
- AlphaGo 4:1 World Champion Lee Sedol 9-Dan
- March 9 - March 15, 2016

Alphago Master

- Won 60 pros online
- 3-0 World Champion Ke Jie May 23, 2017 - May 27, 2017
- Ke Jie: “ Alphago is like the god of go” “Future belongs to AI”

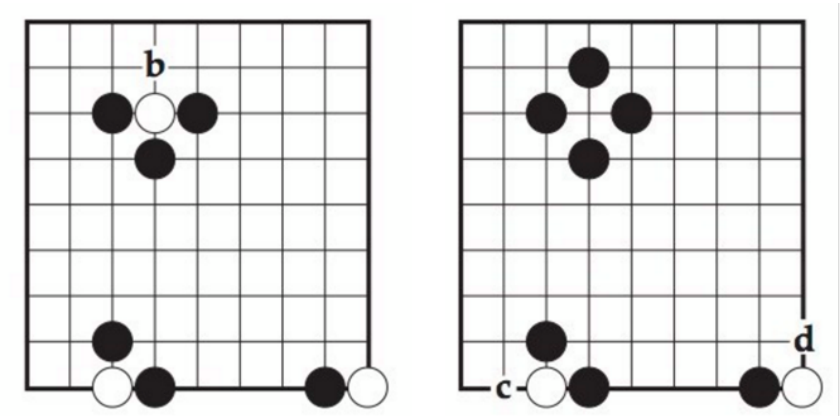
Alphago Zero (This paper)

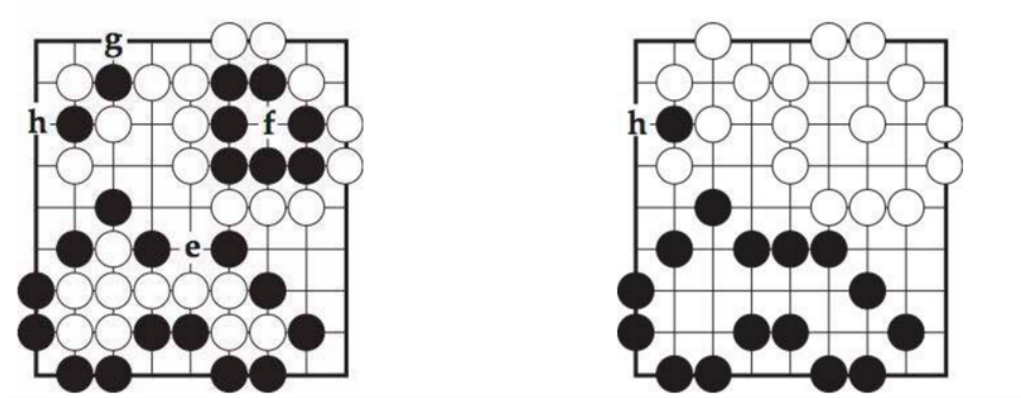
- The second Alphago paper
- Mastering the game of Go without human knowledge
- 100 - 0 Alphago Lee



Go

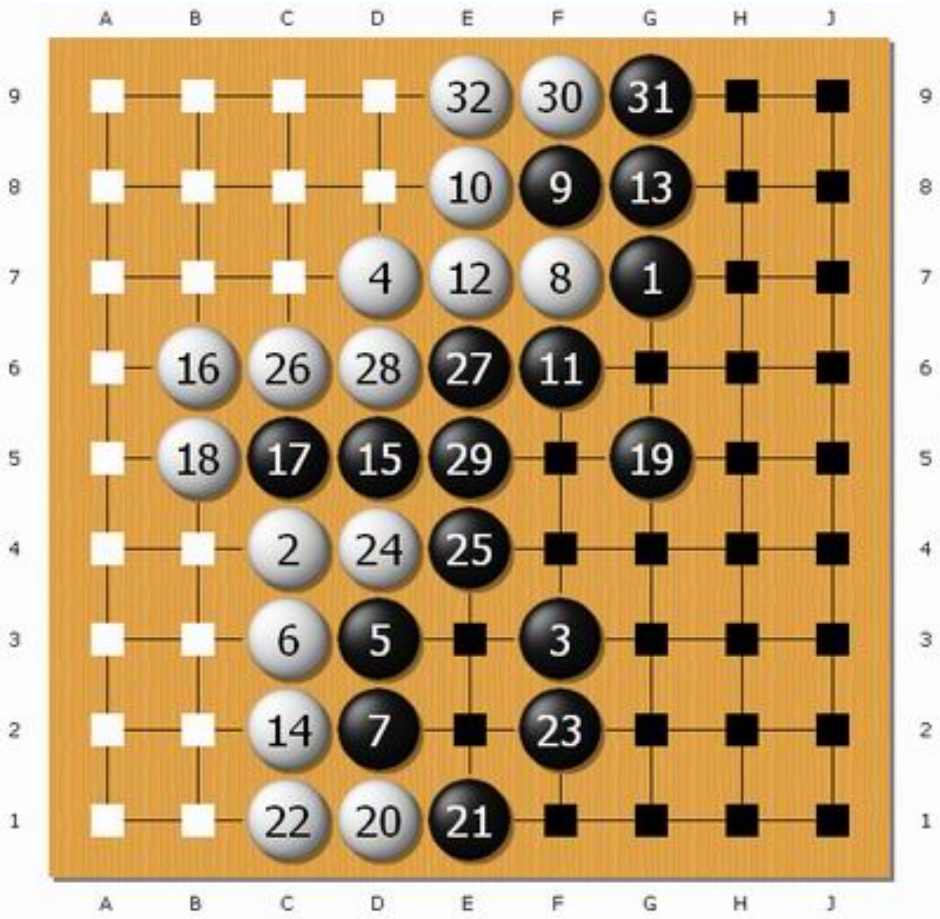
- Start with 19x19 empty board
- One player take black stones and the other take white stones
- Two players take turns to put stones on the board
- Rules:
  - If one connected part is completely surrounded by the opponents stones, remove it from the board.
  - Ko rule: Forbids a board play to repeat a board position





Go

- End when theres no valuable moves on the board.
- Count the territory of both players.
- Add 7.5 points to whites points (called **Komi**).



Method Overview - Play

- How do AlphaGo play go?
- The answer: CNN + MCTS
- Deep Convolutional Neural Network: $(p, v) = f_\theta(s)$. p is the probability of selecting a move(including pass), and v is the winning probability.
- Previous version: Two networks, one for p and one for v. This version: One network for both.
- Monte Carlo Tree Search: Further improve the move based on the neural network.

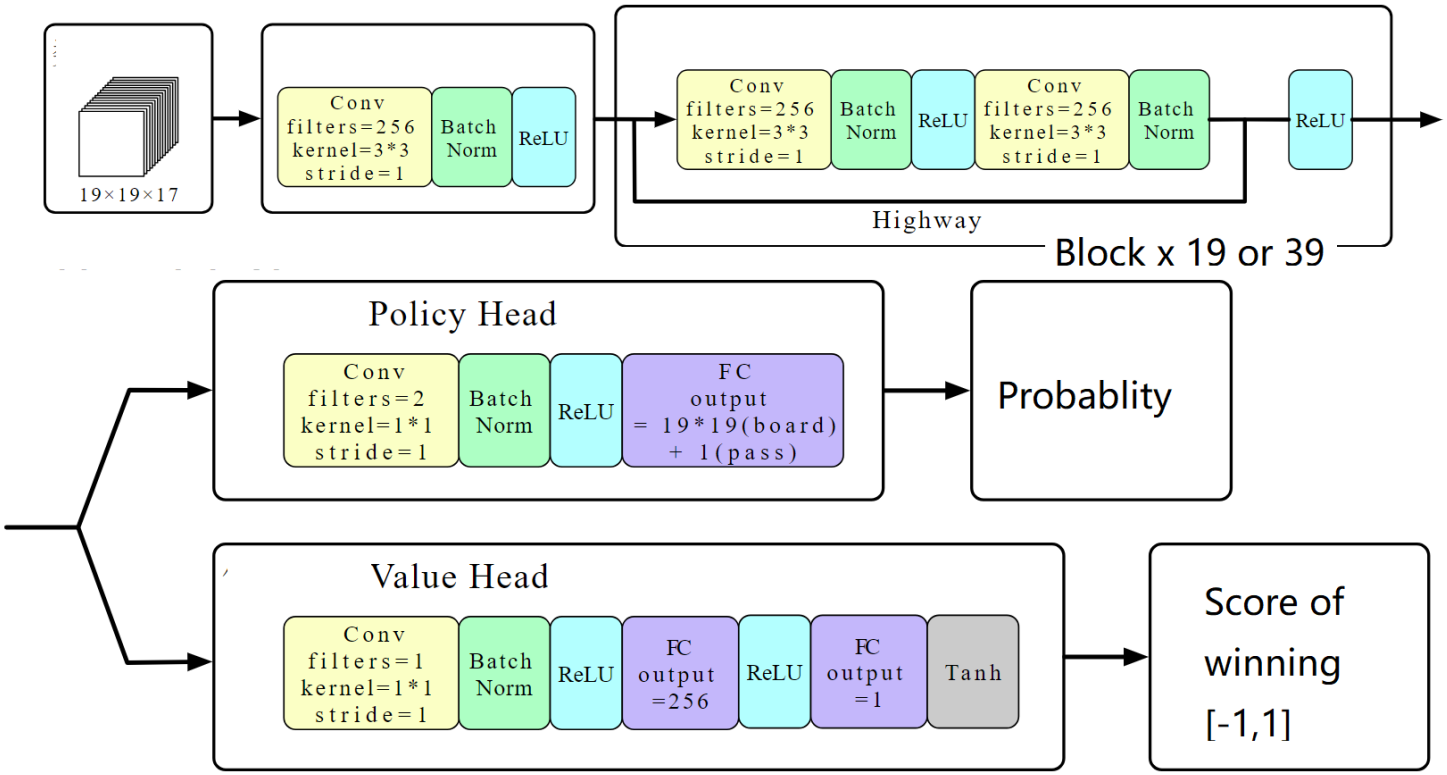
Method Overview - Train

- Training of the network: Reinforcement Learning
- At iteration i:
  - Generate a bunch games.
  - At time-step t of a game, an MCTS search $\pi_t = \alpha_{\theta_{i-1}}(s_t)$ is executed using the parameter of $f_{\theta_{i-1}}$, and a move is played by sampling.
  - The game is played until the end time T, record a final reward $r_T = \{-1, +1\}$.
  - Experience replay: store (s_t, π_t, z_t) in the library, while $z_t = \pm r_T$
  - Sample datas from the experience library, and train the network θ. Loss:

$$l = (z - v)^2 - \pi^T \log p + c||\theta||^2$$

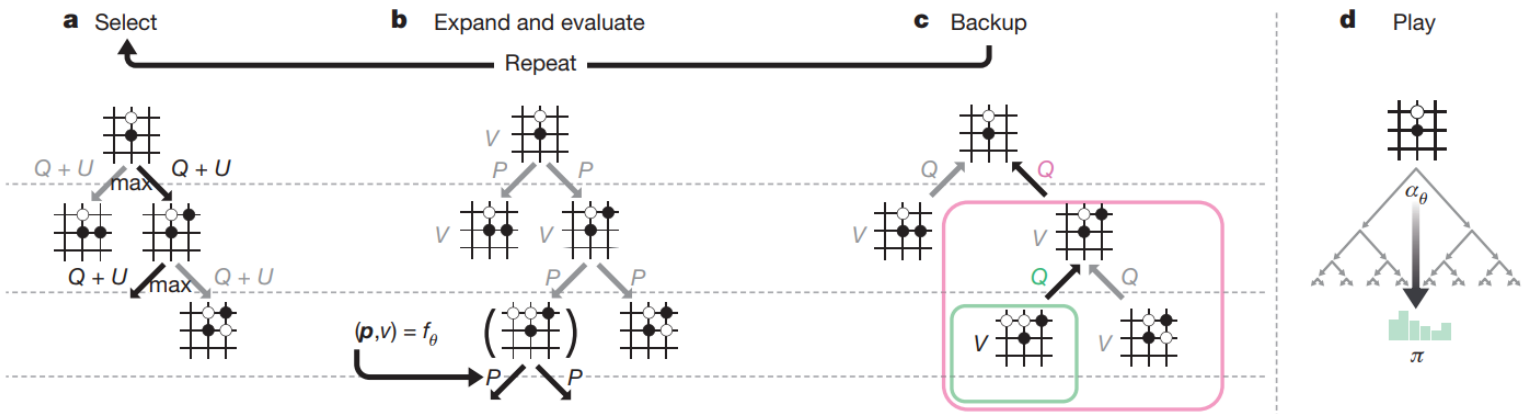
Network structure

- Input: 19x19x17. 17 binary channels instead of previous version 48 channels.
- 16 channels for the condition of board on last 16 moves (1 for having a stone of the color, and 0 for not). 1 channel for current color playing.
- Use ResNet. Either 20 blocks and 40 blocks.



Monte Carlo Tree Search

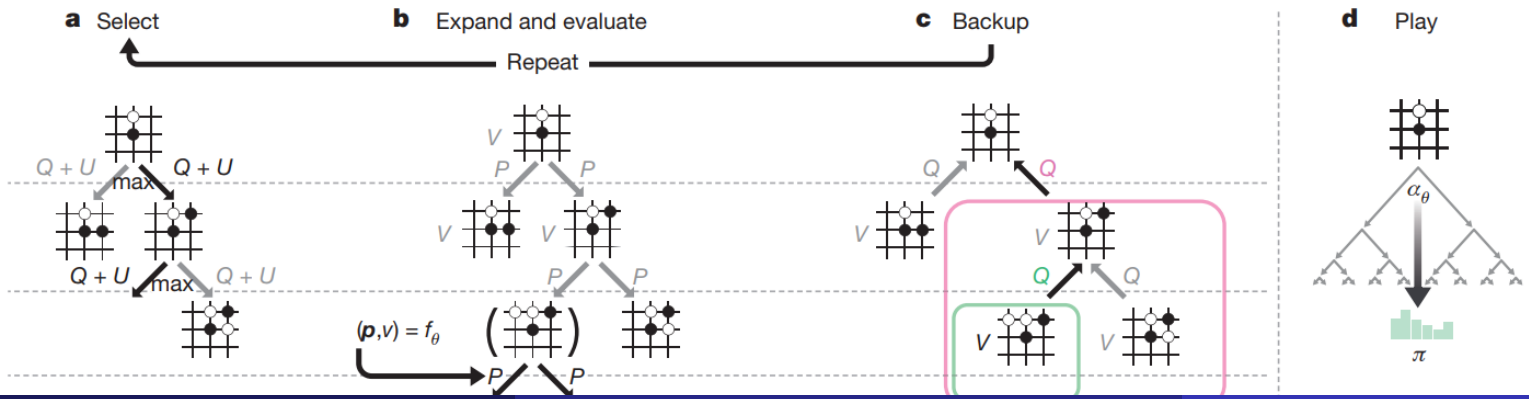
- Each edge (s, a) in the search tree store a prior probability $p(s, a)$, a visit count $N(s, a)$ and action value $Q(s, a)$.
- Each step, find a leaf node by maximizing an upper confidence bound.
- Expand the leaf node using only the network information.
- Update N and Q for every edge on the path.



Monte Carlo Tree Search

3+1 steps:

- Select: At every node s_t, choose $a_t = \arg\max_a(Q(s_t, a) + U(s_t, a))$.
  Where $U(s, a) = c_{puct} P(s, a) \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}$
- Expand and evaluate: Select a random i in [1..8], make a rotation or reflection of the board, evaluate the current board using the network. The leaf node s_L is expanded, and each edge (s_L, a) is initialized as $[N(s_L, a) = 0, Q(s_L, a) = 0, W(s_L, a) = 0, P(s_L, a) = p_a]$
- Backup: Every edge on the path $N(s, a) = N(s, a) + 1$, $W(s, a) = W(s, a) + v$, $Q(s, a) = \frac{W(s,a)}{N(s,a)}$
- Play: In training, $\pi(a|s_0) = \frac{N(s_0,a)^{1/\tau}}{\sum_b N(s_0,b)^{1/\tau}}$. $\tau \to 0$ in real play.



Self play

- In each iteration, play 25000 games.
- Each move, use 1600 MCTS simulations (around 0.4s).
- For the first 30 moves, $\tau = 1$ to encourage exploration. For the reminder of the game $\tau \to 0$.
- Add a Dirichlet noise to the prior to further encourage exploration: $P(s, a) = (1 - \epsilon)p_a + \epsilon\eta_a$, where $\eta \sim \text{Dir}(0.03)$ and $\epsilon = 0.25$.
- To save calculation, define a resign rate. Automatically ensure the false positives are below 5% (run a subset with 10% data with no resign to decide the threshold).

Checkpoint

- On each checkpoint, evaluate the current network
- Compare the current network with previous network, using the winning rate of 400 games.
- Update the network only if the winning rate is larger than 55%

Summary

- Play by RNN and MCTS
- Train the RNN by MCTS based RL
- Major difference to previous versions:
  - No human knowledge. Previous version contains a policy network learned from human plays.
  - Combine 2 networks into a single one.
  - Greatly reduce the number of feature channels.

Go learned

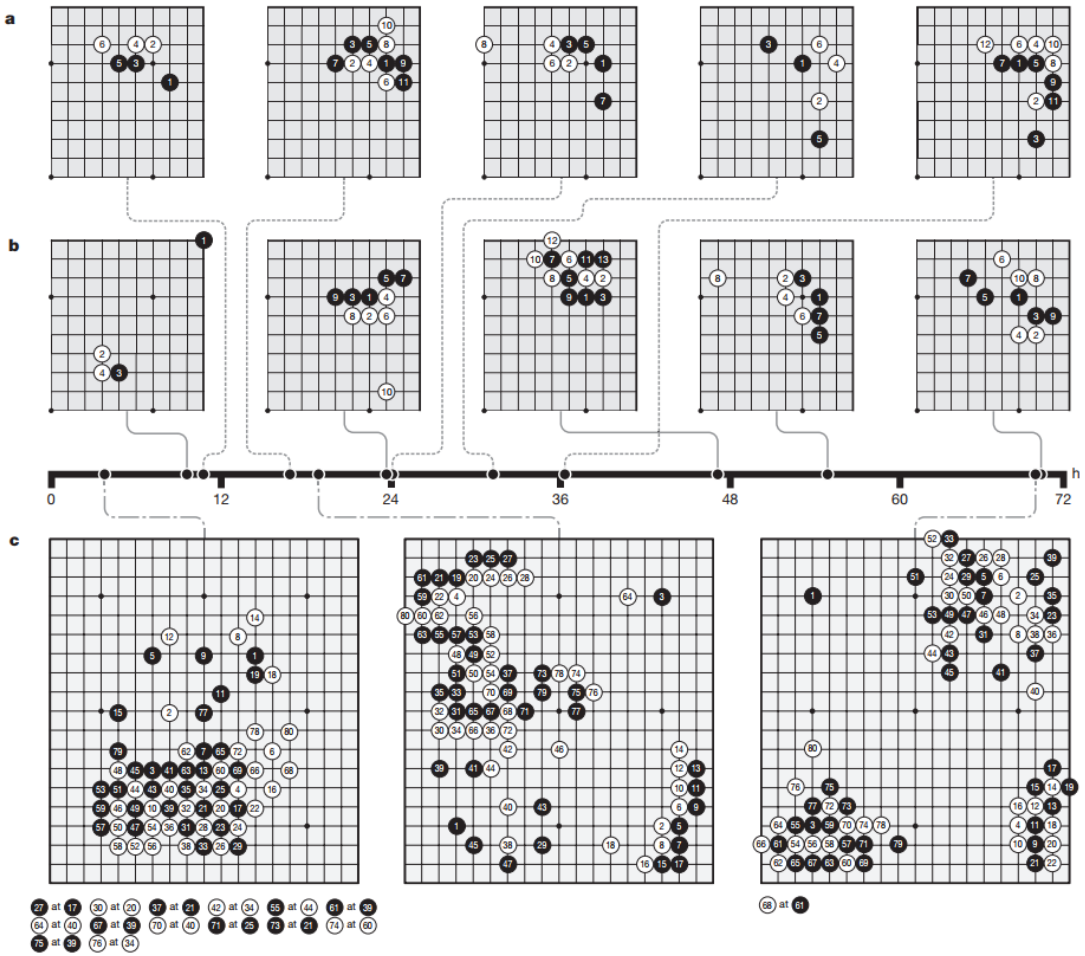


Figure 5 | Go knowledge learned by AlphaGo Zero. a, Five human *joseki* (common corner sequences) discovered during AlphaGo Zero training. The associated timestamps indicate the first time each sequence occurred (taking account of rotation and reflection) during self-play training. Extended Data Figure 2 provides the frequency of occurence over training for each sequence. **b**, Five *joseki* favoured at different stages of self-play training. Each displayed corner sequence was played with the greatest frequency, among all corner sequences, during an iteration of self-play training. The timestamp of that iteration is indicated on the timeline. At 10 h a weak corner move was preferred. At 47 h the 3–3 invasion was most frequently played. This *joseki* is also common in human professional play; however AlphaGo Zero later discovered and preferred a new variation. Extended Data Figure 3 provides the frequency of occurence over time for all five sequences and the new variation. **c**, The first 80 moves of three self-play games that were played at different stages of training, using 1,600 simulations (around 0.4 s) per search. At 3 h, the game focuses greedily on capturing stones, much like a human beginner. At 19 h, the game exhibits the fundamentals of life-and-death, influence and territory. At 70 h, the game is remarkably balanced, involving multiple battles and a complicated *ko* fight, eventually resolving into a half-point win for white. See Supplementary Information for the full games.

Different structures

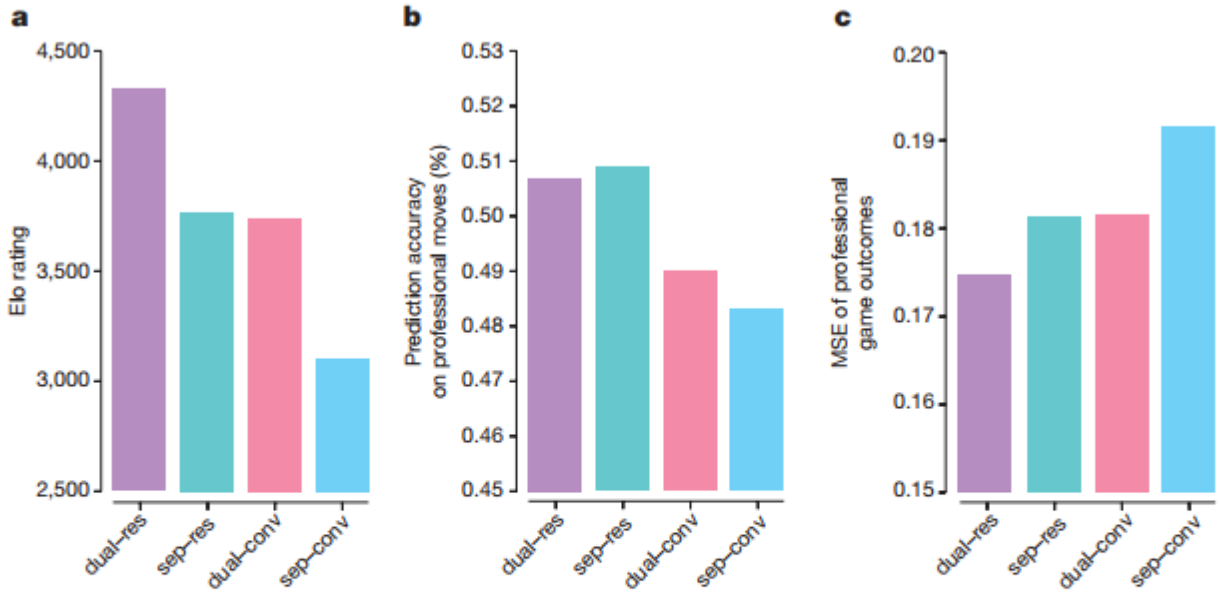
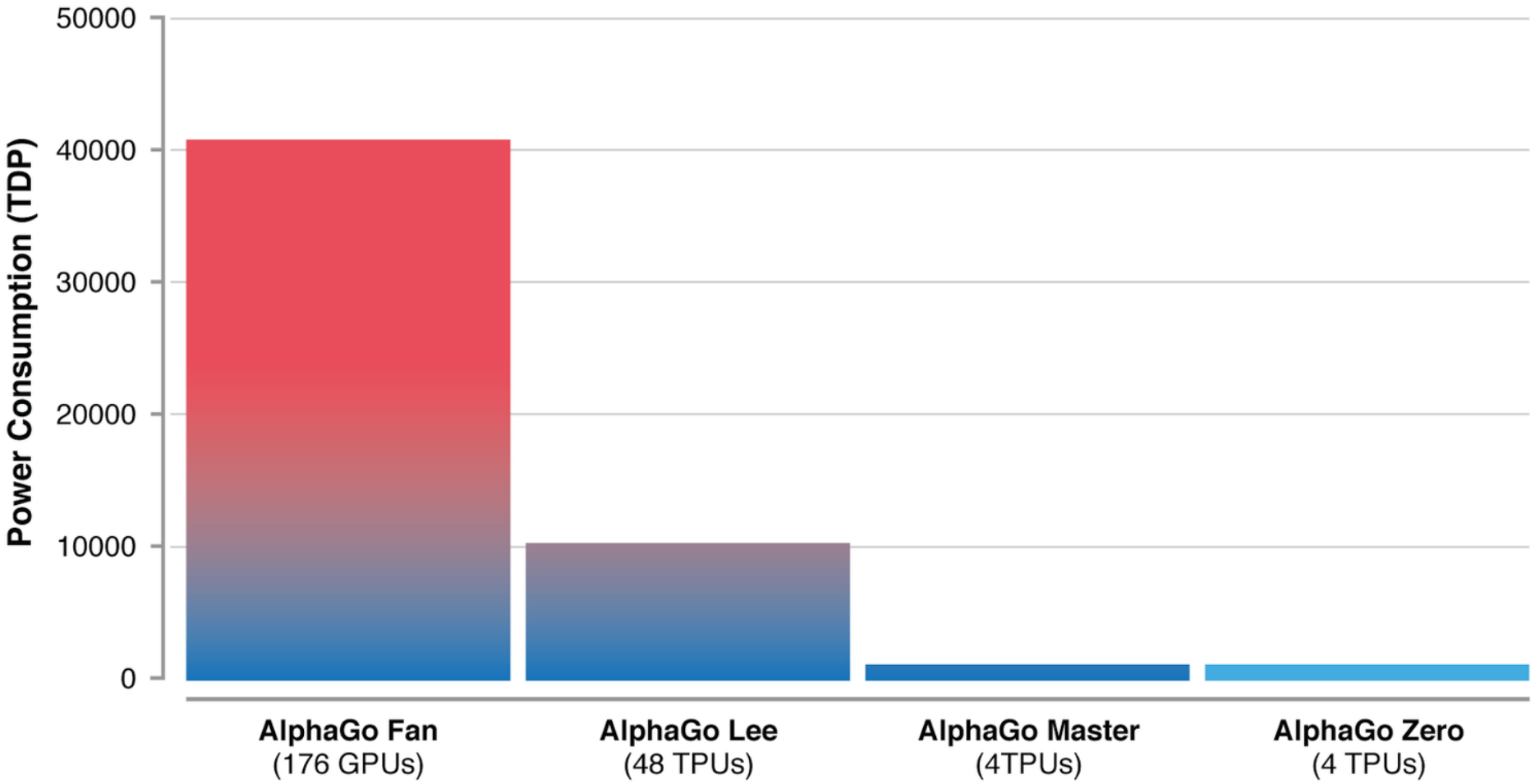


Figure 4 | Comparison of neural network architectures in AlphaGo Zero and AlphaGo Lee. Comparison of neural network architectures using either separate (sep) or combined policy and value (dual) networks, and using either convolutional (conv) or residual (res) networks. The combinations 'dual–res' and 'sep–conv' correspond to the neural network architectures used in AlphaGo Zero and AlphaGo Lee, respectively. Each network was trained on a fixed dataset generated by a previous run of AlphaGo Zero. **a**, Each trained network was combined with AlphaGo Zero's search to obtain a different player. Elo ratings were computed from evaluation games between these different players, using 5 s of thinking time per move. **b**, Prediction accuracy on human professional moves (from the GoKifu dataset) for each network architecture. **c** MSE of human professional game outcomes (from the GoKifu dataset) for each network architecture.

Cost


Power Consumption (TDP)
50000
40000
30000
20000
10000
0
AlphaGo Fan
(176 GPUs)
AlphaGo Lee
(48 TPUs)
AlphaGo Master
(4TPUs)
AlphaGo Zero
(4 TPUs)

Supervised Training

Extended Data Table 1 | Move prediction accuracy

	KGS train	*KGS* test	*GoKifu* validation
Supervised learning (20 block)	62.0	60.4	54.3
Supervised learning (12 layer [12])	59.1	55.9	-
Reinforcement learning (20 block)	-	-	49.0
Reinforcement learning (40 block)	-	-	51.3

Percentage accuracies of move prediction for neural networks trained by reinforcement learning (that is, AlphaGo Zero) or supervised learning. For supervised learning, the network was trained for 3 days on KGS data (amateur games); comparative results are also shown from ref. 12. For reinforcement learning, the 20-block network was trained for 3 days and the 40-block network was trained for 40 days. Networks were also evaluated on a validation set based on professional games from the GoKifu dataset.

Extended Data Table 2 | Game outcome prediction error

	KGS train	*KGS* test	*GoKifu* validation
Supervised learning (20 block)	0.177	0.185	0.207
Supervised learning (12 layer [12])	0.19	0.37	-
Reinforcement learning (20 block)	-	-	0.177
Reinforcement learning (40 block)	-	-	0.180

Mean squared error on game outcome predictions for neural networks trained by reinforcement learning (that is, AlphaGo Zero) or supervised learning. For supervised learning, the network was trained for 3 days on KGS data (amateur games); comparative results are also shown from ref. 12. For reinforcement learning, the 20 block network was trained for 3 days and the 40 block network was trained for 40 days. Networks were also evaluated on a validation set based on professional games from the GoKifu dataset.