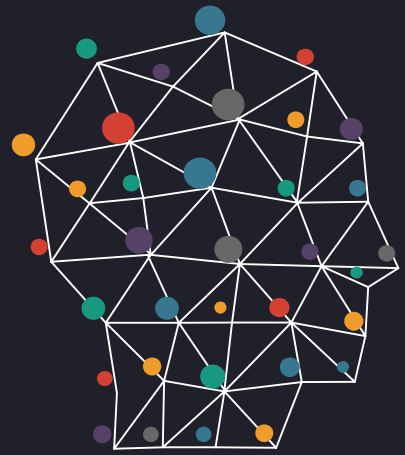
Mastering the game of Go without human knowledge

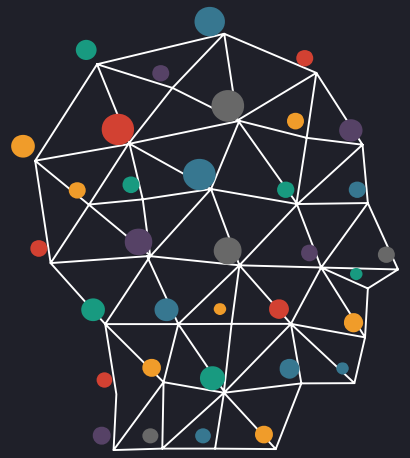
Wei Wang 2019.12.10

East China Normal University
Multi-Agent Artificial Intelligence Laboratory

Silver D, Schrittwieser J, et al. Mastering the game of go without human knowledge[J]. Nature, 2017.

CONTENTS





1. Introduction

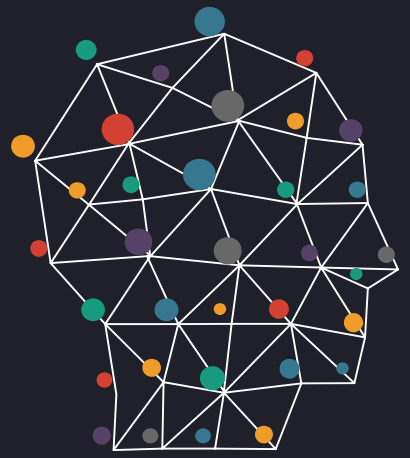
1.1 AlphaGo Fan & AlphaGo Zero

AlphaGo Fan：

AlphaGo was the first program to achieve superhuman performance in Go.
Used two deep neural networks:
- A policy network that outputs move probabilities.
- A value network that outputs a position evaluation.

AlphaGo Zero：

- It is trained by self-play reinforcement learning, starting from random play, without any supervision or use of human data.
- It uses only the black and white stones from the board as input features.
- It uses a single neural network, rather than separate policy and value networks.
- It uses a simpler tree search that relies upon this single neural network to evaluate positions and sample moves, without performing any Monte Carlo rollouts.



2. Method

2.1 Problem Description

Go expression：

- 19×19=361positions; white:1, black:-1, None:0
- Checkerboard state vector:

$$\vec{s} = (\underbrace{1, 0, -1, \ldots}_{361})$$

- Next action vector:

$$\vec{s} = (\underbrace{1, 0, -1, \ldots}_{361})$$

Go problem： Given a state arbitrarily, looking for the best policy, finally get the most site on the checkerboard.

2.2 Network Structure

Deep neural network f(θ):

- Parameters were used: θ

- Input: raw board representation s of the position and its history(Seven steps)

- Output: move probabilities and a value (p, v) = fθ(s)
  - The vector of move probabilities p represents the probability of selecting each move a (including pass), pa = Pr(a| s).
  - The value v is a scalar evaluation, estimating the probability of the current player winning from position s.

The neural network consists of many residual blocks of convolutional layers with batch normalization and rectifier nonlinearities.

2.3 Reinforcement Learning

Reinforcement Learning：

- Self - play
- In each position s, an MCTS search is executed, guided by the neural network fθ.
- These search probabilities usually select much stronger moves than the raw move probabilities p of the neural network fθ(s).
- Self-play with search—using the improved MCTS based policy to select each move, then using the game winner z as a sample of the value—may be viewed as a powerful policy evaluation operator.
- Use these search operators repeatedly in a policy iteration procedure: the neural network's parameters are updated to make the move probabilities and value more closely match the improved search probabilities and self-play winner (π, z); these new parameters are used in the next iteration of self-play to make the search even stronger.

2.4 Self-play reinforcement learning

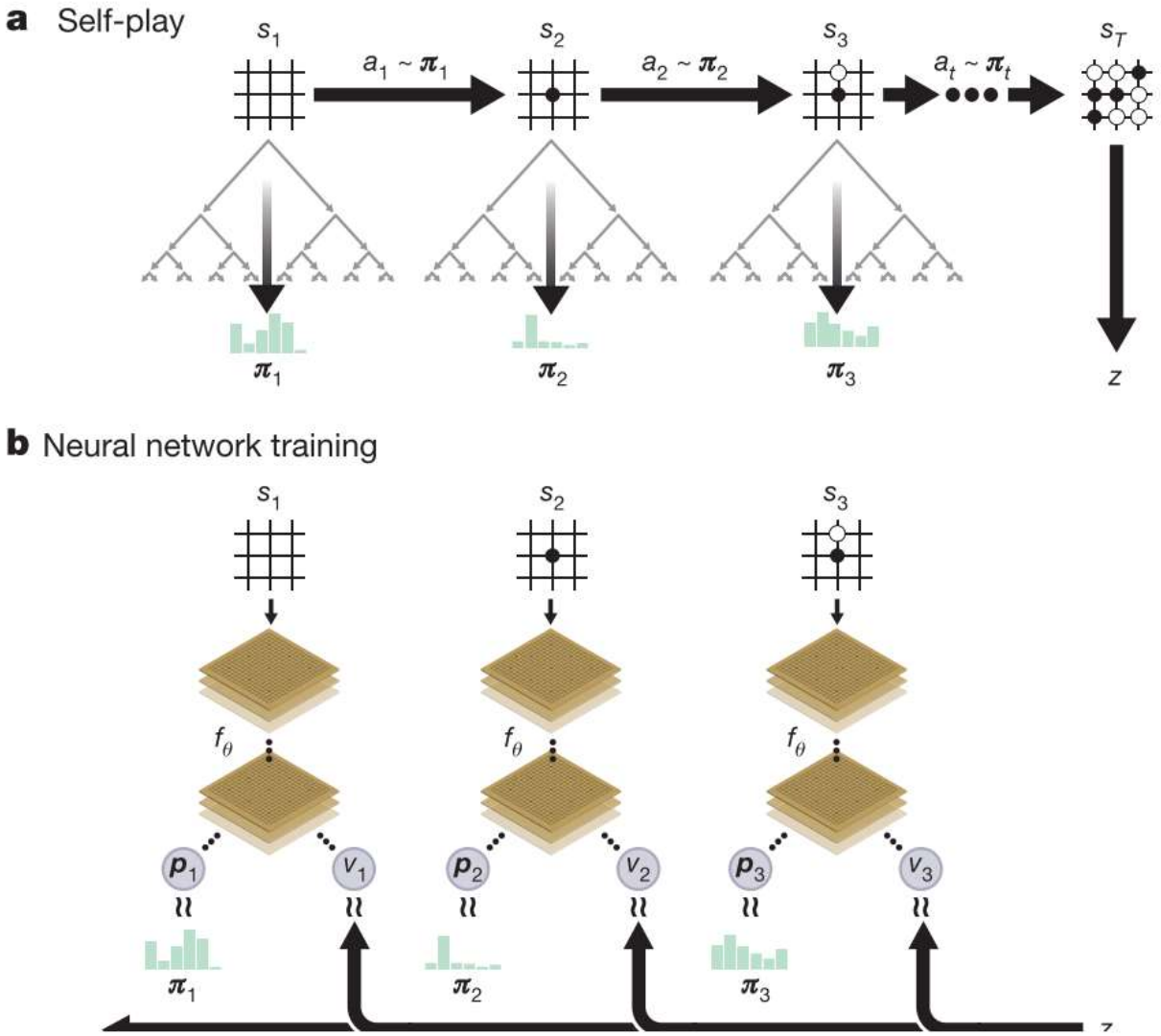


Figure 1 | Self-play reinforcement learning in AlphaGo Zero. a, The program plays a game $s_1, \ldots, s_T$ against itself. In each position s_t, an MCTS α_θ is executed (see Fig. 2) using the latest neural network f_θ. Moves are selected according to the search probabilities computed by the MCTS, $a_t \sim \pi_t$. The terminal position s_T is scored according to the rules of the game to compute the game winner z. **b**, Neural network training in AlphaGo Zero. The neural network takes the raw board position s_t as its input, passes it through many convolutional layers with parameters θ, and outputs both a vector $\boldsymbol{p}_t$, representing a probability distribution over moves, and a scalar value v_t, representing the probability of the current player winning in position s_t. The neural network parameters θ are updated to maximize the similarity of the policy vector $\boldsymbol{p}_t$ to the search probabilities π_t, and to minimize the error between the predicted winner v_t and the game winner z (see equation (1)). The new parameters are used in the next iteration of self-play as in **a**.

$$Q(s,a) = 1/N(s,a) \sum_{s'|s,a \to s'} V(s')$$

2.5 MCTS

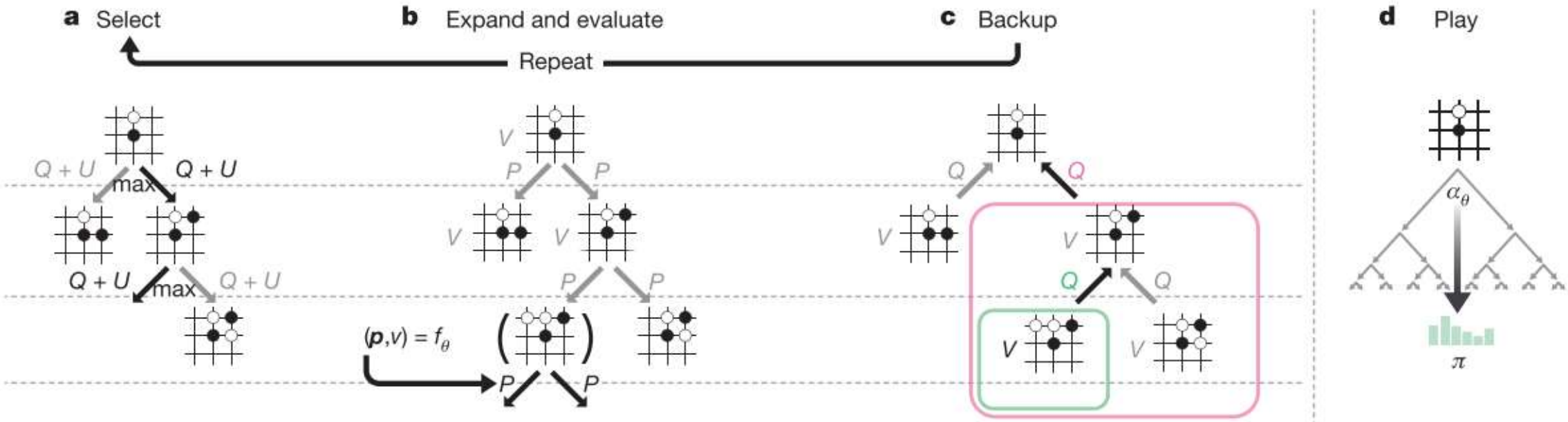
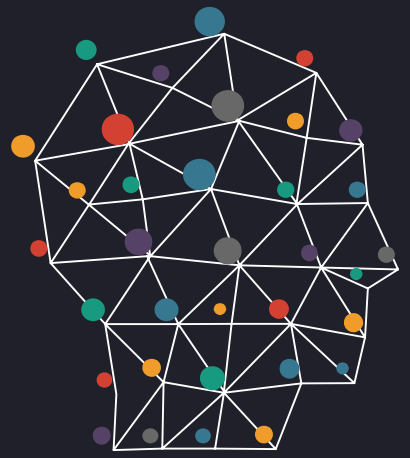


Figure 2 | MCTS in AlphaGo Zero. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus an upper confidence bound U that depends on a stored prior probability P and visit count N for that edge (which is incremented once traversed). **b**, The leaf node is expanded and the associated position s is evaluated by the neural network $(P(s,\cdot),V(s)) = f_\theta(s)$; the vector of P values are stored in the outgoing edges from s. **c**, Action value Q is updated to track the mean of all evaluations V in the subtree below that action. **d**, Once the search is complete, search probabilities π are returned, proportional to $N^{1/\tau}$, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.

$$(\boldsymbol{p}, v) = f_\theta(s) \text{ and } l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$



3. Experiment

3.1 Empirical evaluation of AlphaGo Zero

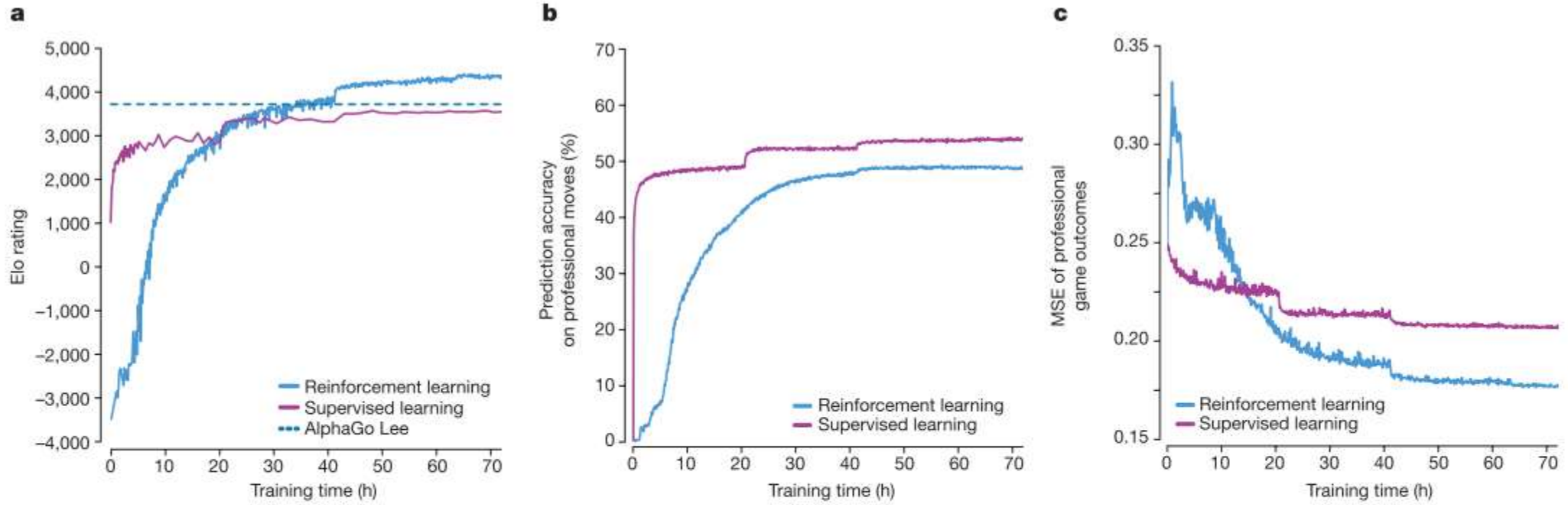


Figure 3 | Empirical evaluation of AlphaGo Zero. a, Performance of self-play reinforcement learning. The plot shows the performance of each MCTS player α_{θ_i} from each iteration i of reinforcement learning in AlphaGo Zero. Elo ratings were computed from evaluation games between different players, using 0.4 s of thinking time per move (see Methods). For comparison, a similar player trained by supervised learning from human data, using the KGS dataset, is also shown. **b**, Prediction accuracy on human professional moves. The plot shows the accuracy of the neural network f_{θ_i}, at each iteration of self-play i, in predicting human professional moves from the GoKifu dataset. The accuracy measures the percentage of positions in which the neural network assigns the highest probability to the human move. The accuracy of a neural network trained by supervised learning is also shown. **c**, Mean-squared error (MSE) of human professional game outcomes. The plot shows the MSE of the neural network f_{θ_i}, at each iteration of self-play i, in predicting the outcome of human professional games from the GoKifu dataset. The MSE is between the actual outcome $z \in \{-1, +1\}$ and the neural network value v, scaled by a factor of $\frac{1}{4}$ to the range of 0–1. The MSE of a neural network trained by supervised learning is also shown.

3.3 Knowledge learned by AlphaGo Zero

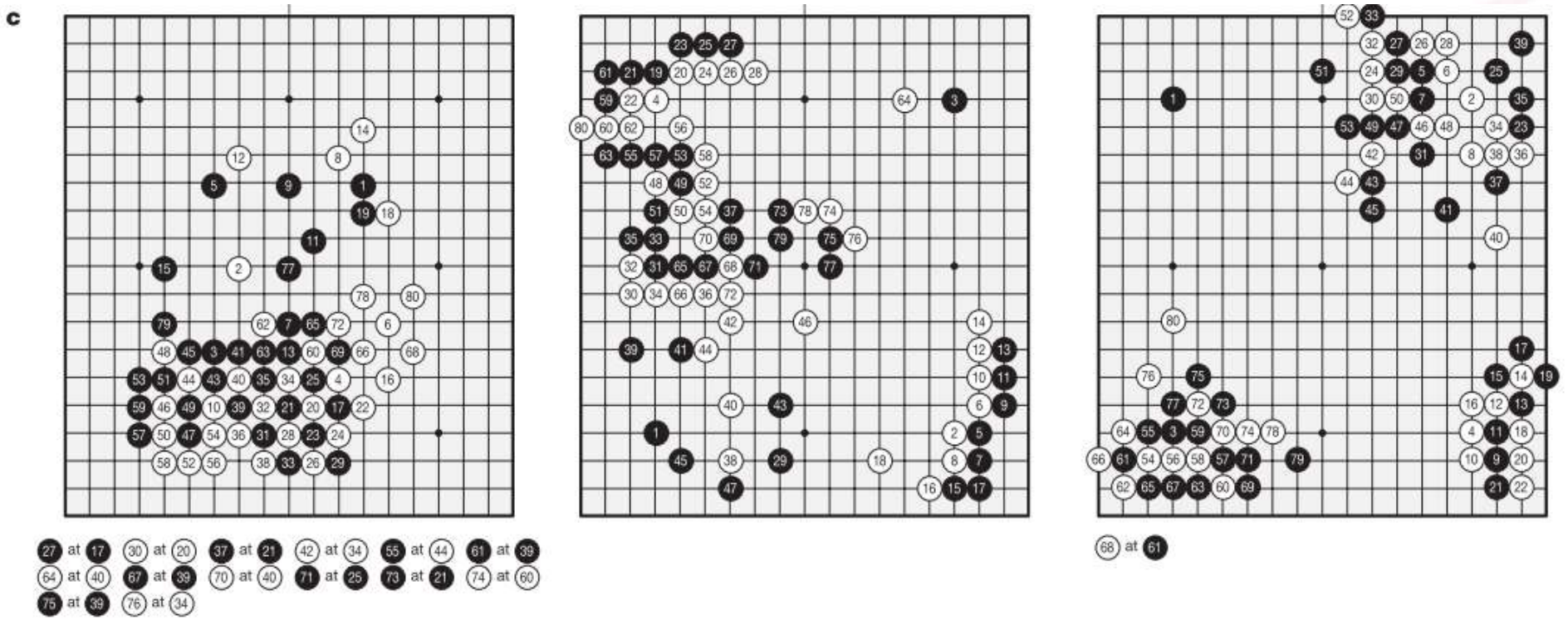


Figure 5 | Go knowledge learned by AlphaGo Zero. a, Five human *joseki* (common corner sequences) discovered during AlphaGo Zero training. The associated timestamps indicate the first time each sequence occurred (taking account of rotation and reflection) during self-play training. Extended Data Figure 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.

3.4 Final Performance of AlphaGo Zero

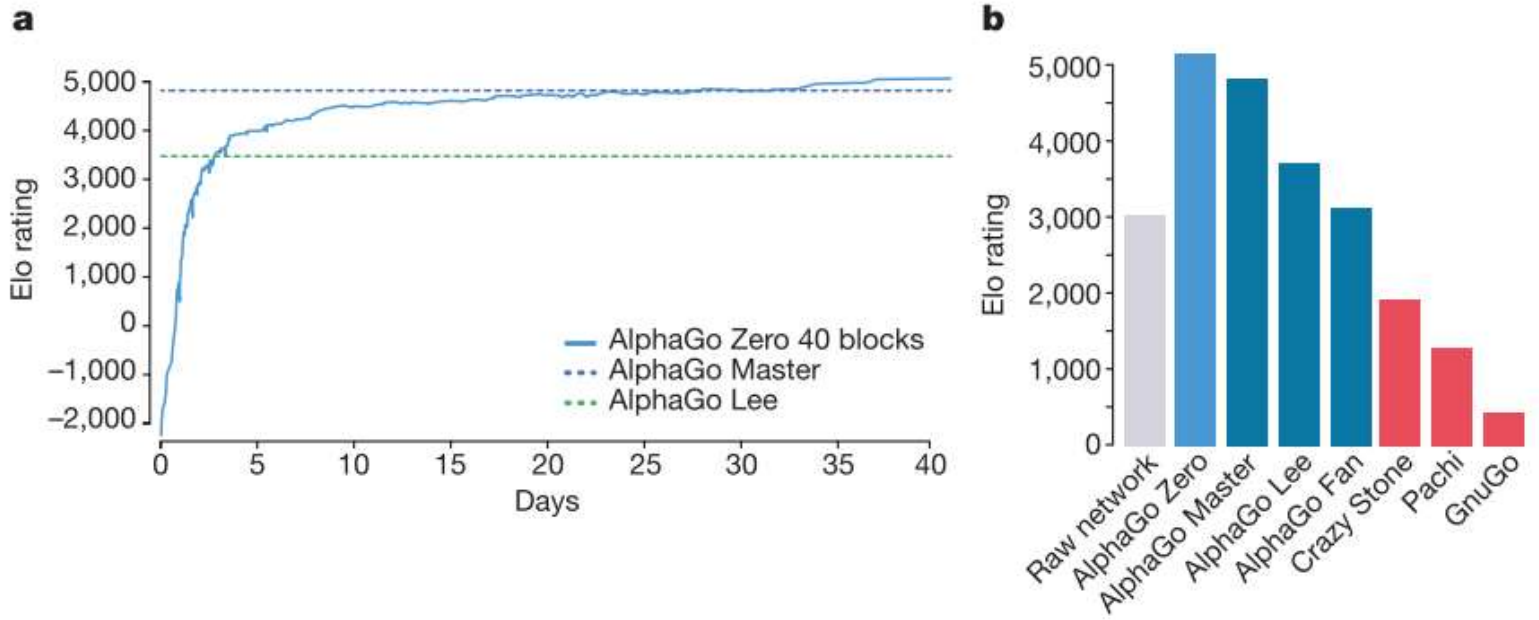
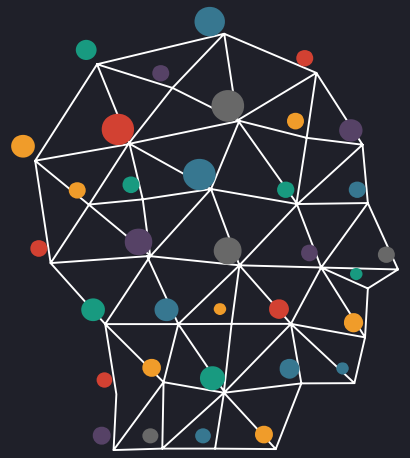


Figure 6 | Performance of AlphaGo Zero. a, Learning curve for AlphaGo Zero using a larger 40-block residual network over 40 days. The plot shows the performance of each player α_{θ_i} from each iteration i of our reinforcement learning algorithm. Elo ratings were computed from evaluation games between different players, using 0.4 s per search (see Methods). **b**, Final performance of AlphaGo Zero. AlphaGo Zero was trained for 40 days using a 40-block residual neural network. The plot shows the results of a tournament between: AlphaGo Zero, AlphaGo Master (defeated top human professionals 60–0 in online games), AlphaGo Lee (defeated Lee Sedol), AlphaGo Fan (defeated Fan Hui), as well as previous Go programs Crazy Stone, Pachi and GnuGo. Each program was given 5 s of thinking time per move. AlphaGo Zero and AlphaGo Master played on a single machine on the Google Cloud; AlphaGo Fan and AlphaGo Lee were distributed over many machines. The raw neural network from AlphaGo Zero is also included, which directly selects the move a with maximum probability p_a, without using MCTS. Programs were evaluated on an Elo scale[25]: a 200-point gap corresponds to a 75% probability of winning.



4. Conclusion

4 Conclusion

Our results comprehensively demonstrate that a pure reinforcement learning approach is fully feasible, even in the most challenging of domains: it is possible to train to superhuman level, without human examples or guidance, given no knowledge of the domain beyond basic rules. Furthermore, a pure reinforcement learning approach requires just a few more hours to train, and achieves much better asymptotic performance, compared to training on human expert data. Using this approach, AlphaGo Zero defeated the strongest previous versions of AlphaGo, which were trained from human data using handcrafted features, by a large margin.

Humankind has accumulated Go knowledge from millions of games played over thousands of years, collectively distilled into patterns, proverbs and books. In the space of a few days, starting tabula rasa, AlphaGo Zero was able to rediscover much of this Go knowledge, as well as novel strategies that provide new insights into the oldest of games.

Mastering the game of Go without human knowledge

Thanks!