COMPUTER SCIENCE

A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

David Silver[1,2]*†, Thomas Hubert[1]*, Julian Schrittwieser[1]*, Ioannis Antonoglou[1], Matthew Lai[1], Arthur Guez[1], Marc Lanctot[1], Laurent Sifre[1], Dharshan Kumaran[1], Thore Graepel[1], Timothy Lillicrap[1], Karen Simonyan[1], Demis Hassabis[1]†

The game of chess is the longest-studied domain in the history of artificial intelligence. The strongest programs are based on a combination of sophisticated search techniques, domain-specific adaptations, and handcrafted evaluation functions that have been refined by human experts over several decades. By contrast, the AlphaGo Zero program recently achieved superhuman performance in the game of Go by reinforcement learning from self-play. In this paper, we generalize this approach into a single AlphaZero algorithm that can achieve superhuman performance in many challenging games. Starting from random play and given no domain knowledge except the game rules, AlphaZero convincingly defeated a world champion program in the games of chess and shogi (Japanese chess), as well as Go.

The study of computer chess is as old as computer science itself. Charles Babbage, Alan Turing, Claude Shannon, and John von Neumann devised hardware, algorithms, and theory to analyze and play the game of chess. Chess subsequently became a grand challenge task for a generation of artificial intelligence researchers, culminating in high-performance computer chess programs that play at a superhuman level (*1*, *2*). However, these systems are highly tuned to their domain and cannot be generalized to other games without substantial human effort, whereas general game-playing systems (*3*, *4*) remain comparatively weak.

A long-standing ambition of artificial intelligence has been to create programs that can instead learn for themselves from first principles (*5*, *6*). Recently, the AlphaGo Zero algorithm achieved superhuman performance in the game of Go by representing Go knowledge with the use of deep convolutional neural networks (*7*, *8*), trained solely by reinforcement learning from games of self-play (*9*). In this paper, we introduce AlphaZero, a more generic version of the AlphaGo Zero algorithm that accommodates, without special casing, a broader class of game rules. We apply AlphaZero to the games of chess and shogi, as well as Go, by using the same algorithm and network architecture for all three games. Our results demonstrate that a general-purpose reinforcement learning algorithm can learn, tabula rasa—without domain-specific human knowledge or data, as evidenced by the same algorithm succeeding in multiple domains—superhuman performance across multiple challenging games.

A landmark for artificial intelligence was achieved in 1997 when Deep Blue defeated the human world chess champion (*1*). Computer chess programs continued to progress steadily beyond human level in the following two decades. These programs evaluate positions by using handcrafted features and carefully tuned weights, constructed by strong human players and programmers, combined with a high-performance alpha-beta search that expands a vast search tree by using a large number of clever heuristics and domain-specific adaptations. In (*10*) we describe these augmentations, focusing on the 2016 Top Chess Engine Championship (TCEC) season 9 world champion Stockfish (*11*); other strong chess programs, including Deep Blue, use very similar architectures (*1*, *12*).

In terms of game tree complexity, shogi is a substantially harder game than chess (*13*, *14*): It is played on a larger board with a wider variety of pieces; any captured opponent piece switches sides and may subsequently be dropped anywhere on the board. The strongest shogi programs, such as the 2017 Computer Shogi Association (CSA) world champion Elmo, have only recently defeated human champions (*15*). These programs use an algorithm similar to those used by computer chess programs, again based on a highly optimized alpha-beta search engine with many domain-specific adaptations.

AlphaZero replaces the handcrafted knowledge and domain-specific augmentations used in traditional game-playing programs with deep neural networks, a general-purpose reinforcement learning algorithm, and a general-purpose tree search algorithm.

Instead of a handcrafted evaluation function and move-ordering heuristics, AlphaZero uses a deep neural network $(\mathbf{p}, v) = f_\theta(s)$ with parameters θ. This neural network $f_\theta(s)$ takes the board position s as an input and outputs a vector of move probabilities $\mathbf{p}$ with components $p_a = \Pr(a|s)$ for each action a and a scalar value v estimating the expected outcome z of the game from position s, $v \approx \mathbb{E}[z|s]$. AlphaZero learns these move probabilities and value estimates entirely from self-play; these are then used to guide its search in future games.

Instead of an alpha-beta search with domain-specific enhancements, AlphaZero uses a general-purpose Monte Carlo tree search (MCTS) algorithm. Each search consists of a series of simulated games of self-play that traverse a tree from root state s_{root} until a leaf state is reached. Each simulation proceeds by selecting in each state s a move a with low visit count (not previously frequently explored), high move probability, and high value (averaged over the leaf states of

[1]DeepMind, 6 Pancras Square, London N1C 4AG, UK. [2]University College London, Gower Street, London WC1E 6BT, UK.
*These authors contributed equally to this work.
†Corresponding author. Email: davidsilver@google.com (D.S.); dhcontact@google.com (D.H.)

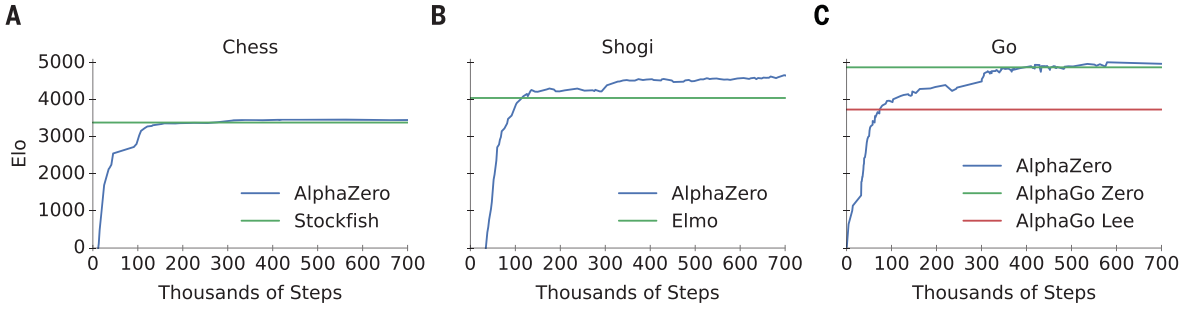

Fig. 1. Training AlphaZero for 700,000 steps. Elo ratings were computed from games between different players where each player was given 1 s per move. (**A**) Performance of AlphaZero in chess compared with the 2016 TCEC world champion program Stockfish. (**B**) Performance of AlphaZero in shogi compared with the 2017 CSA world champion program Elmo. (**C**) Performance of AlphaZero in Go compared with AlphaGo Lee and AlphaGo Zero (20 blocks over 3 days).

simulations that selected a from s) according to the current neural network f_θ. The search returns a vector π representing a probability distribution over moves, $\pi_a = \Pr(a|s_{root})$.

The parameters θ of the deep neural network in AlphaZero are trained by reinforcement learning from self-play games, starting from randomly initialized parameters θ. Each game is played by running an MCTS from the current position $s_{root} = s_t$ at turn t and then selecting a move, $a_t \sim \pi_t$, either proportionally (for exploration) or greedily (for exploitation) with respect to the visit counts at the root state. At the end of the game, the terminal position s_T is scored according to the rules of the game to compute the game outcome z: –1 for a loss, 0 for a draw, and +1 for a win. The neural network parameters θ are updated to minimize the error between the predicted outcome v_t and the game outcome z and to maximize the similarity of the policy vector $\mathbf{p}_t$ to the search probabilities π_t. Specifically, the parameters θ are adjusted by gradient descent on a loss function l that sums over mean-squared error and cross-entropy losses

$$(\mathbf{p}, v) = f_\theta(s), l = (z - v)^2 - \pi^{\mathrm{T}} \log \mathbf{p} + c\|\theta\|^2, (1)$$

where c is a parameter controlling the level of L_2 weight regularization. The updated parameters are used in subsequent games of self-play.

The AlphaZero algorithm described in this paper [see (*10*) for the pseudocode] differs from the original AlphaGo Zero algorithm in several respects.

AlphaGo Zero estimated and optimized the probability of winning, exploiting the fact that Go games have a binary win or loss outcome. However, both chess and shogi may end in drawn outcomes; it is believed that the optimal solution to chess is a draw (*16*–*18*). AlphaZero instead estimates and optimizes the expected outcome.

The rules of Go are invariant to rotation and reflection. This fact was exploited in AlphaGo and AlphaGo Zero in two ways. First, training data were augmented by generating eight symmetries for each position. Second, during MCTS, board positions were transformed by using a randomly selected rotation or reflection before being evaluated by the neural network, so that the Monte Carlo evaluation was averaged over different biases. To accommodate a broader class of games, AlphaZero does not assume symmetry; the rules of chess and shogi are asymmetric (e.g., pawns only move forward, and castling is different on kingside and queenside). AlphaZero does not augment the training data and does not transform the board position during MCTS.

In AlphaGo Zero, self-play games were generated by the best player from all previous iterations. After each iteration of training, the performance of the new player was measured against the best player; if the new player won by a margin of 55%, then it replaced the best player. By contrast, AlphaZero simply maintains a single neural network that is updated continually rather than waiting for an iteration to complete. Self-play games are always generated by using the latest parameters for this neural network.

As in AlphaGo Zero, the board state is encoded by spatial planes based only on the basic rules for each game. The actions are encoded by either spatial planes or a flat vector, again based only on the basic rules for each game (*10*).

AlphaGo Zero used a convolutional neural network architecture that is particularly well-suited to Go: The rules of the game are translationally invariant (matching the weight-sharing structure of convolutional networks) and are defined in terms of liberties corresponding to the adjacencies between points on the board (matching the local structure of convolutional networks). By contrast, the rules of chess and shogi are position dependent (e.g., pawns may move two steps forward from the second rank and promote on the eighth rank) and include long-range interactions (e.g., the queen may traverse the board in one move). Despite these differences, AlphaZero uses the same convolutional network architecture as AlphaGo Zero for chess, shogi, and Go.

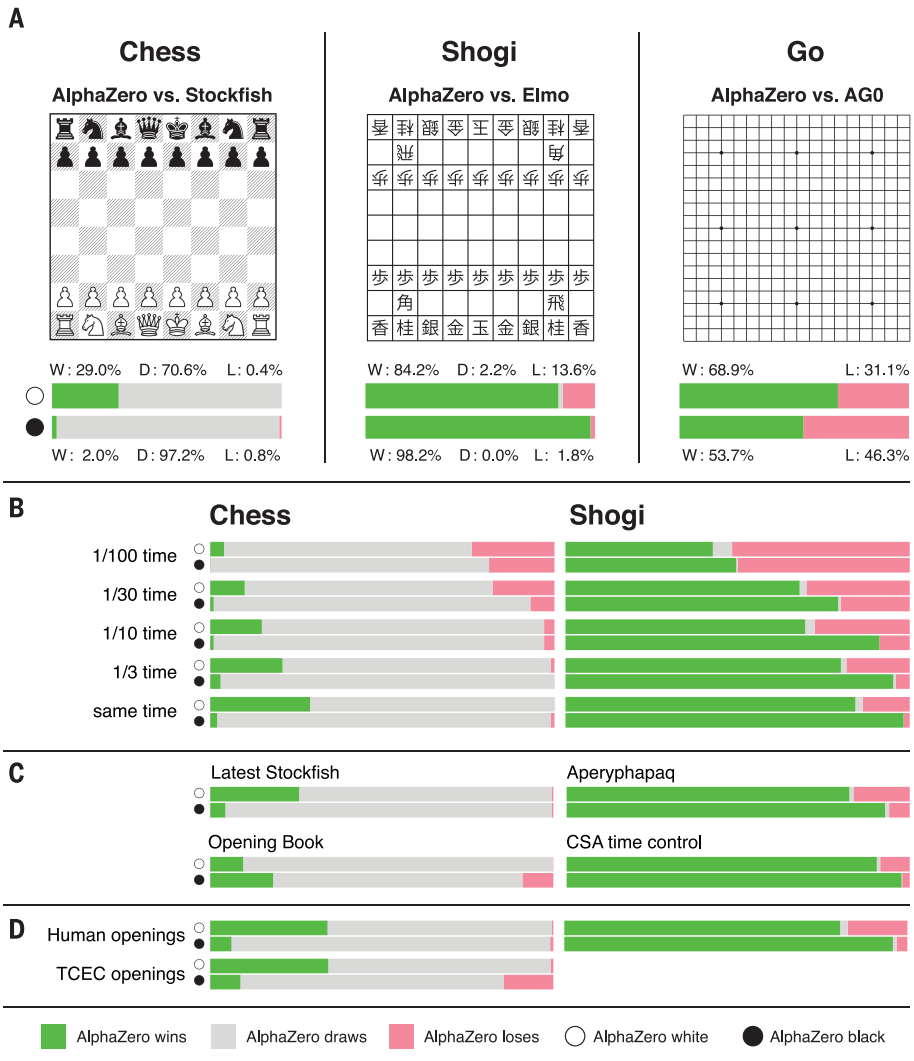

Fig. 2. Comparison with specialized programs. (**A**) Tournament evaluation of AlphaZero in chess, shogi, and Go in matches against, respectively, Stockfish, Elmo, and the previously published version of AlphaGo Zero (AG0) that was trained for 3 days. In the top bar, AlphaZero plays white; in the bottom bar, AlphaZero plays black. Each bar shows the results from AlphaZero's perspective: win (W; green), draw (D; gray), or loss (L; red). (**B**) Scalability of AlphaZero with thinking time compared with Stockfish and Elmo. Stockfish and Elmo always receive full time (3 hours per game plus 15 s per move); time for AlphaZero is scaled down as indicated. (**C**) Extra evaluations of AlphaZero in chess against the most recent version of Stockfish at the time of writing (*27*) and against Stockfish with a strong opening book (*28*). Extra evaluations of AlphaZero in shogi were carried out against another strong shogi program, Aperyqhapaq (*29*), at full time controls and against Elmo under 2017 CSA world championship time controls (10 min per game and 10 s per move). (**D**) Average result of chess matches starting from different opening positions, either common human positions (see also Fig. 3) or the 2016 TCEC world championship opening positions (see also fig. S4), and average result of shogi matches starting from common human positions (see also Fig. 3). CSA world championship games start from the initial board position. Match conditions are summarized in tables S8 and S9.

The hyperparameters of AlphaGo Zero were tuned by Bayesian optimization. In AlphaZero, we reuse the same hyperparameters, algorithm settings, and network architecture for all games without game-specific tuning. The only exceptions are the exploration noise and the learning rate schedule [see (*10*) for further details].

We trained separate instances of AlphaZero for chess, shogi, and Go. Training proceeded for 700,000 steps (in mini-batches of 4096 training positions) starting from randomly initialized parameters. During training only, 5000 first-generation tensor processing units (TPUs) (*19*) were used to generate self-play games, and 16 second-generation TPUs were used to train the neural networks. Training lasted for approximately 9 hours in chess, 12 hours in shogi, and 13 days in Go (see table S3) (*20*). Further details of the training procedure are provided in (*10*).

Figure 1 shows the performance of AlphaZero during self-play reinforcement learning, as a

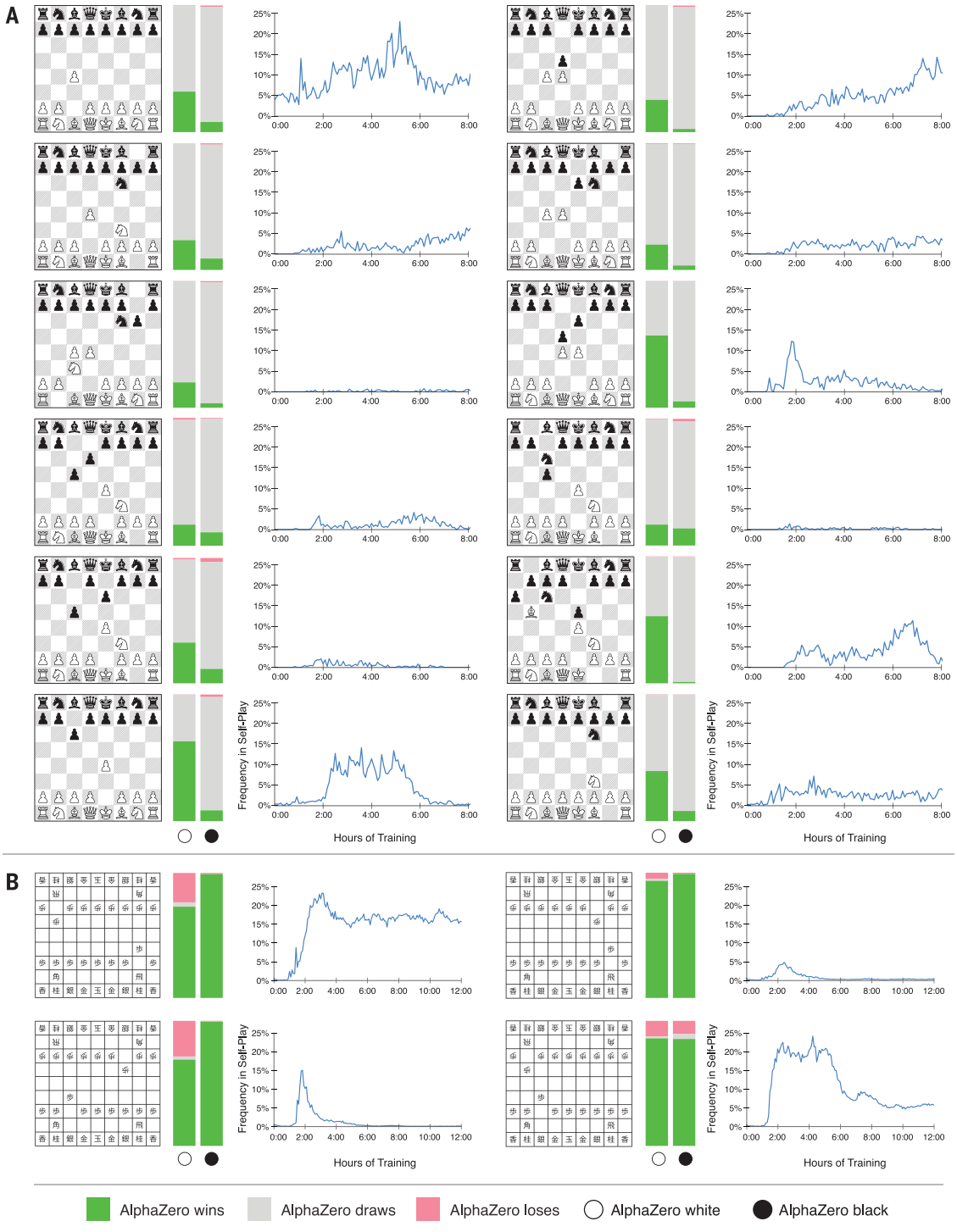

Fig. 3. Matches starting from the most popular human openings. AlphaZero plays against (**A**) Stockfish in chess and (**B**) Elmo in shogi. In the left bar, AlphaZero plays white, starting from the given position; in the right bar, AlphaZero plays black. Each bar shows the results from AlphaZero's perspective: win (green), draw (gray), or loss (red). The percentage frequency of self-play training games in which this opening was selected by AlphaZero is plotted against the duration of training, in hours.

function of training steps, on an Elo (*21*) scale (*22*). In chess, AlphaZero first outperformed Stockfish after just 4 hours (300,000 steps); in shogi, AlphaZero first outperformed Elmo after 2 hours (110,000 steps); and in Go, AlphaZero first outperformed AlphaGo Lee (*9*) after 30 hours (74,000 steps). The training algorithm achieved similar performance in all independent runs (see fig. S3), suggesting that the high performance of AlphaZero's training algorithm is repeatable.

We evaluated the fully trained instances of AlphaZero against Stockfish, Elmo, and the previous version of AlphaGo Zero in chess, shogi, and Go, respectively. Each program was run on the hardware for which it was designed (*23*): Stockfish and Elmo used 44 central processing unit (CPU) cores (as in the TCEC world championship), whereas AlphaZero and AlphaGo Zero used a single machine with four first-generation TPUs and 44 CPU cores (*24*). The chess match was played against the 2016 TCEC (season 9) world champion Stockfish [see (*10*) for details]. The shogi match was played against the 2017 CSA world champion version of Elmo (*10*). The Go match was played against the previously published version of AlphaGo Zero [also trained for 700,000 steps (*25*)]. All matches were played by using time controls of 3 hours per game, plus an additional 15 s for each move.

In Go, AlphaZero defeated AlphaGo Zero (*9*), winning 61% of games. This demonstrates that a general approach can recover the performance of an algorithm that exploited board symmetries to generate eight times as much data (see fig. S1).

In chess, AlphaZero defeated Stockfish, winning 155 games and losing 6 games out of 1000 (Fig. 2). To verify the robustness of AlphaZero, we played additional matches that started from common human openings (Fig. 3). AlphaZero defeated Stockfish in each opening, suggesting that AlphaZero has mastered a wide spectrum of chess play. The frequency plots in Fig. 3 and the time line in fig. S2 show that common human openings were independently discovered and played frequently by AlphaZero during self-play training. We also played a match that started from the set of opening positions used in the 2016 TCEC world championship; AlphaZero won convincingly in this match, too (*26*) (fig. S4). We played additional matches against the most recent development version of Stockfish (*27*) and a variant of Stockfish that uses a strong opening book (*28*). AlphaZero won all matches by a large margin (Fig. 2).

Table S6 shows 20 chess games played by AlphaZero in its matches against Stockfish. In several games, AlphaZero sacrificed pieces for long-term strategic advantage, suggesting that it has a more fluid, context-dependent positional evaluation than the rule-based evaluations used by previous chess programs.

In shogi, AlphaZero defeated Elmo, winning 98.2% of games when playing black and 91.2% overall. We also played a match under the faster time controls used in the 2017 CSA world championship and against another state-of-the-art shogi program (*29*); AlphaZero again won both matches by a wide margin (Fig. 2).

Table S7 shows 10 shogi games played by AlphaZero in its matches against Elmo. The frequency plots in Fig. 3 and the time line in fig. S2 show that AlphaZero frequently plays one of the two most common human openings but rarely plays the second, deviating on the very first move.

AlphaZero searches just 60,000 positions per second in chess and shogi, compared with 60 million for Stockfish and 25 million for Elmo (table S4). AlphaZero may compensate for the lower number of evaluations by using its deep neural network to focus much more selectively on the most promising variations (Fig. 4 provides an example from the match against Stockfish)—arguably a more humanlike approach to searching, as originally proposed by Shannon (*30*). AlphaZero also defeated Stockfish when given $^{1}/_{10}$ as much thinking time as its opponent (i.e., searching $^{\sim 1}/_{10,000}$ as many positions) and won 46% of games against Elmo when given $^{1}/_{100}$ as much time (i.e., searching $^{\sim 1}/_{40,000}$ as many positions) (Fig. 2). The high performance

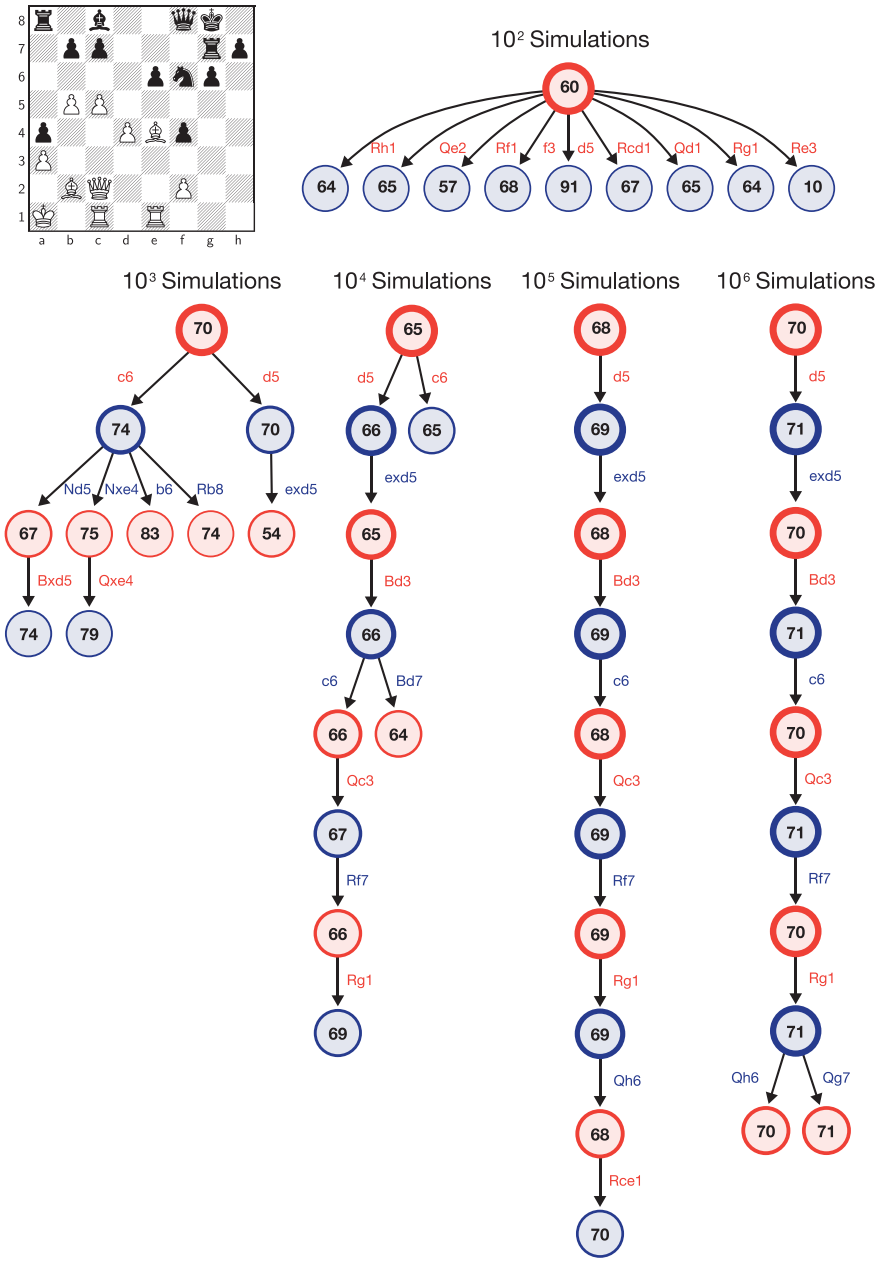

Fig. 4. AlphaZero's search procedure. The search is illustrated for a position (**inset**) from game 1 (table S6) between AlphaZero (white) and Stockfish (black) after 29. ... Qf8. The internal state of AlphaZero's MCTS is summarized after 10^2, ..., 10^6 simulations. Each summary shows the 10 most visited states. The estimated value is shown in each state, from white's perspective, scaled to the range [0, 100]. The visit count of each state, relative to the root state of that tree, is proportional to the thickness of the border circle. AlphaZero considers 30. c6 but eventually plays 30. d5.

of AlphaZero with the use of MCTS calls into question the widely held belief (*31*, *32*) that alpha-beta search is inherently superior in these domains.

The game of chess represented the pinnacle of artificial intelligence research over several decades. State-of-the-art programs are based on powerful engines that search many millions of positions, leveraging handcrafted domain expertise and sophisticated domain adaptations. AlphaZero is a generic reinforcement learning and search algorithm—originally devised for the game of Go—that achieved superior results within a few hours, searching $^{1}/_{1000}$ as many positions, given no domain knowledge except the rules of chess. Furthermore, the same algorithm was applied without modification to the more challenging game of shogi, again outperforming state-of-the-art programs within a few hours. These results bring us a step closer to fulfilling a longstanding ambition of artificial intelligence (*3*): a general game-playing system that can learn to master any game.

REFERENCES AND NOTES

1. M. Campbell, A. J. Hoane Jr., F. Hsu, *Artif. Intell.* **134**, 57–83 (2002).
2. F.-H. Hsu, *Behind Deep Blue: Building the Computer That Defeated the World Chess Champion* (Princeton Univ., 2002).
3. B. Pell, *Comput. Intell.* **12**, 177–198 (1996).
4. M. R. Genesereth, N. Love, B. Pell, *AI Mag.* **26**, 62–72 (2005).
5. A. L. Samuel, *IBM J. Res. Dev.* **11**, 601–617 (1967).
6. G. Tesauro, *Neural Comput.* **6**, 215–219 (1994).
7. C. J. Maddison, A. Huang, I. Sutskever, D. Silver, paper presented at the International Conference on Learning Representations 2015, San Diego, CA, 7 to 9 May 2015.
8. D. Silver *et al*., *Nature* **529**, 484–489 (2016).
9. D. Silver *et al*., *Nature* **550**, 354–359 (2017).
10. See the supplementary materials for additional information.
11. Stockfish: Strong open source chess engine; https://stockfishchess.org/ [accessed 29 November 2017].
12. D. N. L. Levy, M. Newborn, *How Computers Play Chess* (Ishi Press, 2009).
13. V. Allis, "Searching for solutions in games and artificial intelligence," Ph.D. thesis, Transnational University Limburg, Maastricht, Netherlands (1994).
14. H. Iida, M. Sakuta, J. Rollason, *Artif. Intell.* **134**, 121–144 (2002).
15. Computer Shogi Association, Results of the 27th world computer shogi championship; www2.computer-shogi.org/wcsc27/index_e.html [accessed 29 November 2017].
16. W. Steinitz, *The Modern Chess Instructor* (Edition Olms, 1990).
17. E. Lasker, *Common Sense in Chess* (Dover Publications, 1965).
18. J. Knudsen, *Essential Chess Quotations* (iUniverse, 2000).
19. N. P. Jouppi *et al.*, in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, Toronto, Canada, 24 to 28 June 2017 (Association for Computing Machinery, 2017), pp. 1–12.
20. Note that the original AlphaGo Zero study used graphics processing units (GPUs) to train the neural networks.
21. R. Coulom, in *Proceedings of the Sixth International Conference on Computers and Games*, Beijing, China, 29 September to 1 October 2008 (Springer, 2008), pp. 113–124.
22. The prevalence of draws in high-level chess tends to compress the Elo scale, compared with that for shogi or Go.
23. Stockfish is designed to exploit CPU hardware and cannot make use of GPUs or TPUs, whereas AlphaZero is designed to exploit GPU-TPU hardware rather than CPU hardware.
24. A first generation TPU is roughly similar in inference speed to a Titan V GPU, although the architectures are not directly comparable.
25. AlphaGo Zero was ultimately trained for 3.1 million steps over 40 days.
26. Many TCEC opening positions are unbalanced according to both AlphaZero and Stockfish, resulting in more losses for both players.
27. Newest available version of Stockfish as of 13 January 2018 (resolved_base b508f9561cc2302c129efe8d60f201ff03ee72c8), from https://github.com/official-stockfish/Stockfish/commit/.
28. The Stockfish variant used the Cerebellum opening book downloaded from https://zipproth.de/#Brainfish. AlphaZero did not use an opening book. To ensure diversity against a deterministic opening book, AlphaZero used a small amount of randomization in its opening moves (*10*); this avoided duplicate games but also resulted in more losses.
29. Aperyqhapaq's evaluation files are available at https://github.com/qhapaq-49/qhapaq-bin/releases/tag/eloqhappa.
30. C. E. Shannon, *London Edinburgh Dublin Philos. Mag. J. Sci.* **41**, 256–275 (1950).
31. O. Arenz, "Monte Carlo chess," master's thesis, Technische Universität Darmstadt (2012).
32. O. E. David, N. S. Netanyahu, L. Wolf, in *Artificial Neural Networks and Machine Learning—ICANN 2016, Part II*, Barcelona, Spain, 6 to 9 September 2016 (Springer, 2016), pp. 88–96.

ACKNOWLEDGMENTS

We thank M. Sadler for analyzing chess games; Y. Habu for analyzing shogi games; L. Bennett for organizational assistance; B. Konrad, E. Lockhart, and G. Ostrovski for reviewing the paper; and the rest of the DeepMind team for their support. **Funding:** All research described in this report was funded by DeepMind and Alphabet. **Author contributions:** D.S., J.S., T.H., and I.A. designed the AlphaZero algorithm with advice from T.G., A.G., T.L., K.S., M.Lai, L.S., and M.Lan.; J.S., I.A., T.H., and M.Lai implemented the AlphaZero program; T.H., J.S., D.S., M.Lai, I.A., T.G., K.S., D.K., and D.H. ran experiments and/or analyzed data; D.S., T.H., J.S., and D.H. managed the project; D.S., J.S., T.H., M.Lai, I.A., and D.H. wrote the paper. **Competing interests:** DeepMind has filed the following patent applications related to this work: PCT/EP2018/063869, US15/280,711, and US15/280,784. **Data and materials availability:** A full description of the algorithm in pseudocode as well as details of additional games between AlphaZero and other programs is available in the supplementary materials.

SUPPLEMENTARY MATERIALS

www.sciencemag.org/content/362/6419/1140/suppl/DC1
Materials and Methods
Figs. S1 to S4
Tables S1 to S9
References (*33*–*50*)
Data S1

2 March 2018; accepted 7 November 2018
10.1126/science.aar6404