We use cookies to improve your experience with our site. Accept and close | More info.

ARTICLE PREVIEW

view full access options

NATURE **| ARTICLE**

日本語要約

Mastering the game of Go with deep neural networks and tree search

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

Nature **529**, 484–489 (28 January 2016) doi:10.1038/nature16961
Received 11 November 2015 Accepted 05 January 2016 Published online 27 January 2016

Abstract

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses ‘value networks’ to evaluate board positions and ‘policy networks’ to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

Subject terms: Computer science Computational science Reward

READ THE FULL ARTICLE

Subscribe to *Nature* for full access:	ReadCube Access*:	Purchase article full text and PDF:
$199	**$4.99 rent** **$9.99 buy** *printing and sharing restrictions apply	**$32**
Subscribe	Buy/Rent now	Buy now

Already a subscriber? Log in now or Register for online access.

Additional access options:
Use a document delivery service | Rent for $4.00 at DeepDyve | Login via OpenAthens | Purchase a site license | Institutional access

We use cookies to improve your experience with our site. Accept and close | More info.

References

1. Allis, L. V. *Searching for Solutions in Games and Artificial Intelligence.* PhD thesis, Univ. Limburg, Maastricht, The Netherlands (1994)
2. van den Herik, H., Uiterwijk, J. W. & van Rijswijck, J. Games solved: now and in the future. *Artif. Intell.* **134**, 277–311 (2002)
3. Schaeffer, J. The games computers (and people) play. *Advances in Computers* **52**, 189–266 (2000)
4. Campbell, M., Hoane, A. & Hsu, F. Deep Blue. *Artif. Intell.* **134**, 57–83 (2002)
5. Schaeffer, J. *et al.* A world championship caliber checkers program. *Artif. Intell.* **53**, 273–289 (1992)
6. Buro, M. From simple features to sophisticated evaluation functions. In *1st International Conference on Computers and Games*, 126–145 (1999)
7. Müller, M. Computer Go. *Artif. Intell.* **134**, 145–179 (2002)
8. Tesauro, G. & Galperin, G. On-line policy improvement using Monte-Carlo search. In *Advances in Neural Information Processing*, 1068–1074 (1996)
9. Sheppard, B. World-championship-caliber Scrabble. *Artif. Intell.* **134**, 241–275 (2002)
10. Bouzy, B. & Helmstetter, B. Monte-Carlo Go developments. In *10th International Conference on Advances in Computer Games*, 159–174 (2003)
11. Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In *5th International Conference on Computers and Games*, 72–83 (2006)
12. Kocsis, L. & Szepesvári, C. Bandit based Monte-Carlo planning. In *15th European Conference on Machine Learning*, 282–293 (2006)
13. Coulom, R. Computing Elo ratings of move patterns in the game of Go. *ICGA J.* **30**, 198–208 (2007)
14. Baudiš, P. & Gailly, J.-L. Pachi: State of the art open source Go program. In *Advances in Computer Games*, 24–38 (Springer, 2012)
15. Müller, M., Enzenberger, M., Arneson, B. & Segal, R. Fuego – an open-source framework for board games and Go engine based on Monte-Carlo tree search. *IEEE Trans. Comput. Intell. AI in Games* **2**, 259–270 (2010)
16. Gelly, S. & Silver, D. Combining online and offline learning in UCT. In *17th International Conference on Machine Learning*, 273–280 (2007)
17. Krizhevsky, A., Sutskever, I. & Hinton, G. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 1097–1105 (2012)
18. Lawrence, S., Giles, C. L., Tsoi, A. C. & Back, A. D. Face recognition: a convolutional neural-network approach. *IEEE Trans. Neural Netw.* **8**, 98–113 (1997)
19. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518**, 529–533 (2015)
20. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015)
21. Stern, D., Herbrich, R. & Graepel, T. Bayesian pattern ranking for move prediction in the game of Go. In *International Conference of Machine Learning*, 873–880 (2006)
22. Sutskever, I. & Nair, V. Mimicking Go experts with convolutional neural networks. In *International Conference on Artificial Neural Networks*, 101–110 (2008)
23. Maddison, C. J., Huang, A., Sutskever, I. & Silver, D. Move evaluation in Go using deep convolutional neural networks. *3rd International Conference on Learning Representations* (2015)
24. Clark, C. & Storkey, A. J. Training deep convolutional neural networks to play go. In *32nd International Conference on Machine Learning*, 1766–1774 (2015)
25. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.* **8**, 229–256 (1992)

We use cookies to improve your experience with our site. Accept and close | More info.

26. Sutton, R., McAllester, D., Singh, S. & Mansour, Y. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, 1057–1063 (2000)

27. Sutton, R. & Barto, A. *Reinforcement Learning: an Introduction* (MIT Press, 1998)

28. Schraudolph, N. N., Dayan, P. & Sejnowski, T. J. Temporal difference learning of position evaluation in the game of Go. *Adv. Neural Inf. Process. Syst.* **6**, 817–824 (1994)

29. Enzenberger, M. Evaluation in Go by a neural network using soft segmentation. In *10th Advances in Computer Games Conference*, 97–108 (2003). 267

30. Silver, D., Sutton, R. & Müller, M. Temporal-difference search in computer Go. *Mach. Learn.* **87**, 183–219 (2012)

31. Levinovitz, A. The mystery of Go, the ancient game that computers still can't win. *Wired Magazine* (2014)

32. Mechner, D. All Systems Go. *The Sciences* **38**, 32–37 (1998)

33. Mandziuk, J. Computational intelligence in mind games. In *Challenges for Computational Intelligence*, 407–442 (2007)

34. Berliner, H. A chronology of computer chess and its literature. *Artif. Intell.* **10**, 201–214 (1978)

35. Browne, C. *et al.* A survey of Monte-Carlo tree search methods. *IEEE Trans. Comput. Intell. AI in Games* **4**, 1–43 (2012)

36. Gelly, S. *et al.* The grand challenge of computer Go: Monte Carlo tree search and extensions. *Commun. ACM* **55**, 106–113 (2012)

37. Coulom, R. Whole-history rating: A Bayesian rating system for players of time-varying strength. In *International Conference on Computers and Games*, 113–124 (2008)

38. KGS. Rating system math. http://www.gokgs.com/help/rmath.html

39. Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *11th International Conference on Machine Learning*, 157–163 (1994)

40. Knuth, D. E. & Moore, R. W. An analysis of alpha-beta pruning. *Artif. Intell.* **6**, 293–326 (1975)

41. Sutton, R. Learning to predict by the method of temporal differences. *Mach. Learn.* **3**, 9–44 (1988)

42. Baxter, J., Tridgell, A. & Weaver, L. Learning to play chess using temporal differences. *Mach. Learn.* **40**, 243–263 (2000)

43. Veness, J., Silver, D., Blair, A. & Uther, W. Bootstrapping from game tree search. In *Advances in Neural Information Processing Systems* (2009)

44. Samuel, A. L. Some studies in machine learning using the game of checkers II - recent progress. *IBM J. Res. Develop.* **11**, 601–617 (1967)

45. Schaeffer, J., Hlynka, M. & Jussila, V. Temporal difference learning applied to a high-performance game-playing program. In *17th International Joint Conference on Artificial Intelligence*, 529–534 (2001)

46. Tesauro, G. TD-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* **6**, 215–219 (1994)

47. Dahl, F. Honte, a Go-playing program using neural nets. In *Machines that learn to play games*, 205–223 (Nova Science, 1999)

48. Rosin, C. D. Multi-armed bandits with episode context. *Ann. Math. Artif. Intell.* **61**, 203–230 (2011)

49. Lanctot, M., Winands, M. H. M., Pepels, T. & Sturtevant, N. R. Monte Carlo tree search with heuristic evaluations using implicit minimax backups. In *IEEE Conference on Computational Intelligence and Games*, 1–8 (2014)

50. Gelly, S., Wang, Y., Munos, R. & Teytaud, O. Modification of UCT with patterns in Monte-Carlo Go. *Tech. Rep.* **6062**, INRIA (2006)

51. Silver, D. & Tesauro, G. Monte-Carlo simulation balancing. In *26th International Conference on Machine Learning*, **119** (2009)

52. Huang, S.-C., Coulom, R. & Lin, S.-S. Monte-Carlo simulation balancing in practice. In *7th International Conference on Computers and Games*, 81–92 (Springer-Verlag, 2011)

53. Baier, H. & Drake, P. D. The power of forgetting: improving the last-good-reply policy in Monte Carlo Go. *IEEE Trans. Comput. Intell. AI in Games* **2**, 303–309 (2010)

We use cookies to improve your experience with our site. Accept and close | More info.

54. Huang, S. & Müller, M. Investigating the limits of Monte-Carlo tree search methods in computer Go. In *8th International Conference on Computers and Games*, 39–48 (2013)

55. Segal, R. B. On the scalability of parallel UCT. *Computers and Games* **6515**, 36–47 (2011)

56. Enzenberger, M. & Müller, M. A lock-free multithreaded Monte-Carlo tree search algorithm. In *12th Advances in Computer Games Conference*, 14–20 (2009)

57. Huang, S.-C., Coulom, R. & Lin, S.-S. Time management for Monte-Carlo tree search applied to the game of Go. In *International Conference on Technologies and Applications of Artificial Intelligence*, 462–466 (2010)

58. Gelly, S. & Silver, D. Monte-Carlo tree search and rapid action value estimation in computer Go. *Artif. Intell.* **175**, 1856–1875 (2011)

59. Baudiš, P. Balancing MCTS by dynamically adjusting the komi value. *ICGA J.* **34**, 131 (2011)

60. Baier, H. & Winands, M. H. Active opening book application for Monte-Carlo tree search in 19×19 Go. In *Benelux Conference on Artificial Intelligence*, 3–10 (2011)

61. Dean, J. *et al.* Large scale distributed deep networks. In *Advances in Neural Information Processing Systems*, 1223–1231 (2012)

62. Go ratings. http://www.goratings.org

Download references

Author information

These authors contributed equally to this work.
David Silver & Aja Huang

Affiliations

Google DeepMind, 5 New Street Square, London EC4A 3TW, UK
David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, Nal Kalchbrenner, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

Google, 1600 Amphitheatre Parkway, Mountain View, California 94043, USA
John Nham & Ilya Sutskever

Contributions
A.H., G.v.d.D., J.S., I.A., M.La., A.G., T.G. and D.S. designed and implemented the search in AlphaGo. C.J.M., A.G., L.S., A.H., I.A., V.P., S.D., D.G., N.K., I.S., K.K. and D.S. designed and trained the neural networks in AlphaGo. J.S., J.N., A.H. and D.S. designed and implemented the evaluation framework for AlphaGo. D.S., M.Le., T.L., T.G., K.K. and D.H. managed and advised on the project. D.S., T.G., A.G. and D.H. wrote the paper.

Competing financial interests
The authors declare no competing financial interests.

Corresponding authors
Correspondence to: David Silver or Demis Hassabis

Extended data figures and tables

Extended Data Tables

1. Extended Data Table 1: Details of match between AlphaGo and Fan Hui (194 KB)

2. Extended Data Table 2: Input features for neural networks (156 KB)

3. Extended Data Table 3: Supervised learning results for the policy network (109 KB)

We use cookies to improve your experience with our site. Accept and close | More info.

4. Extended Data Table 4: Input features for rollout and tree policy (132 KB)
5. Extended Data Table 5: Parameters used by AlphaGo (71 KB)
6. Extended Data Table 6: Results of a tournament between different Go programs (108 KB)
7. Extended Data Table 7: Results of a tournament between different variants of AlphaGo (62 KB)
8. Extended Data Table 8: Results of a tournament between AlphaGo and distributed AlphaGo, testing scalability with hardware (103 KB)
9. Extended Data Table 9: Cross-table of win rates in per cent between programs (192 KB)
10. Extended Data Table 10: Cross-table of win rates in per cent between programs in the single-machine scalability study (153 KB)
11. Extended Data Table 11: Cross-table of win rates in per cent between programs in the distributed scalability study (117 KB)

Supplementary information

Zip files

1. Supplementary Information (3.4 KB)
   This zipped file contains game records for the 5 formal match games played between AlphaGo and Fan Hui.

Comments

2016-01-29 10:09 AM

Justin Brown said: Can someone explain the process of expanding in more detail? I have a lot of questions that might be stemming from some more fundamental misunderstanding. An example like figure 3 that starts from a blank board and goes forward a few steps in more detail would be helpful.

*Here's what I know so far: *

- s is a configuration of the board and a node of the search tree. s' is a possible configuration of the board(but not yet a node in the search tree) that gets determined by f(s,a), where a is the action taken to produce s'. s' gets produced and added to the tree when (s,a) has been visited more than 40 times.
- The policy network takes in s, and outputs the probability of an action being taken next, for all legal actions in s
- The value network takes in s and outputs the chance of the player winning that game

Questions:

Starting from a blank board, how does (s,a) get added to the tree?

What counts as a visit to (s,a)? The algorithm only selects nodes with the highest Q and new nodes(s') only get added to the tree if (s,a) has been visited 40 times. Does that mean the node with the highest Q has to be selected 40 times before it can produce s'?

How do these two quotes make sense?

- *'The leaf node may be expanded; *the new node is processed once by the policy network p'* and the output probabilities are stored as prior probabilities P for each action.'*

We use cookies to improve your experience with our site. Accept and close | More info.

- *'When the traversal reaches a leaf node s~L~ at step L, the leaf node may be expanded. *The leaf position sL is processed just once by the SL policy network p'.*'*

It seems it's saying the leaf node, s~L~, and the new expanded node, s', both get processed by the policy network. In the second quote, did the leaf node change to become s'? Does that mean step L changed?

When a new node, s', gets created, it's prior probability gets initialized by the policy network p~?~^?^(-|s'). Is this any different than inputting s into the policy network and getting the probability that action a will be taken?

2016-03-15 08:14 AM

Tet Yoon Lee said: Looking at extended data table 8 or figure 4, was wondering about the big jump in Elo rating with the increase from 1 GPU to 2 GPUs. The improvement in Elo rating after an increase to 4 or 8 GPUs, or even to the distributed config are a lot more modest.There are a lot of possible reasons for this & I think I came up with a few of them myself before I wondered if I was missing the obvious.

As also shown in figure 4 and clearer in extended data table 7, it seems with the 8 GPU config you are using 2 GPUs for the policy network and 6 GPUs for the value network with rollouts and a mixing constant of 0.5 as it gives the best results. Considering the 8 GPUs and the distributed configs may have been where you concentrated most of your efforts; is the problem with 1 GPU that you are forced to limit it to value network only, or the sole GPU is dedicated to either the value or policy network, or if it is shared between both networks, perhaps your code isn't well optimised for this?

In any case, congratulations on your excellent research and success!

Subscribe to comments

Nature ISSN 0028-0836 EISSN 1476-4687

SPRINGER NATURE

© 2016 Macmillan Publishers Limited, part of Springer Nature. All rights reserved.
partner of AGORA, HINARI, OARE, INASP, ORCID, CrossRef, COUNTER and COPE