fun-ai-talks

Let's read 6 DeepMind Alpha* papers!

2016 AlphaGo	2017 AlphaGo Zero	2018 AlphaZero
2019 AlphaStar	2021 AlphaFold	2022 AlphaCode

hululu.zhu@gmail.com

05/2022

Agenda

- Intro
- AI for game
  - 2016 AlphaGo: AI for Go with human data
  - 2017 AlphaGo Zero: AI for Go without human data
  - 2018 AlphaZero: A general AI framework for Go, Chess, Shoji
  - 2019 AlphaStar: Multi-agent AI for Starcraft
- AI for science/engineering
  - 2021 AlphaFold: AI to predict protein folding 3d structure
  - 2022 AlphaCode: AI to solve competitional coding problems
- Summary

How are you today?

About me, 10+ years at Google

My ML work experience starts at 2016 at Google Fiber

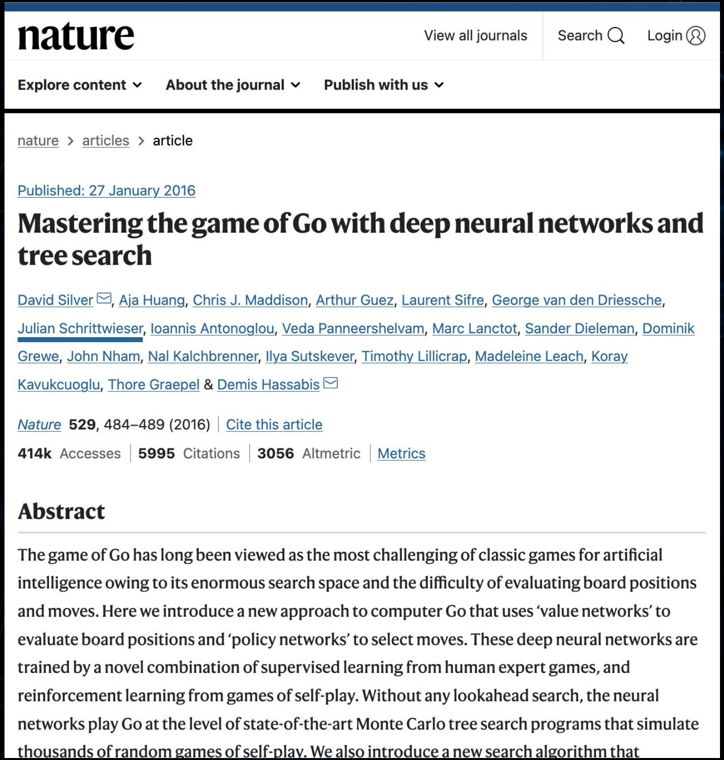
intern

SWE

Why I create this course?

- DeepMind published a few breakthrough research over the years since 2015
  - Some of their best work is named with *Alpha** pattern
  - It seems meaningful to combine them into a shared talk!
- To my best knowledge, nobody has made a course/deck to summarize all the Alpha* papers as of 05/2022, so I will try
- And sharing is fun!
  - Sharing and simplification for more in-depth learning
  - go/fun-ai-courses is my personal project for ML education

AlphaGo 2016



nature View all journals Search Login

Explore content About the journal Publish with us

nature > articles > article

Published: 27 January 2016

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

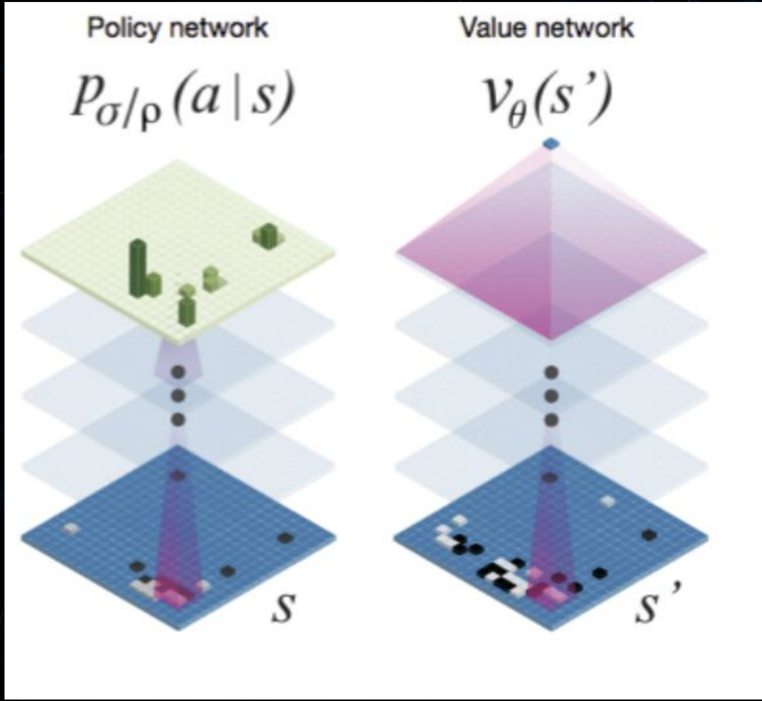
Nature **529**, 484–489 (2016) | Cite this article

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

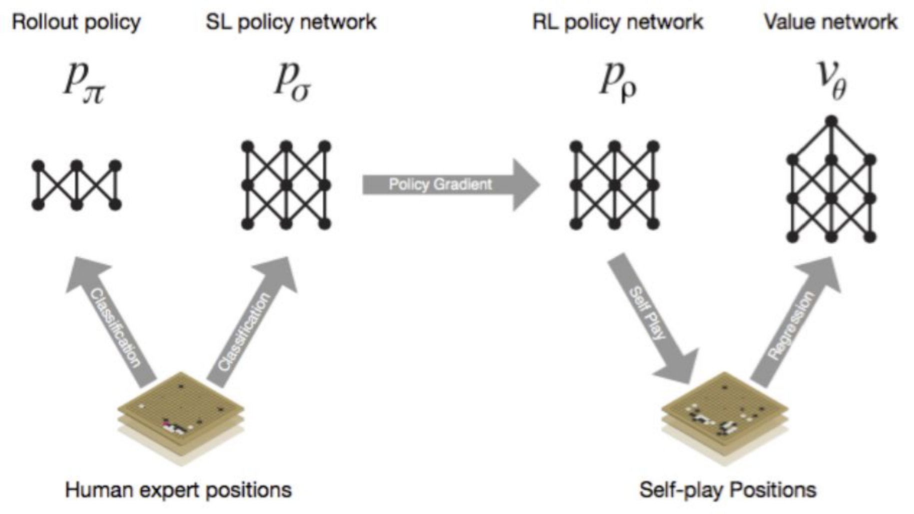
- AI for Go game is hard!
- 2 types of networks
  - Value Network
  - Policy Network
  - Monte Carlo Tree Search to effectively combine value and policy networks
- 2 learnings applied
  - supervised learning
  - reinforcement learning

AlphaGo 2016, 2 networks: policy and value



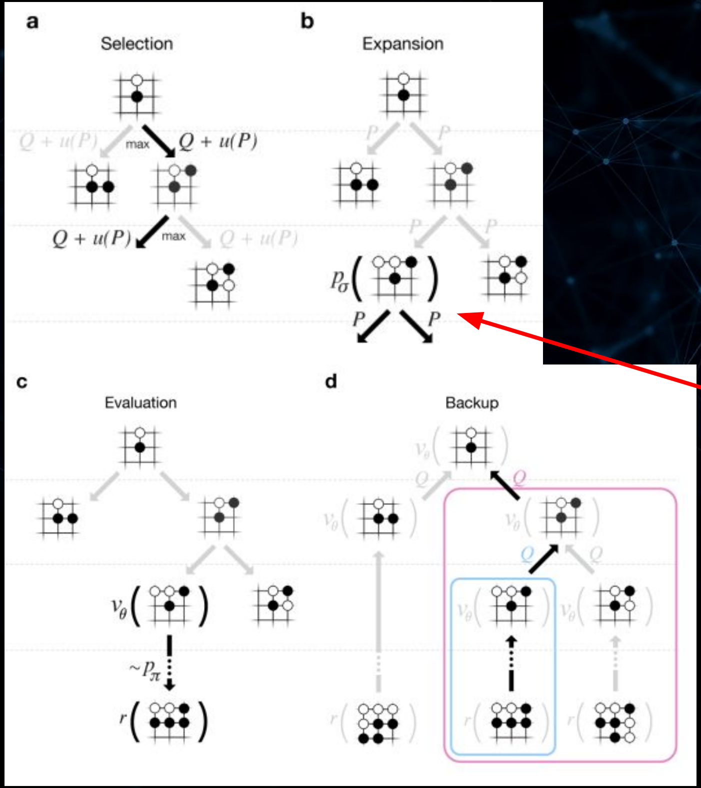
- CNN Policy Network
  - Input: Encoded go board state features s (more than positions)
  - Output: Prob distribution of action a
- CNN Value Network
  - Input: Encoded go board state features s
  - Output: A single scalar value to predict win or loss

AlphaGo 2016, Supervised Learning (SL) & Reinforcement learning (RL)



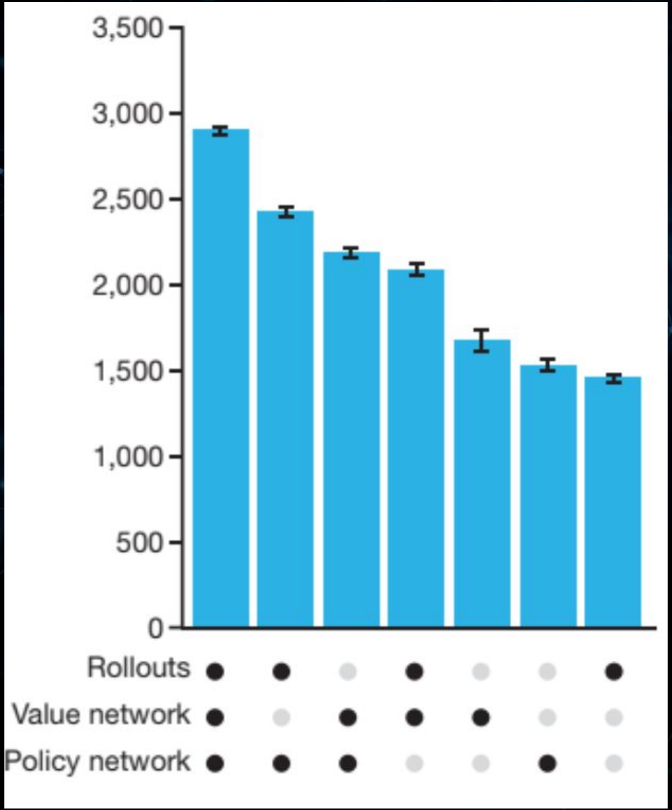
- Fast Rollout Policy network
  - A “simpler, less accurate” but 1000x faster network to imitate expert actions
- SL policy network
  - An accurate but slower network to better imitate expert actions
- RL policy network (self-play RL)
  - Initiated from SL policy network, optimize for “win” by self-play!
- Value network
  - Predict outcome from the RL policy network self-play

AlphaGo 2016, Monte Carlo Tree Search to “lookahead search”

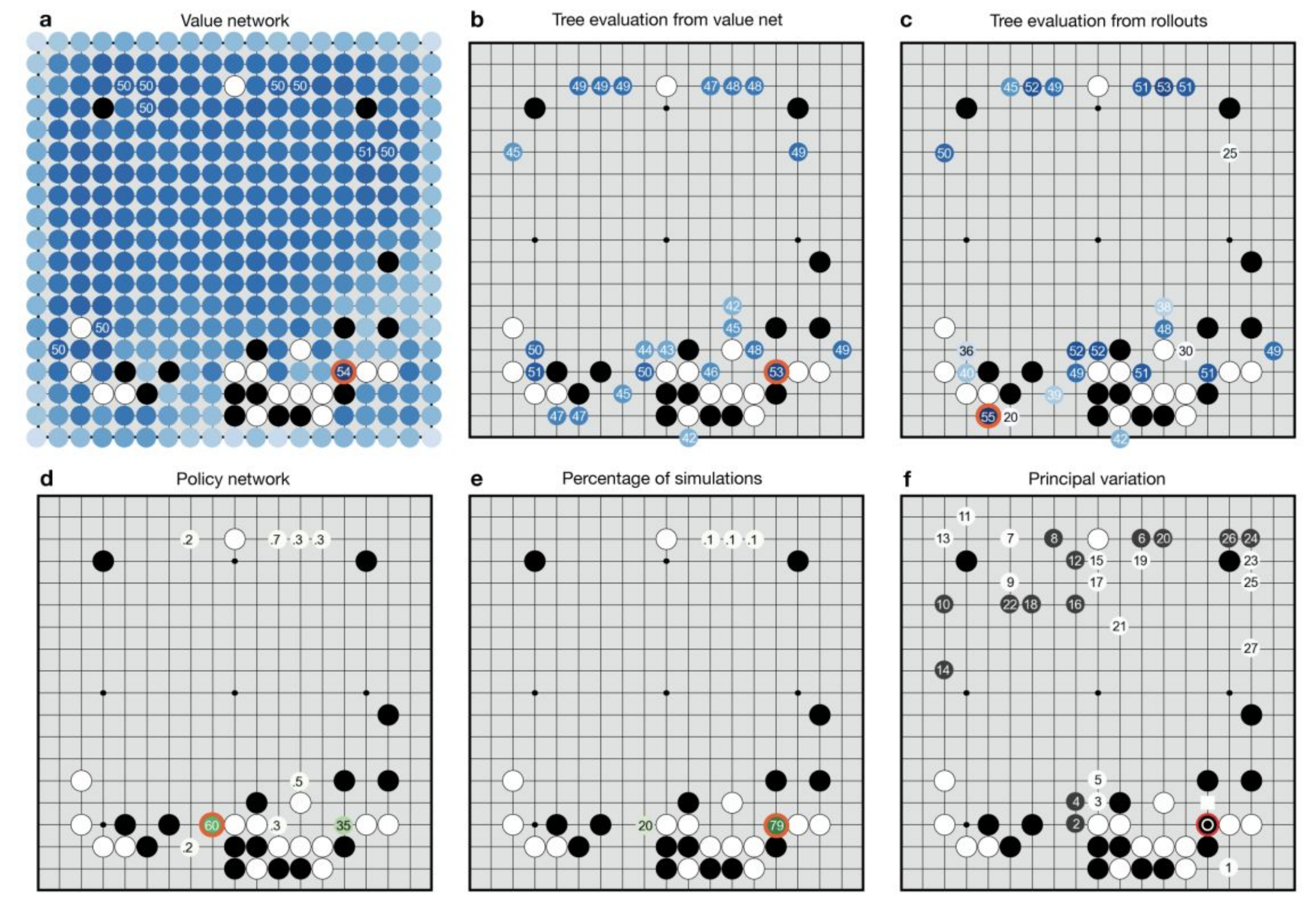


- MCTS is efficient to sample and approximate “optimal policy”
- Combined value network (predict win/loss) and fast rollout outcome (fast simulate to report win/loss)
    - Discounted by repeated visits to encourage “exploration”
- Note the supervised learning policy network is used to get prior action distribution!
    - SL network to imitate experts may “*select a diverse beam of promising moves*”, while RL network “*optimized for single best move*”

AlphaGo 2016, ablation analysis of different networks, all 3 networks are necessary!



AlphaGo 2016, example how MCTS works



AlphaGo Zero 2017

nature Search Login

Explore content ⌄ About the journal ⌄ Publish with us ⌄

Published: 19 October 2017

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

Nature **550**, 354–359 (2017) | Cite this article

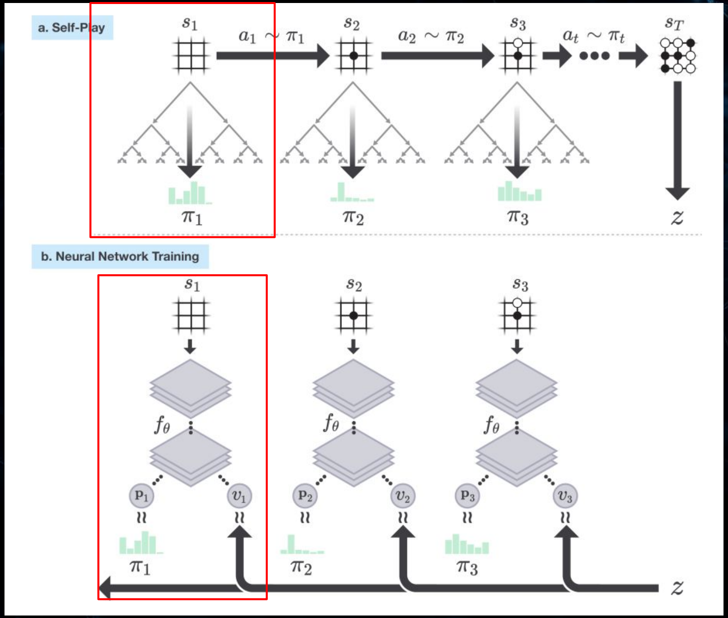
Abstract

A long-standing goal of artificial intelligence is an algorithm that learns, *tabula rasa*, superhuman proficiency in challenging domains. Recently, AlphaGo became the first program to defeat a world champion in the game of Go. The tree search in AlphaGo evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert moves, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules. AlphaGo becomes its own teacher: a neural network is trained to predict AlphaGo's own move selections and also the

- Simpler than AlphaGo
  - No human expert data!
  - Only board info as input
  - Shared value/policy network to predict win/loss and action distribution

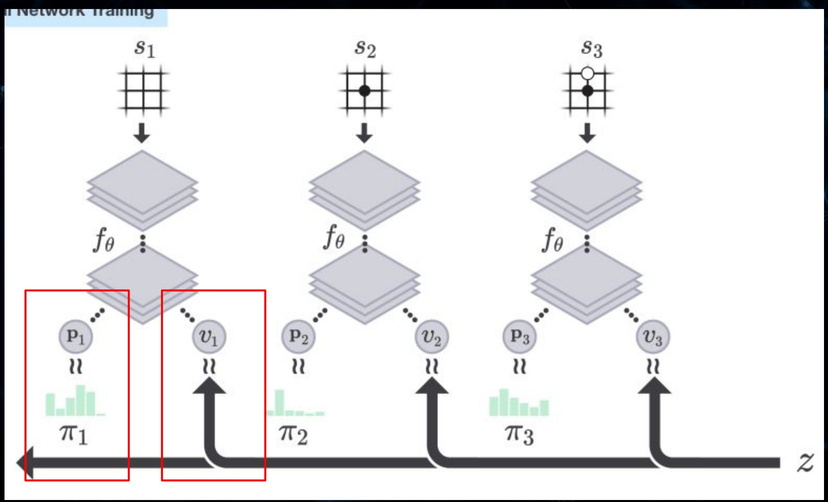
- Monte Carlo Tree Search merged into training
  - AlphaGo used MCTS with model inference only

AlphaGo Zero 2017, network and MCTS



- CNN + Residual block (resnet)
- Shared network outputs v_t (win prob), and ρ_t (action distribution)
- MCTS applied at t during self-play
  - Action dist ρ_t to expand tree
  - Value output v_t to predict win/loss
  - Combined scores by discounting visits to encourage exploration
  - Sampling is applied on MCTS in self-play (to decide a_t) instead of choosing max value
- Game stops at s_t with outcome z

AlphaGo Zero 2017, loss and training



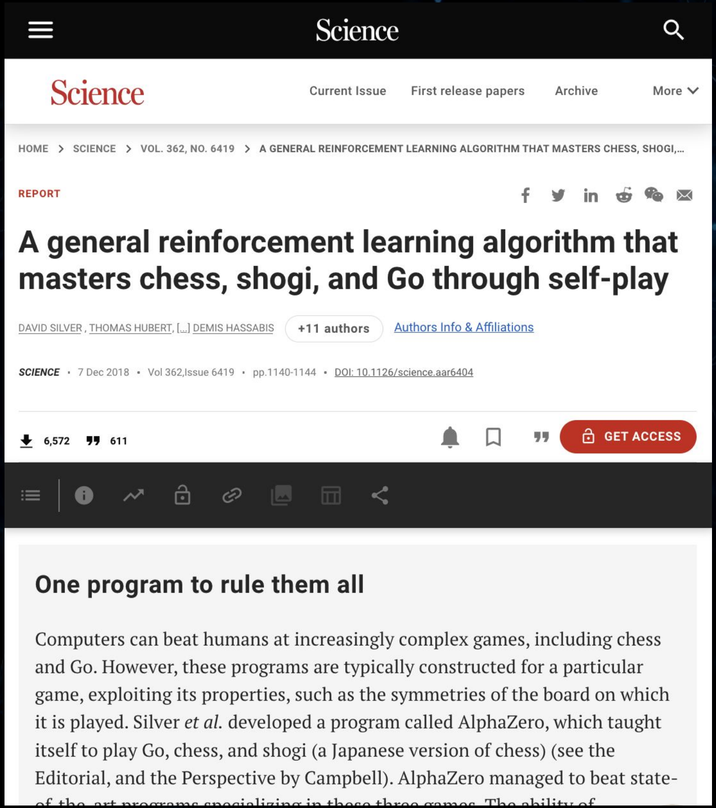
loss = $(z - v)^2 - \pi^T \log p + c||\theta||^2$

- "More accurately predict win/loss"
  - minimize($(z - v)^2$) so model can predict outcome more accurately
- "More align with the MCTS self-play algorithm"
  - Cross entropy ($- \pi T \log p$) so that p_t approximate MCTS distribution π_t, with a temperature param
- "Smaller weights", weight decay for ($c||\theta||^2$)

AlphaGo Zero 2017, training with "selected self play games" with "best player"!

Evaluator To ensure we always generate the best quality data, we evaluate each new neural network checkpoint against the current best network f_{θ_*} before using it for data generation. The neural network f_{θ_i} is evaluated by the performance of an MCTS search α_{θ_i} that uses f_{θ_i} to evaluate leaf positions and prior probabilities (see Search Algorithm). Each evaluation consists of 400 games, using an MCTS with 1,600 simulations to select each move, using an infinitesimal temperature $\tau \to 0$ (i.e. we deterministically select the move with maximum visit count, to give the strongest possible play). If the new player wins by a margin of $> 55\%$ (to avoid selecting on noise alone) then it becomes the best player α_{θ_*}, and is subsequently used for self-play generation, and also becomes the baseline for subsequent comparisons.

AlphaZero 2018



Science

Science

Current Issue First release papers Archive More

HOME > SCIENCE > VOL. 362, NO. 6419 > A GENERAL REINFORCEMENT LEARNING ALGORITHM THAT MASTERS CHESS, SHOGI,...

REPORT

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

DAVID SILVER, THOMAS HUBERT, [...] DEMIS HASSABIS +11 authors Authors Info & Affiliations

SCIENCE • 7 Dec 2018 • Vol 362,Issue 6419 • pp.1140-1144 • DOI: 10.1126/science.aar6404

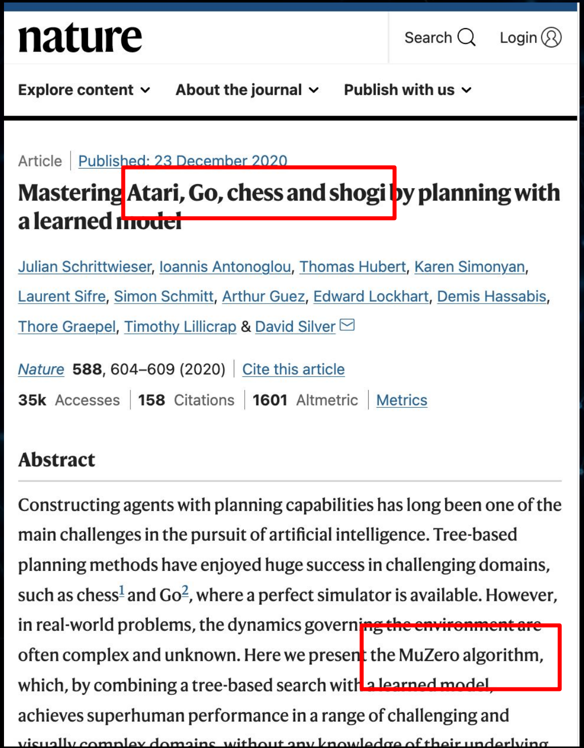
6,572 611 GET ACCESS

One program to rule them all

Computers can beat humans at increasingly complex games, including chess and Go. However, these programs are typically constructed for a particular game, exploiting its properties, such as the symmetries of the board on which it is played. Silver *et al.* developed a program called AlphaZero, which taught itself to play Go, chess, and shogi (a Japanese version of chess) (see the Editorial, and the Perspective by Campbell). AlphaZero managed to beat state-

- The same AI architecture can play chess, shogi, & Go
- Same arch as AlphaGo Zero
  - Same shared network to predict value and policy
- Different from AlphaGo Zero
  - Different size of board
  - No rotation/reflection
  - Non-binary outcome
  - “Continuous update” during self play, vs AlphaGo Zero only optimized on “best model” that won 55%+ over previous versions

AlphaZero 2018, future related work MuZero (2019) and Gato (2022, *not reinforcement learning*)



nature

Search Login

Explore content About the journal Publish with us

Article | Published: 23 December 2020

Mastering Atari, Go, chess and shogi by planning with a learned model

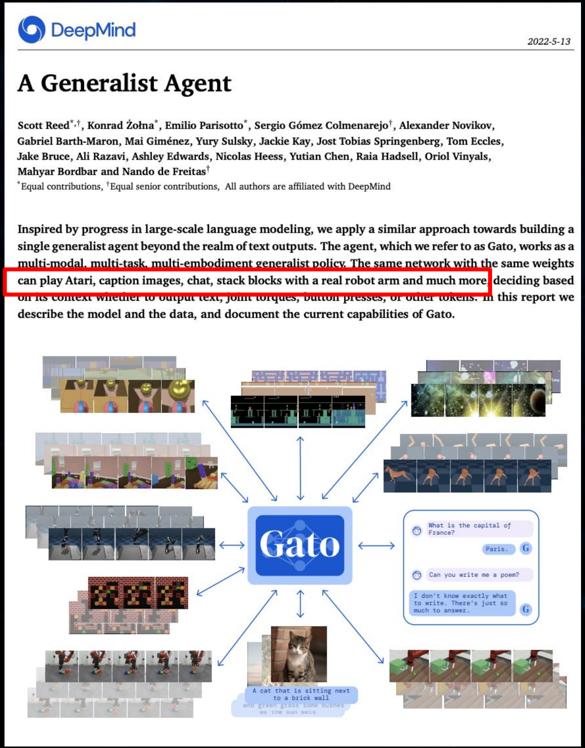
Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy Lillicrap & David Silver

Nature **588**, 604–609 (2020) | Cite this article

35k Accesses | **158** Citations | **1601** Altmetric | Metrics

Abstract

Constructing agents with planning capabilities has long been one of the main challenges in the pursuit of artificial intelligence. Tree-based planning methods have enjoyed huge success in challenging domains, such as chess[1] and Go[2], where a perfect simulator is available. However, in real-world problems, the dynamics governing the environment are often complex and unknown. Here we present the MuZero algorithm, which, by combining a tree-based search with a learned model, achieves superhuman performance in a range of challenging and visually complex domains, without any knowledge of their underlying



DeepMind 2022-5-13

A Generalist Agent

Scott Reed[*,†], Konrad Żołna[*], Emilio Parisotto[*], Sergio Gómez Colmenarejo[†], Alexander Novikov, Gabriel Barth-Maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar and Nando de Freitas[†]
[*]Equal contributions, [†]Equal senior contributions, All authors are affiliated with DeepMind

Inspired by progress in large-scale language modeling, we apply a similar approach towards building a single generalist agent beyond the realm of text outputs. The agent, which we refer to as Gato, works as a multi-modal, multi-task, multi-embodiment generalist policy. The same network with the same weights can play Atari, caption images, chat, stack blocks with a real robot arm and much more, deciding based on its context whether to output text, joint torques, button presses, or other tokens. In this report we describe the model and the data, and document the current capabilities of Gato.

AlphaStar 2019

nature | View all journals | Search | Login

Explore content | About the journal | Publish with us

nature > articles > article

Article | Published: 30 October 2019

Grandmaster level in StarCraft II using multi-agent reinforcement learning

Oriol Vinyals, Igor Babuschkin, … David Silver + Show authors

Nature **575**, 350–354 (2019) | Cite this article

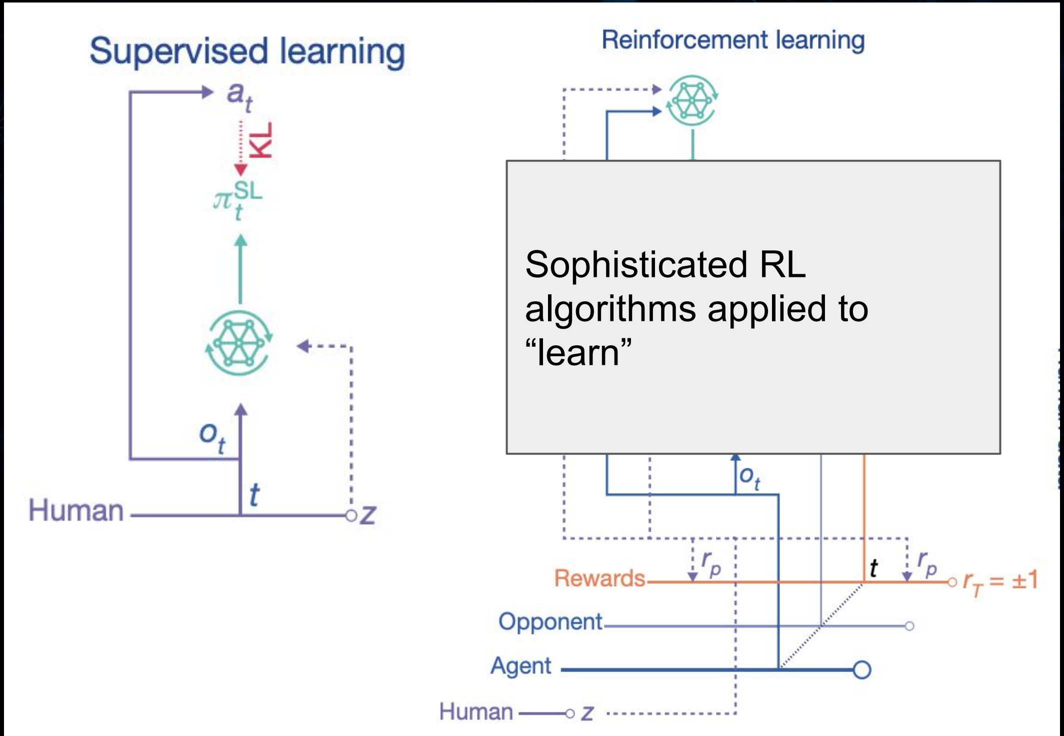
88k Accesses | **573** Citations | **992** Altmetric | Metrics

Abstract

Many real-world applications require artificial agents to compete and coordinate with other agents in complex environments. As a stepping stone to this goal, the domain of StarCraft has emerged as an important challenge for artificial intelligence research, owing to its iconic and enduring status among the most difficult professional esports and its relevance to the real world in terms of its raw complexity and multi-agent challenges. Over the course of a decade and numerous competitions[1,2,3], the strongest agents have simplified important aspects of the game, utilized superhuman capabilities, or employed hand-crafted sub-systems[4]. Despite these advantages, no previous agent has come close to matching the overall skill of top StarCraft players. We chose to address the challenge of StarCraft using general-purpose learning methods

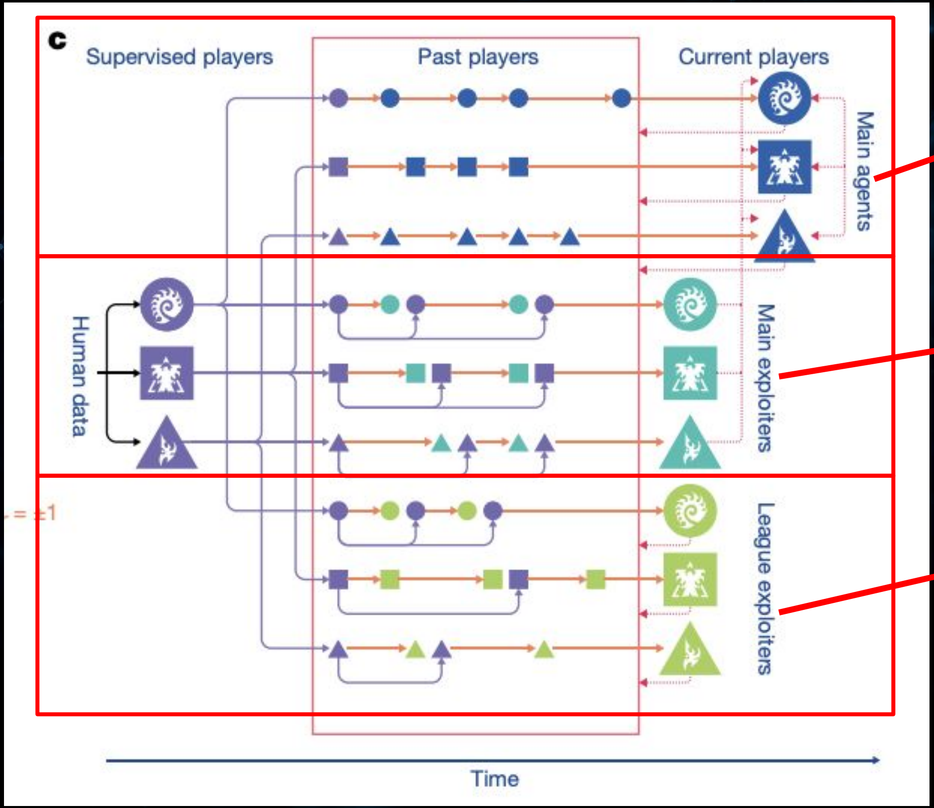
- Starcraft is more complex than board games like Go
    - Go has ~19*19 action spaces
    - Starcraft has huge (thus sparse) action spaces!
        - E.g. image pixel level
        - Or "what, who, where, when" action sequence!
- To make things more challenging!
    - Imperfect info (e.g. no direct visibility to opponent)
    - AI has to respond in ~100ms

AlphaStar 2019, combine supervised and reinforcement learning (remember alphago?)



- Imitate experts by learning human replays, conditioned on z (i.e. build order stats)
- Off-policy on learning from "experience replay" using [a2c] and other algorithms
- "exploration of novel strategy" vs exploitation vs "robustness"
  - more reward toward supervised policy

AlphaStar 2019, multi-agent league training

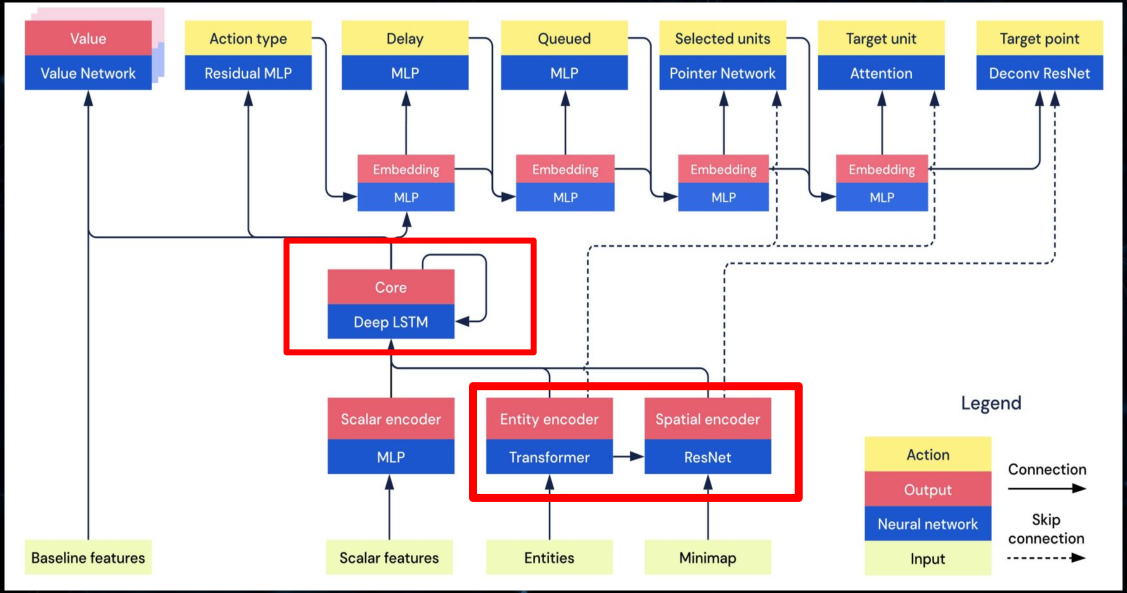


3 groups, 12 agents in total

- Main agents (1 x 3)
  - Self-play with all other 3 groups of agents and try to improve
- Main exploiters (1 x 3)
  - Exploit weakness in “Main” group by playing with Main agents only
- League exploiters (2 x 3)
  - Try to find systematic weakness of the “entire league”

Exploiters periodically re-initialized (purple shapes) to increase diversity and robustness

AlphaStar 2019, network architecture



- Notice the network output is structured
    - What: Action type
    - When: Delay/Queued
    - Who: Selected unit and Target units
    - Where: Target point
- LSTM (RNN) used to process sequence
- ResNet for minimap
- Transformer to apply “self attention” on units/entities

AlphaFold 2021

nature

View all journals | Search | Login

Explore content | About the journal | Publish with us

nature > articles > article

Download PDF

Article | Open Access | Published: 15 July 2021

Highly accurate protein structure prediction with AlphaFold

John Jumper, Richard Evans, ... Demis Hassabis + Show authors

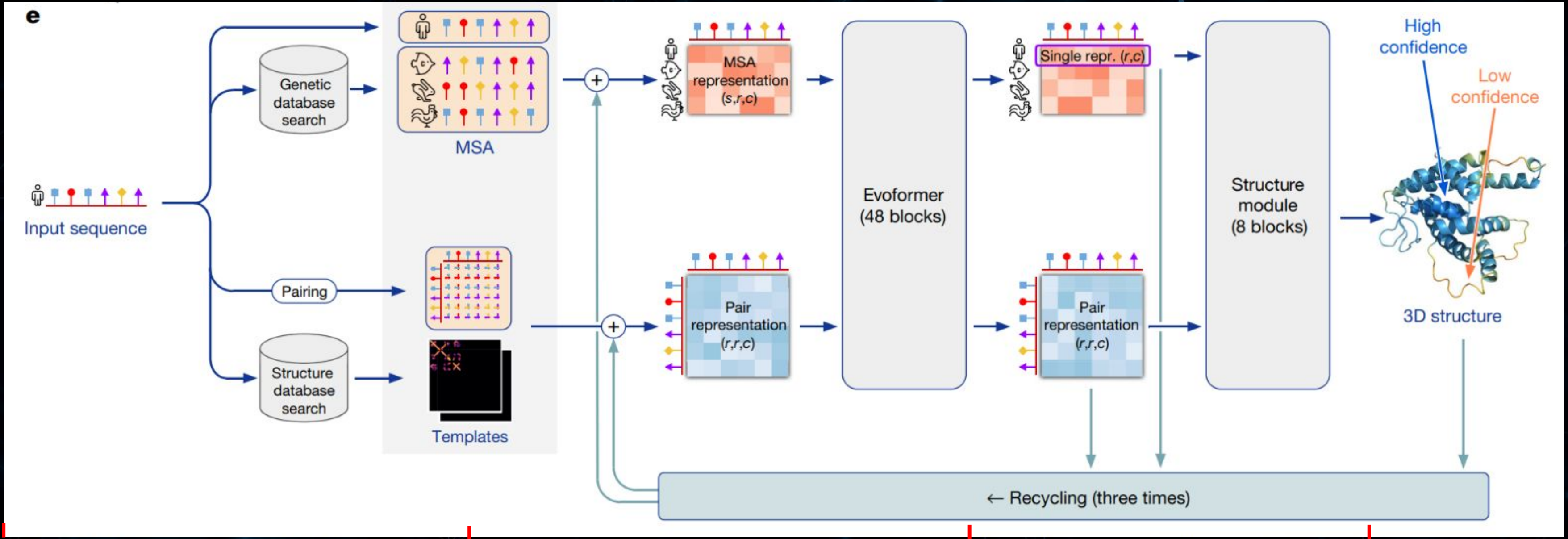
Nature **596**, 583–589 (2021) | Cite this article

Abstract

Proteins are essential to life, and understanding their structure can facilitate a mechanistic understanding of their function. Through an enormous experimental effort[1,2,3,4], the structures of around 100,000 unique proteins have been determined[5], but this represents a small fraction of the billions of known protein sequences[6,7]. Structural coverage is bottlenecked by the months to years of painstaking effort required to determine a single protein structure. Accurate computational approaches are needed to address this gap and to enable large-scale structural bioinformatics. Predicting the three-dimensional structure that a protein will adopt based solely on its amino acid sequence—the structure prediction component of the 'protein folding problem'[8]—has been an important open research problem for more than 50 years[9]. Despite

- Protein folding 3d structure prediction is a 50-year scientific problem!
    - Without AlphaFold, people rely on expensive and tedious work to observe and record every protein shape
    - AlphaFold completely changed the game to get atomic level precision!
- Transformer alike attention model applied

AlphaFold 2021, main model structure

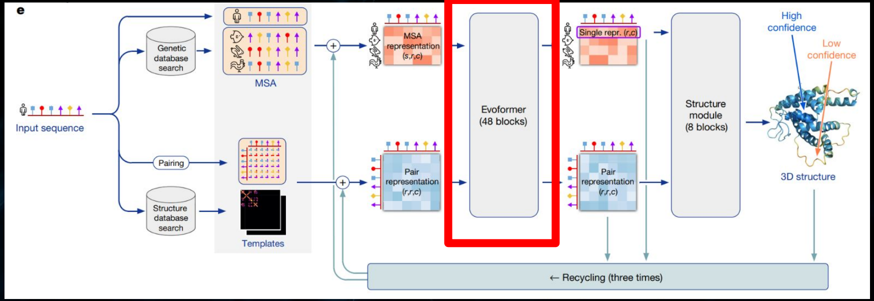


Feature Engineering

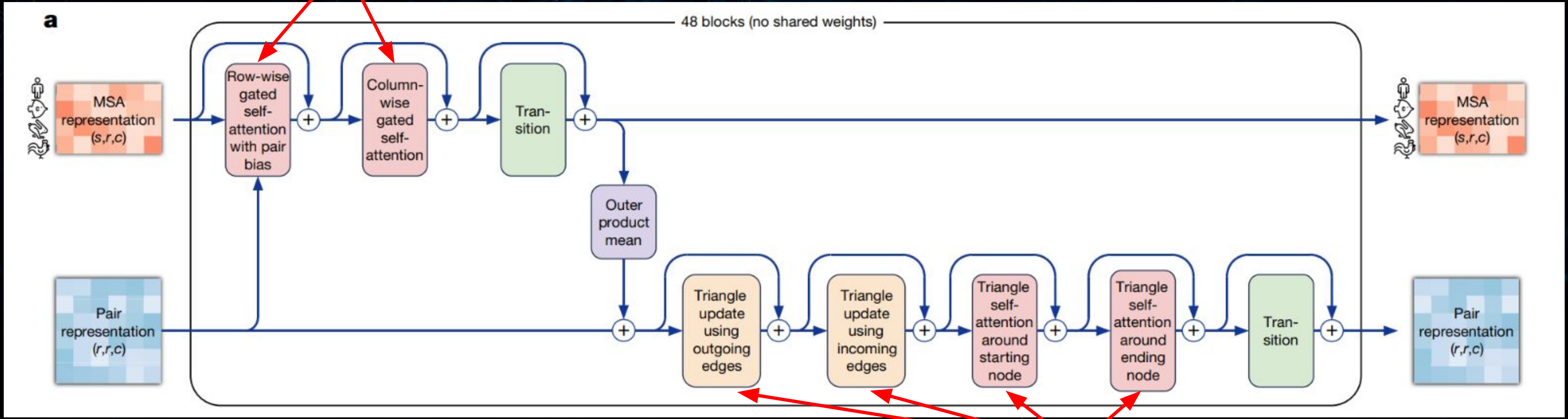
Feature fusion and Encoding

Decoding

AlphaFold 2021, Evoformer



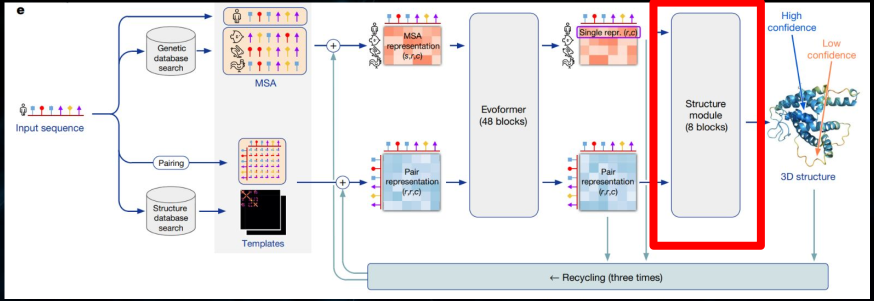
Row and column wise gated attention for 2D data

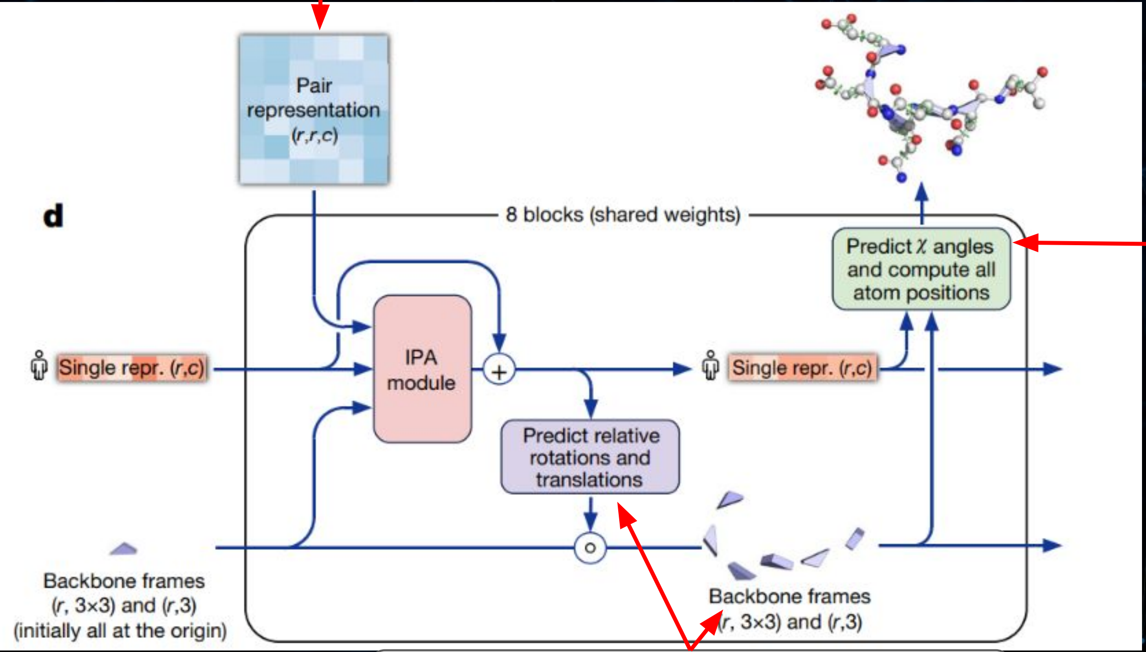


- Triangle Update? Think about triangle as stable structure in 3d space, and d1+d2 > d3
- Directed graph, thus treat outgoing/incoming differently

AlphaFold 2021, Structure Module

Pair rep from Evoformer output

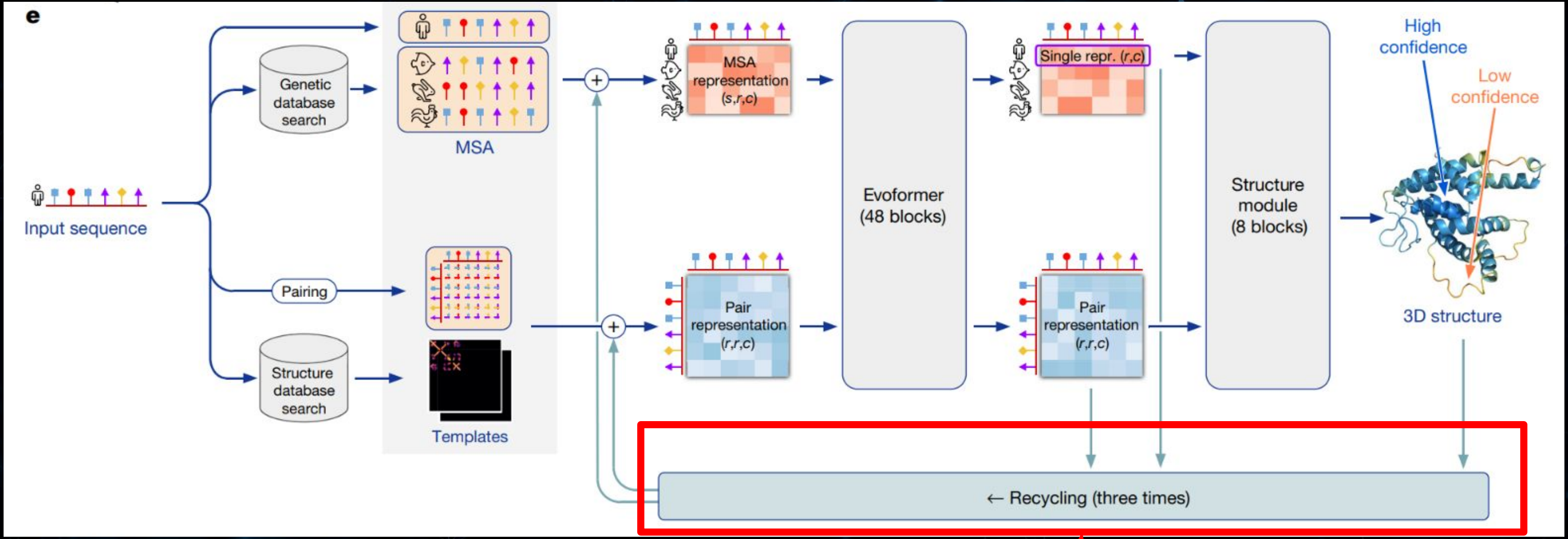




With rotation and translation vectors, build 3D representation

(3x3) vector as rotation, and (3) vector as position translation

AlphaFold 2021, 3x “recycling”



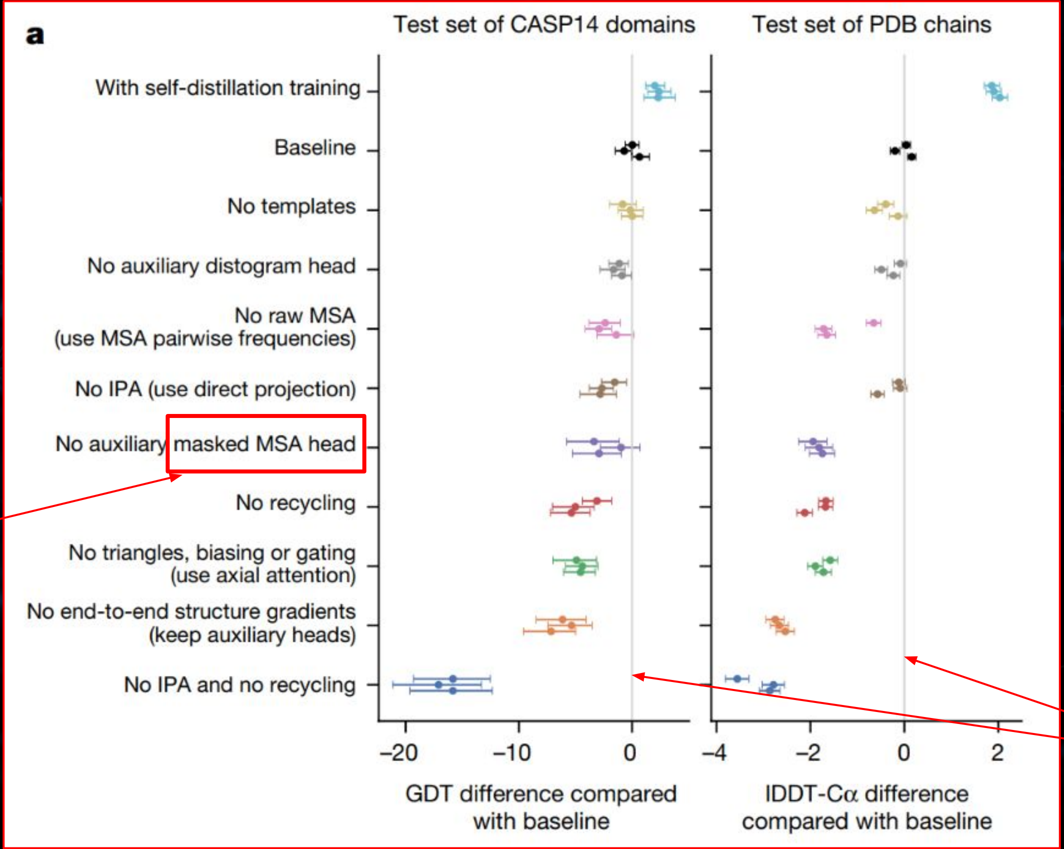
Equivalent to a 3x deep network
with residual connection

AlphaFold 2021, special training tricks

Training with labelled and unlabelled data

The AlphaFold architecture is able to train to high accuracy using only supervised learning on PDB data, but we are able to enhance accuracy (Fig. 4a) using an approach similar to noisy student self-distillation[35]. In this procedure, we use a trained network to predict the structure of around 350,000 diverse sequences from Uniclust30[36] and make a new dataset of predicted structures filtered to a high-confidence subset. We then train the same architecture again from scratch using a mixture of PDB data and this new dataset of predicted structures as the training data, in which the various training data augmentations such as cropping and MSA subsampling make it challenging for the network to recapitulate the previously predicted structures. This self-distillation procedure makes effective use of the unlabelled sequence data and considerably improves the accuracy of the resulting network.

AlphaFold 2021, model/training feature ablation



BERT-alike pretraining is everywhere

baselines

AlphaCode 2022

DeepMind

2022-3-16

Competition-Level Code Generation with AlphaCode

Yujia Li*, David Choi*, Junyoung Chung*, Nate Kushman*, Julian Schrittwieser*, Rémi Leblond*, Tom Eccles*, James Keeling*, Felix Gimeno*, Agustin Dal Lago*, Thomas Hubert*, Peter Choy*, Cyprien de Masson d'Autume*, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu and Oriol Vinyals
*Joint first authors

Programming is a powerful and ubiquitous problem-solving tool. Developing systems that can assist programmers or even generate programs independently could make programming more productive and accessible, yet so far incorporating innovations in AI has proven challenging. Recent large-scale language models have demonstrated an impressive ability to generate code, and are now able to complete simple programming tasks. However, these models still perform poorly when evaluated on more complex, unseen problems that require problem-solving skills beyond simply translating instructions into code. For example, competitive programming problems which require an understanding of algorithms and complex natural language remain extremely challenging. To address this gap, we introduce AlphaCode, a system for code generation that can create novel solutions to these problems that require deeper reasoning. In simulated evaluations on recent programming competitions on the Codeforces platform, AlphaCode achieved on average a ranking of top 54.3% in competitions with more than 5,000 participants. We found that three key components were critical to achieve good and reliable performance: (1) an extensive and clean competitive programming dataset for training and evaluation, (2) large and efficient-to-sample transformer-based architectures, and (3) large-scale model sampling to explore the search space, followed by filtering based on program behavior to a small set of submissions.

1 Introduction 2
2 Problem setup 5
2.1 Competitive programming 5
2.2 Evaluation 6
3 Datasets 6
3.1 Pre-training dataset 7
3.2 CodeContests fine-tuning dataset 7
4 Approach 9
4.1 Model architecture 10
4.2 Pre-training 11
5.4 Results on APPS 19
6 AlphaCode's capabilities & limitations 20
6.1 Copying from training data 21
6.2 Model solution characteristics 22
6.3 Sensitivity to problem descriptions 24
6.4 Sensitivity to provided metadata 24
6.5 Loss is a poor proxy for solve rate 26
7 Related work 27
7.1 Program synthesis 27
7.2 Transformers for program synthesis 28
7.3 Scaling sampling 28

- One of the many powerful LLMs
  - GPT3 (codex), T5, LaMDA, PaLM…
- Focus on the very hard "Competitive coding" problem
  - GPT3 or codex alike AI are more "instruction to code" translation
  - AlphaCode targets to understand a long and difficult problem statement and provide a solution
- Achieved average competitor level

AlphaCode 2022, architecture and training

- Encoder-decoder transformer
  - Natural language seq, to expect code output seq
  - Encoder output can be cached for sampling later
- Pre-training
  - "Masked language modelling loss", similar to BERT, on Github dataset
  - AdamW with weight decay and gradient clipping
  - *Scaling: 2018 OpenAI scaling rule is used, but Chinchilla paper recommends different scaling settings*
- Fine-tuning
  - CodeContext dataset (the format of the problems to be solved)
  - Focus on precision because one problem has multiple solutions!
    - Ignore missed token penalty if not in its distribution
  - Tags added to input text (prompt), e.g. "dp" (dynamic programming), "bfs" (breadth-first search), and rating (e.g. 1200 points)

AlphaCode 2022, sampling, filtering & clustering

- Sampling
  - Millions of samples for each problem! Half in python, half in c++
  - “Randomize problem tags” (e.g. dp, bfs, dfs) “and ratings” (e.g. 1200)
    - As if a human competitor guesses which difficulty level the problem is and what high-level approach to take
  - Set high temperature (less confident, but encourage diversity) and apply sampling decoding
- Filtering
  - Run the generated code and keep those that passed sample tests
    - Still there can be 10k+ solutions after filtering
    - Or, 10% problems have no single solution passed with 1M+ sampling
- Clustering
  - Additional test inputs added, group the samples by additional outputs
    - Thus a separate test-input-generation model is needed!
  - Select one solution from largest to smallest clusters afterward

Some thoughts

- Breakthrough research from Deepmind is built on top of
  - Vision and support from leads
  - Top researchers
  - Engineering skills
  - Scalable ML infra
  - Computation resource budget

- DeepMind research seems to be more "practically useful"
  - Mostly on games before 2019
  - Biology, weather forecast, math intuition, fusion control, large language models, video segmentation and compression since 2020

References

Papers

- AlphaGo (nature)
- AlphaGo Zero (nature)
- AlphaZero (science)
- AlphaStar (science)
- AlphaFold2 (nature)
- AlphaCode (arxiv)

Other references

- Jonathan Hui: AlphaGo: How it works technically?
- Yannic Kilcher - YouTube
- Mu Li - YouTube
- Shusen Wang - YouTube
- Two Minute Papers - YouTube

Thank you!

Please mark attendance at go/iamhere!

fun-ai-talks

Let's read 6 DeepMind Alpha* papers!

2016 AlphaGo	2017 AlphaGo Zero	2018 AlphaZero
2019 AlphaStar	2021 AlphaFold	2022 AlphaCode

hululu.zhu@gmail.com

05/2022

fun-ai-talks

Review 6 DeepMind Alpha* papers

AlphaGo

AlphaGo Zero

AlphaZero

AlphaStar

AlphaFold

AlphaCode

hululu.zhu@gmail.com

05/2022

Thank you!

Please mark attendance at go/iamhere!

fun-ai-talks

Review 6 DeepMind Alpha* papers

- AlphaGo
- AlphaGo Zero
- AlphaZero
- AlphaStar
- AlphaFold
- AlphaCode

hululu.zhu@gmail.com

05/2022