Mastering the game of Go without human knowledge

Michael Wollowski

Summary of

Silver at al. *Mastering the game of Go without human knowledge*
https://www.nature.com/articles/nature24270

Introduction

- AlphaGoZero is its own teacher: a neural network is trained to predict its own move selections and the winner of games.
- The training happens solely on reinforcement learning, without human data, guidance or domain knowledge beyond game rules.
- AlphaGoZero achieved superhuman performance, winning 100–0 against the previously published, champion-defeating AlphaGo.

Introduction

- Until recently, supervised learning systems were trained to replicate the decisions of human experts.
- However, expert data sets are often expensive, unreliable or simply unavailable.
- Even when reliable data sets are available, they may impose a ceiling on the performance of systems trained in this manner.

Introduction

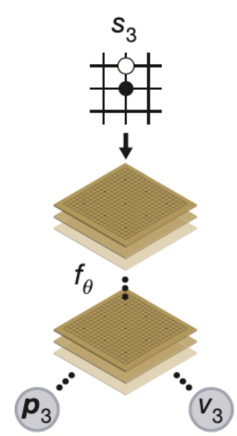
- By contrast, reinforcement learning systems are trained from their own experience.
- Recently, there has been rapid progress towards this goal, using deep neural networks trained by reinforcement learning.
- The game of Go was widely viewed as a grand challenge for artificial intelligence.
- It requires a precise and sophisticated look-ahead in vast search spaces.

Basic Characteristics

- AlphaGoZero differs from AlphaGo Fan and AlphaGo Lee in several important aspects.
  1. It is trained solely by self-play reinforcement learning, starting from random play, without any supervision or use of human data.
  2. It uses only the black and white stones from the board as input features.
  3. It uses a simpler tree search, without performing any Monte Carlo rollouts.

Overall Setup of AlphaGoZero

- It uses a deep neural network.
- Input is the raw board position and its history.
- The network outputs both, move probabilities and a value.
- The vector of move probabilities ***p*** represents the probability of selecting each move.
- The value *v* is a scalar evaluation, estimating the probability of the current player winning from position *s*.
- It only uses its deep neural network to evaluate leaf nodes and to select moves.

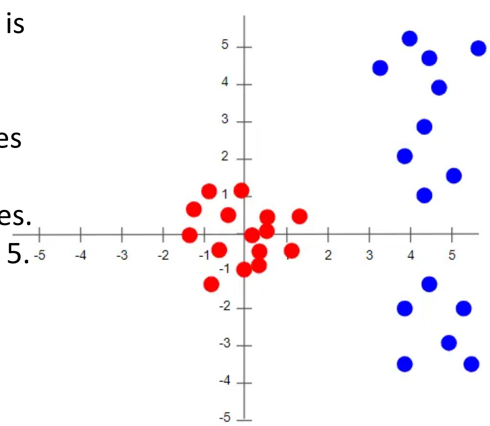


Overall Setup of AlphaGoZero

- It uses many shiny residual blocks of convolutions layers.
  - In traditional neural networks, each layer feeds into the next layer.
  - In a network with residual blocks, each layer feeds into the next layer and directly into the layers about 2–3 hops away.

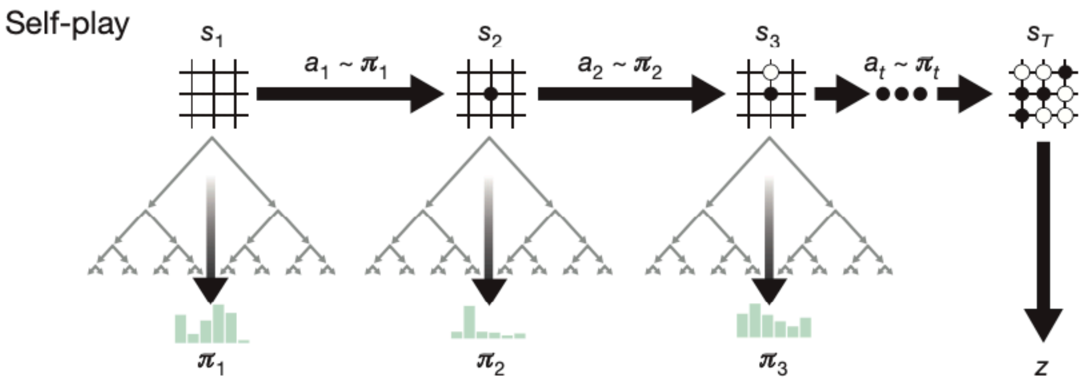
Overall Setup of AlphaGoZero
Batch Normalization

- When inputting data to a deep learning model, it is standard practice to normalize the data to zero mean and unit variance.
- Suppose the input data consists of several features x1, x2,…xn.
- Each feature might have a different range of values.
- Values for feature x1 might range from 1 through 5.
- Values for feature x2 might range from 1000 to 99999.
- The original values (in blue) are now centered around zero (in red).



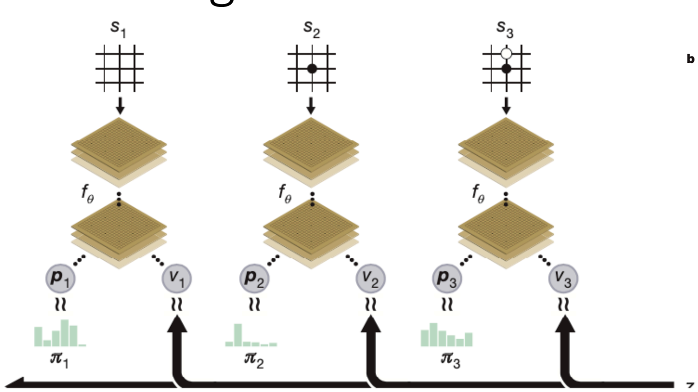
Source: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

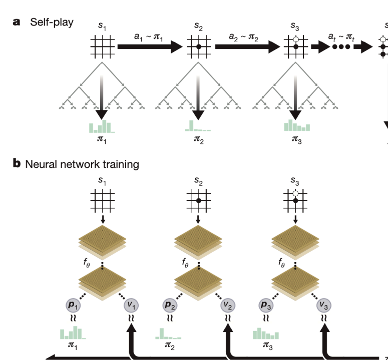
Self-Play



- The program plays a game, s_1, … s_T against itself.
- In each position s_t, a MCTS is executed, resulting in a probabilities $\boldsymbol{\pi}$ of playing each move in s_t.
- The final position s_T is scored to determine the winner, *z*.

NN Training



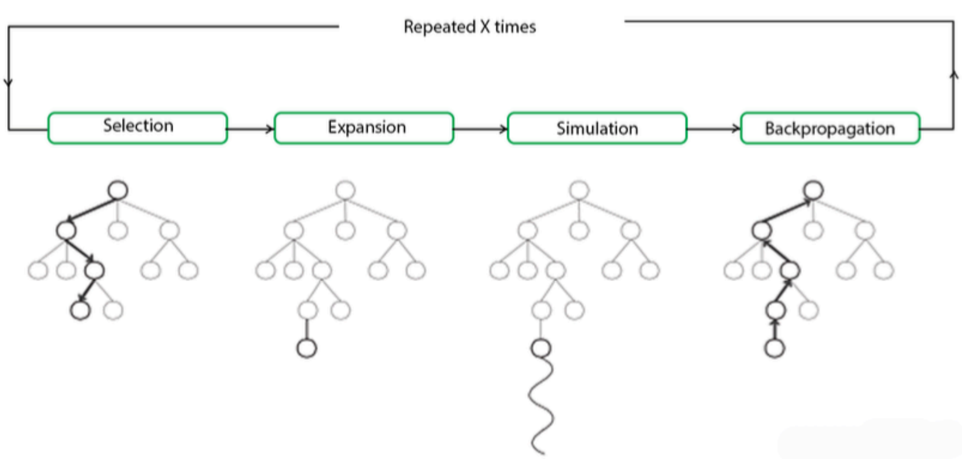


- The neural network takes the raw board position s_t as its input and passes it through many convolutional layers.
- It outputs a vector $\boldsymbol{p}_t$, representing a probability distribution over moves, and
- a scalar value v_t, representing the probability of the current player winning in position s_t.

NN Training

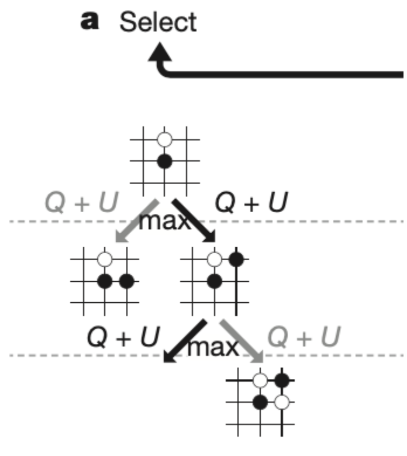
- The neural network parameters are updated:
  - to maximize the similarity of the probability vector $\boldsymbol{p}_t$ to the search probabilities $\boldsymbol{\pi}_t$, and
  - to minimize the error between the predicted winner v_t and the game winner z.
- The new parameters are used in the next iteration of self-play.

MCTS Reminder



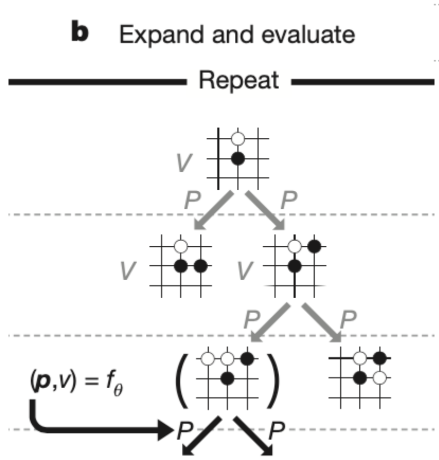
MCTS in AlphaGoZero – Selection

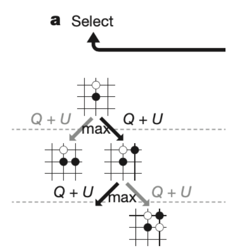
- Each simulation traverses the tree by selecting the edge with maximum action value Q.
- Each edge (*s*, *a*), i.e. (*situation*, *action*) in the search tree stores a prior probability *P*(*s*, *a*), a visit count *N*(*s*, *a*), and an action value *Q*(*s*, *a*).



MCTS in AlphaGoZero – Expansion and Evaluation

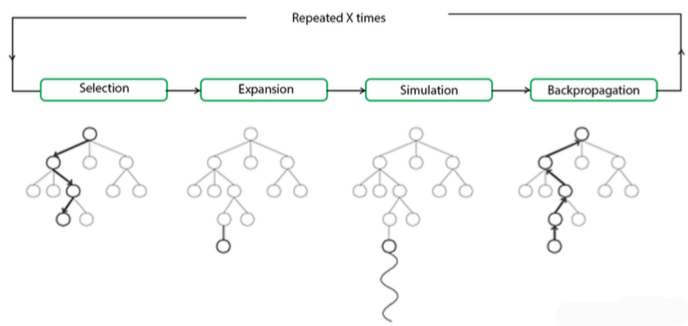
- The leaf node is expanded and the associated position s is evaluated by the neural network





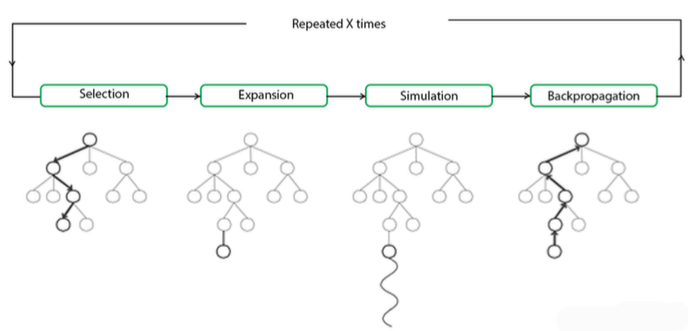
MCTS in AlphaGoZero – Expansion and Evaluation

- Unlike regular MCTS, AlphaGoZero, does not use the simulation step.
- Instead, it uses the NN to evaluate the expanded step.
- Eventually the learned knowledge of game play is used to evaluate the current situation.



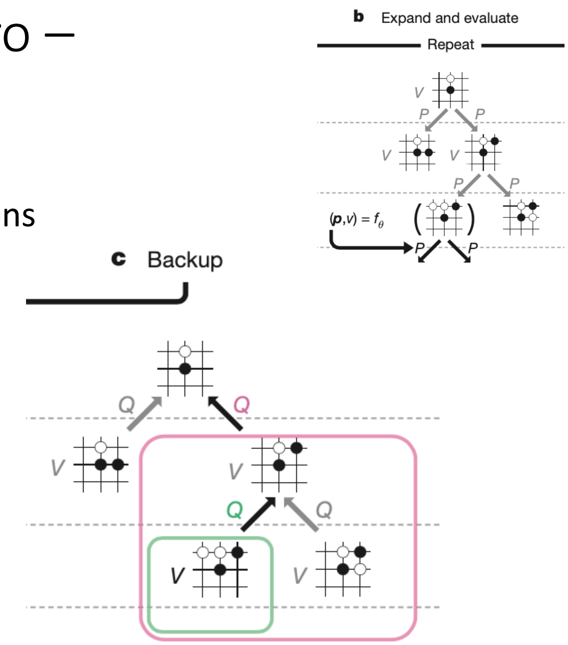
MCTS in AlphaGoZero – Expansion and Evaluation

- This is what an expert Go player would do.
- There are no rules.
- This is all pattern recognition.
- The nearest pattern will be used.
- The network definitely does not store all patterns; it couldn't



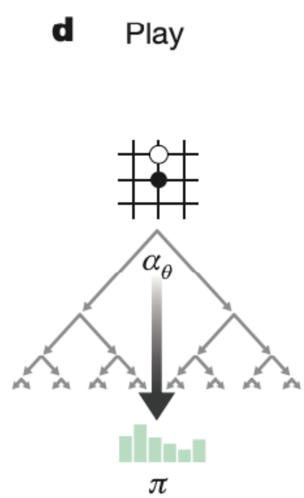
MCTS in AlphaGoZero – Backup

- Action value Q is updated to track the mean of all evaluations V in the subtree below that action.



MCTS in AlphaGoZero – Play

- Once the search is complete, search probabilities π are returned, proportional to N 1/τ, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.



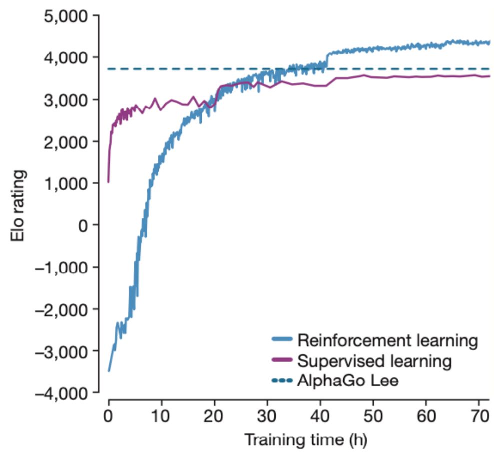
Evaluation

- Training started from completely random behavior.
- It continued without human intervention for approximately three days.
- 4.9 million games of self-play were generated
- Using 1,600 simulations for each MCTS, corresponds to approximately 0.4 s thinking time per move.

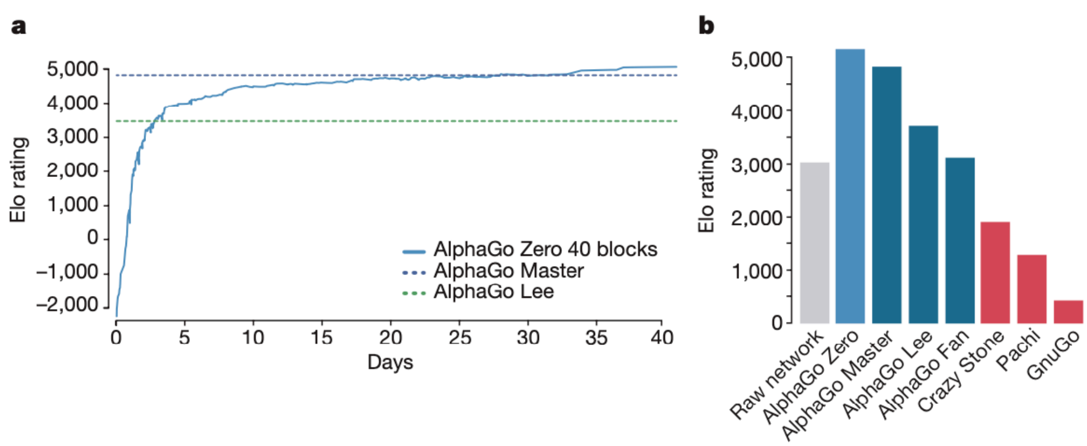
Go Knowledge provided to AlphaGoZero

- The player is provided with the set of legal moves in each position.
- Games terminate when both players pass or after 19 × 19 × 2 = 722 moves.
- AlphaGoZero uses Tromp–Taylor scoring during MCTS simulations and self-1play training.

Evaluation: Elo Rating



Performance of AlphaGoZero



Knowledge Learned by AlphaGoZero

- AlphaGoZero discovered a remarkable level of Go knowledge during its self-play training process.
- This included fundamental elements of human Go knowledge
- As well as nonstandard strategies beyond the scope of traditional Go knowledge.

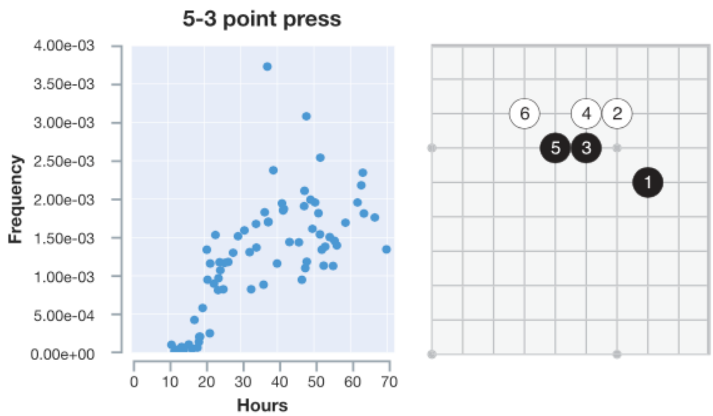
Knowledge Learned by AlphaGoZero

It rapidly progressed from entirely random moves towards a sophisticated understanding of Go concepts, including:

- *fuseki* (opening),
- *tesuji* (tactics),
- life and death, *ko* (repeated board situations),
- *yose* (endgame),
- capturing races, *sente* (initiative),
- shape, influence and territory, all discovered from first principles.
- *shicho* (‘ladder’ capture sequences that may span the whole board)—one of the first elements of Go knowledge learned by humans—were only understood much later in training.

Pattern Recognition

- The network learned common joseki patterns:



Pattern Recognition

- It additionally learned new joseki patterns: