Mastering the game of GO without human knwoledge

Dimitar Davikdov

INTRODUCTION AND PREVIOUS WORK

What is the game of GO

- The game was invented in China more than ơ, əə years ago and is believed to be the oldest board game continuously played to the present day
- Two players take turns placing stones on the vacant intersections (*points*) of a Ơ *Ơ board
- The aim is to surround more territory than the opponent, without letting your opponent surround your stones
- Easy to play hard to master
- Number of legal positions is ơ.ə Ọ Ơ ˀ ᵉ ơþŒəᴼ ə

Previous Work - AlphaGo Fan

- Named after the European champion Fan Hui, which it defeated in October σəǪ
- It utilized two deep neural networks: a policy network that outputs move probabilities, and a value network that outputs a position evaluation
- The policy network was initially trained by supervised learning to predict human expert moves and was improved by policy-gradient reinforcement learning
- The value network was trained to predict the winner of games played by the policy network against itself
- Once trained, these networks were combined with a Monte-Carlo Tree Search (MCTS)

Previous Work - AlphaGo Lee

- Named after Lee Sedol, the winner of Ơ international titles, in March ơəỢ
- Uses a very similar approach as the AlphaGo Fan with minor improvements
- Can be considered as the first algorithm to achieve super human level of GO mastery

ALPHA GO ZERO

Alpha Go Zero

It is a new algorithm having several changes compared to previous iterations:

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 a single neural network, rather than separate policy and value networks
3. 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

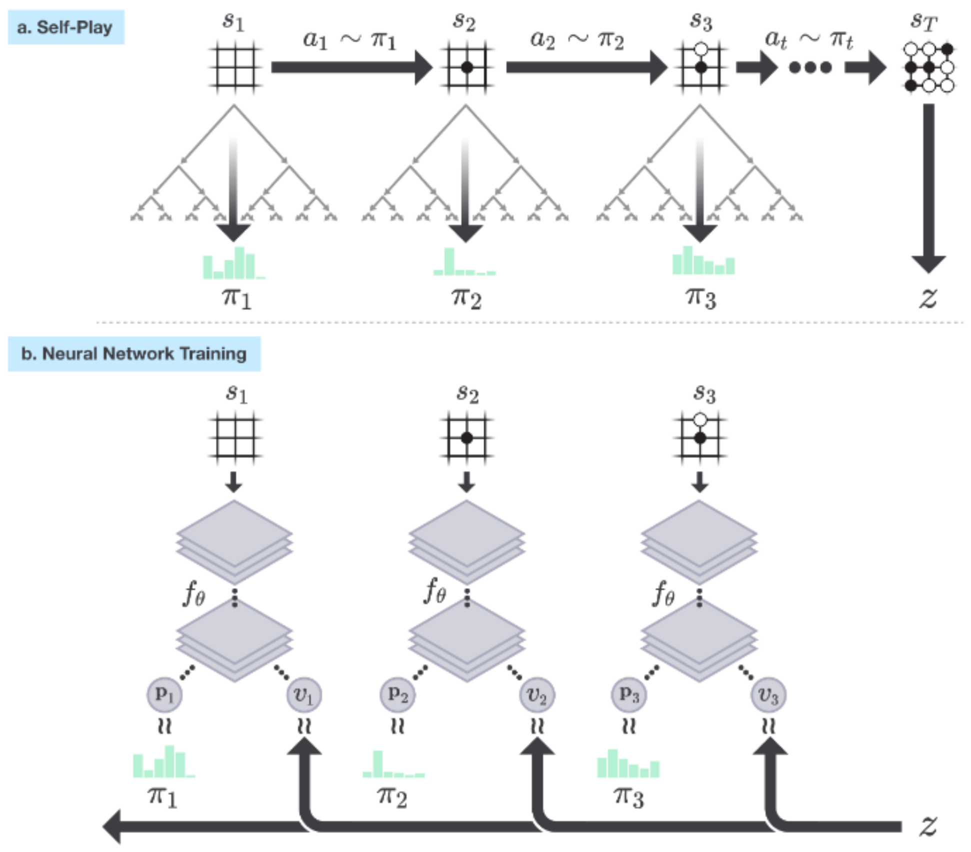
The Neural Network

- The neural network f_o takes as an input the board state s and outputs both move probabilities and a value: $(p,v) = f_o(s)$
- The vector of move probabilities p represents the probability of selecting each move (including pass): $p_a = Pr(a|s)$
- 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 non-linearities

Self play reinforcement learning

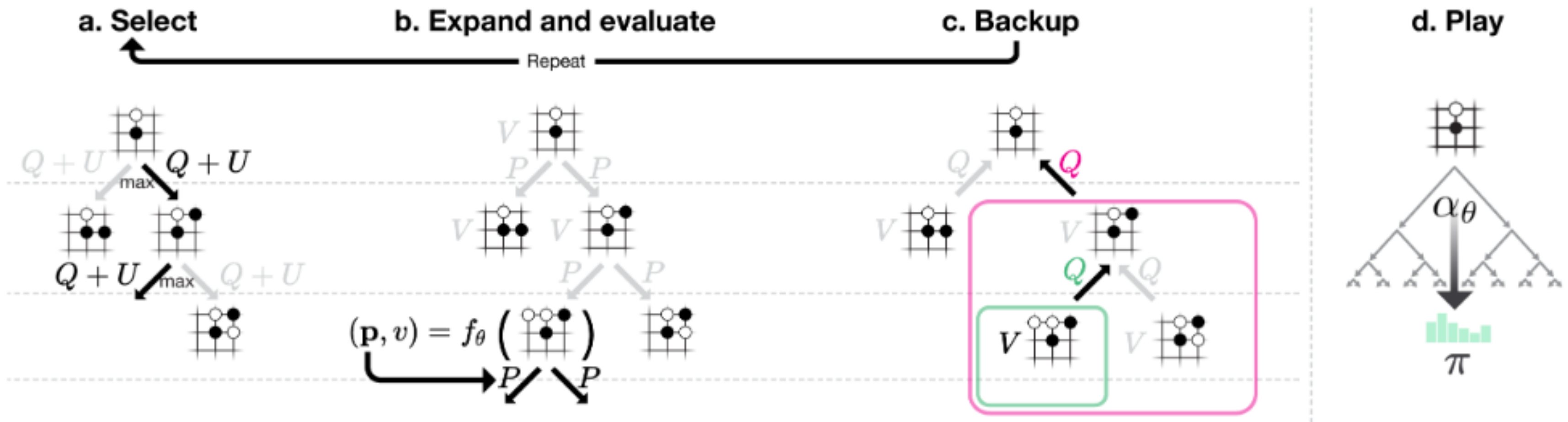
a) The program plays a game $s_0, \dots, s_T$ against itself. Moves are selected by the MCTS and the terminal position s_T is scored to compute the game winner z

b) Neural network training in AlphaGo Zero. The neural network takes the position s_t and outputs probability vector for the next move and a scalar for the win probability



The MCTS

- The Monte-Carlo tree search uses the neural network f_θ to guide its simulations
- Each edge (s,a) in the search tree stores a prior probability P(s,a), a visit count N(s,a), and an action-value Q(s,a)
- The tree iteratively selects nodes for which Q(s,a)+U(s,a) is the largest, where $U(s,a) \propto P(s,a)/(1+N(s,a))$ until it reaches a leaf node s_L
- Then the direct children of s_L are expanded and the network generates both prior probabilities and evaluation: $(P(s_L,\cdot),V(s_L)) = f_\theta(s_L)$
- After that all traversed edges have their visit count (N) incremented and value (Q) updated
- Finally a probability vector π is generated proportional to $N(s,a)^{1/\tau}$, where τ is parameter controlling the temperature



a) Edges are traversed based on Q(s,a)+U(s,a)
b) The leaf node is expanded and the associated positions is evaluated by the neural network(P(s,ф,V(s)) = f_o(s)
c) Action-values Q are updated to track the mean of all evaluations V in the subtree below that action
d) Once the search is complete, search probabilities ! are returned, proportional to $N^{\sigma !}$

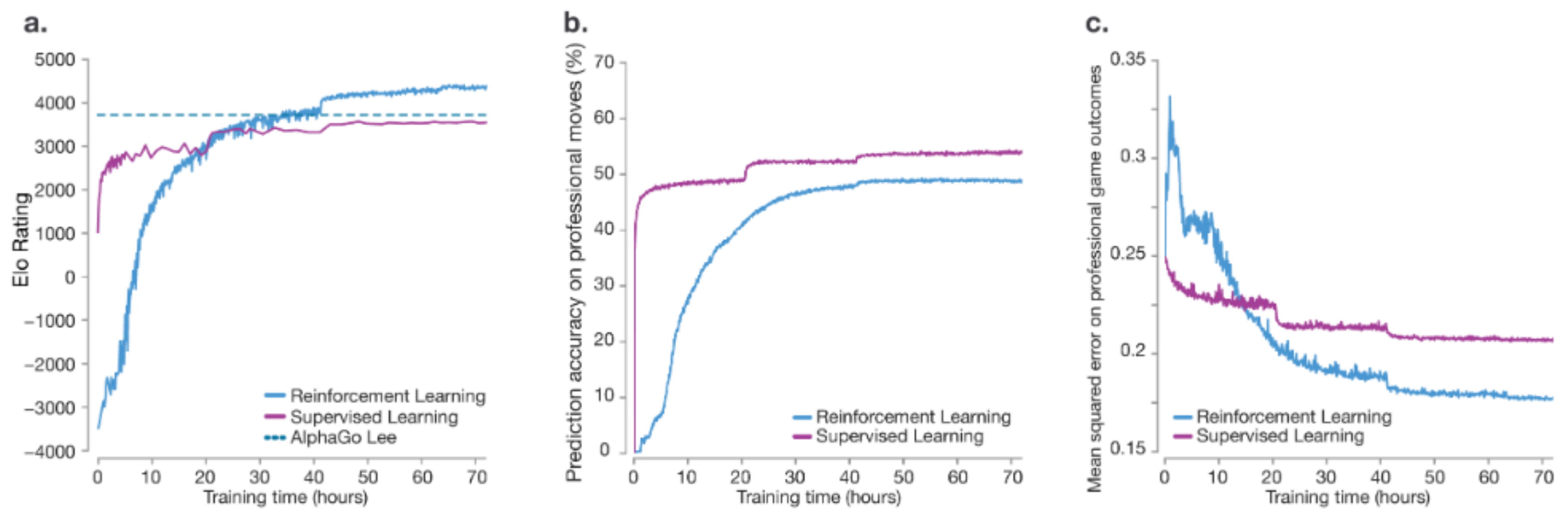
The RL pipeline

- First, the neural network is initialized to random weights $o_{ə}$
- At each time-step t, an MCTS search $!_t = ķ_{oiÞơ}(s_t)$ is executed using the previous iteration of neural network $f_{oiÞơ}$
- The game terminates at step T and a reward of $r_T \in \{Þơ,+ơ\}$ based on the winner
- The data for each time-step t is stored as $(s_t, !_t, z_t)$ where $z_t = \pm r_T$ is the game winner
- The neural network $(p,v) = f_{oi}(s)$ is adjusted to minimize the error between v and z, and to maximize the similarity between p and !
- The parameters o are adjusted based on a GD of a loss function $l = (zÞv)^{ơ} Þ !^{T} \log p + c l l o l l^{ơ}$

EMPIRICAL ANALYSIS OF ALPHAGO ZERO

The training process

- Training started from completely random behavior and continued without human intervention for approximately ̹ days
- ʿ ꜝ. million games of self-play were generated, using Q̦ əə simulations for each MCTS
- This corresponds to approximately ə̔. s thinking time per move
- Then AlphaGo Zero was evaluated against AlphaGo Lee and defeated it Œ́ə to ə
- Additional comparisons were made with a supervised learning algorithm using the same neural network and an expert moves dataset

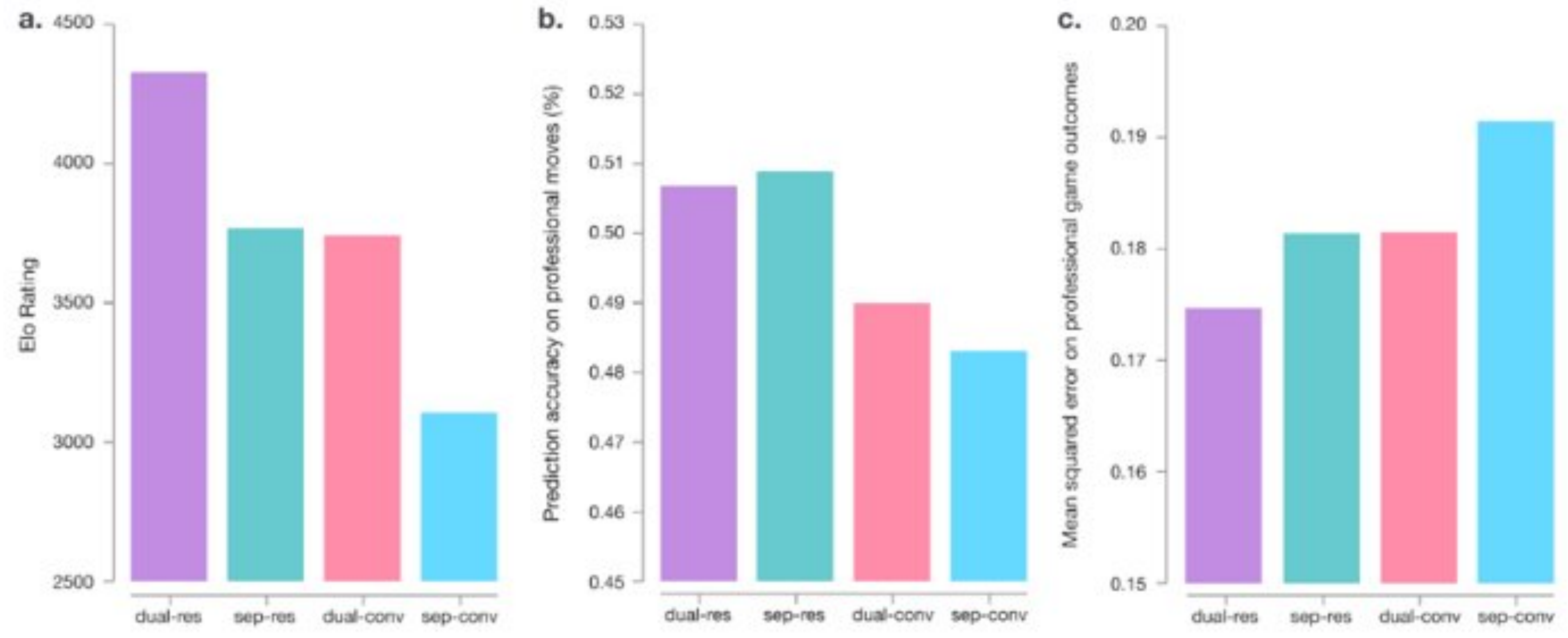


a) Performance of self-play reinforcement learning
b) Prediction accuracy on human professional moves
c) Mean-squared error(MSE) on human professional game outcomes

Additional evaluation

To separate the contributions of architecture and algorithm, four variations of the AlphaGo architectures were compared:

- Algorithms
  - Using separate policy and value networks, as in AlphaGo Lee
  - Using combined policy and value networks, as in AlphaGo Zero
- Architectures:
  - Using the convolutional network architecture, as in AlphaGo Lee
  - Using the residual network architecture, as in AlphaGo Zero

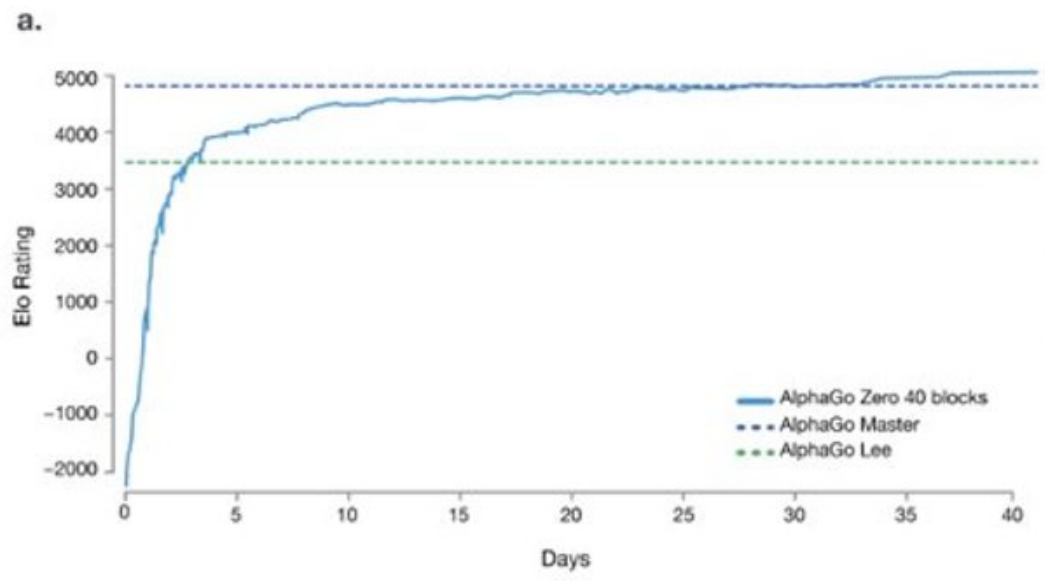


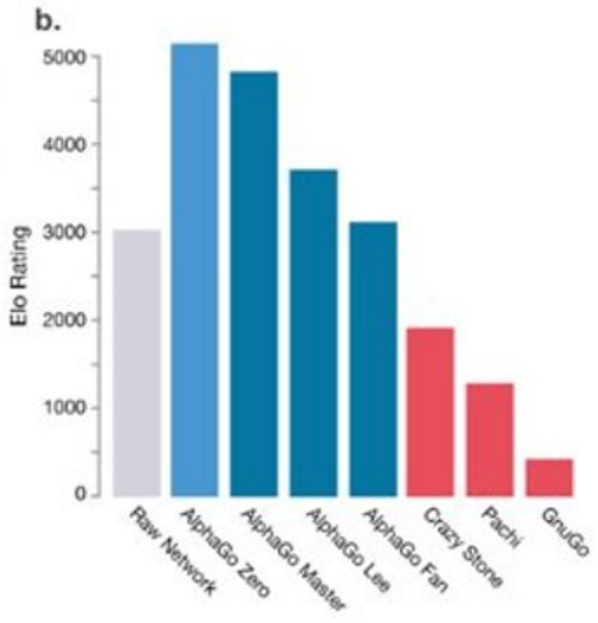
a) Elo ratings of the individual algorithms
b) Prediction accuracy on human professional moves
c) Mean-squared error (MSE) on human professional game outcomes

FINAL VERSION AND CONCLUSION

Final version

- A second instance of AlphaGo Zero was trained from random behavior forˁ ə days
- Over the course of training, ơ̇ million games of self-play were generated
- The fully trained AlphaGo Zero was evaluated using an internal tournament against AlphaGo Fan, AlphaGo Lee, and several previous Go programs
- AlphaGo Master was also included in the tournament œa program based on the algorithm and architecture presented in the paper but utilizing human data and features
- AlphaGo Master defeated the strongest human professional players̡ əœə in online games in January ơəƠ
- Finally, AlphaGo Zero played head to head against AlphaGo Master in a Œ́ə game match. AlphaGo Zero won byᵉ ⁱ games to ŒƠ





a) Learning curve for AlphaGo Zero using largerʿ ə block residual network overʿ ə days
b) Final performance of AlphaGo Zero

Conclusions

- AlphaGo Zero discovered a remarkable level of Go knowledge during its self-play training process including discovering novel Go tactics
- The results comprehensively demonstrate that a pure reinforcement learning approach is fully feasible, even in the most challenging of domains
- Furthermore, a pure reinforcement learning approach requires just a few more hours to train, and achieves much better performance, compared to training on human expert data
- In principle this approach should be applicable to other games with perfect information

THANK YOU AND Q&A