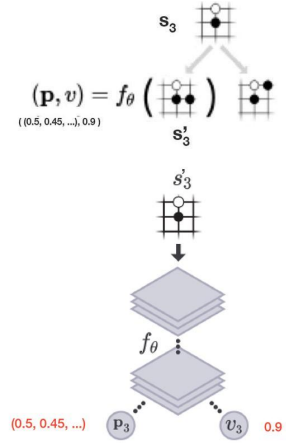
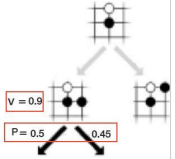
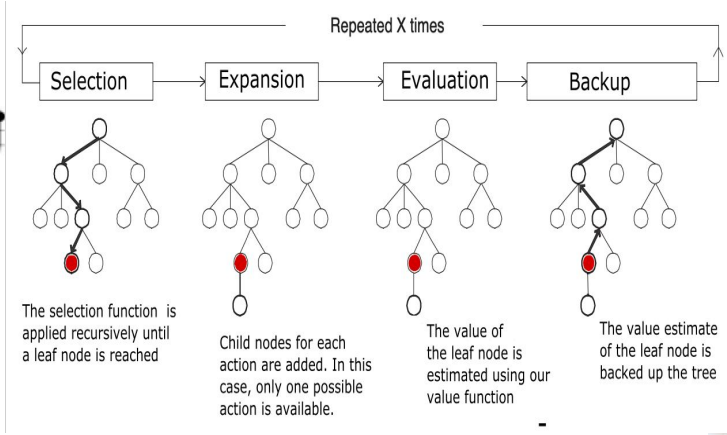
Monte Carlo Tree Search
&
Connect 4

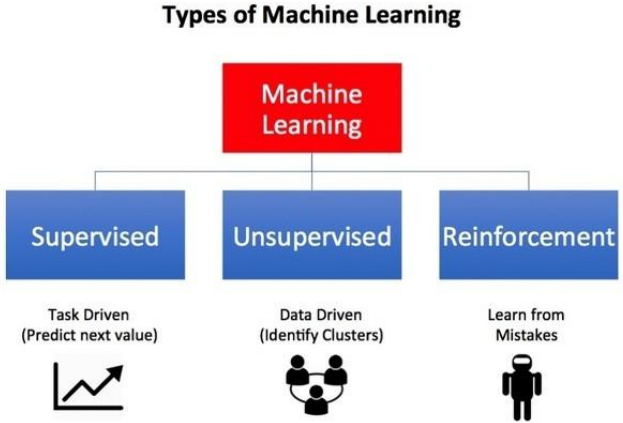
Andreas Olsson & Frans Sjöström



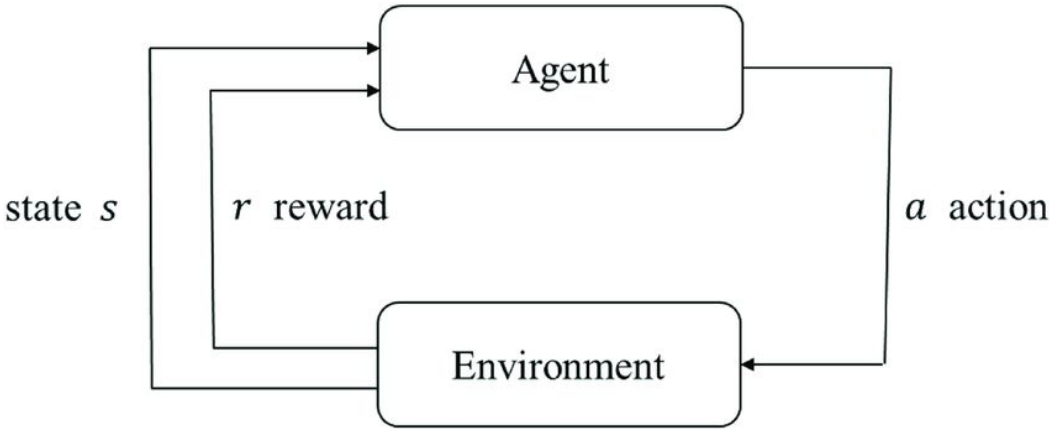




Reinforcement learning

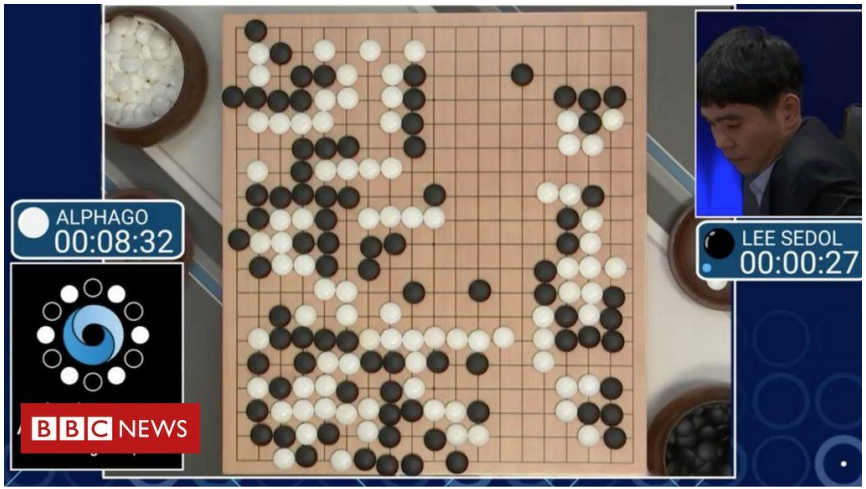


- Train an agent through trial and error
- In every state *s*, pick an action *a*
- Signal *r* is communicated back to agent
- Goal to maximize *r*
- Deep Blue beats current chess champion in 1997



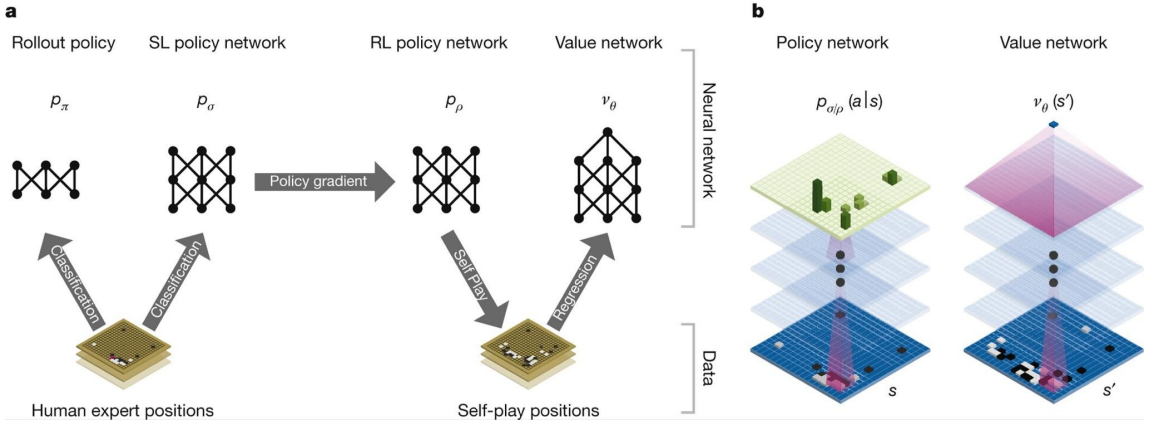
AlphaGo, AlphaGo Zero, AlphaZero

- Action space too large for “conventional” algorithms (minimax, etc)
- For chess, *b = 35, d = 80,* for Go, *b = 250, d = 150* b^d



AlphaGo

- ❏ Developed by DeepMind at Google
- ❏ Agent designed to play the game of Go
- ❏ Two networks, policy and value
- ❏ Pre-trained on expert positions
- ❏ Later trained on self-play
- ❏ One (multiple smaller) more iteration to follow

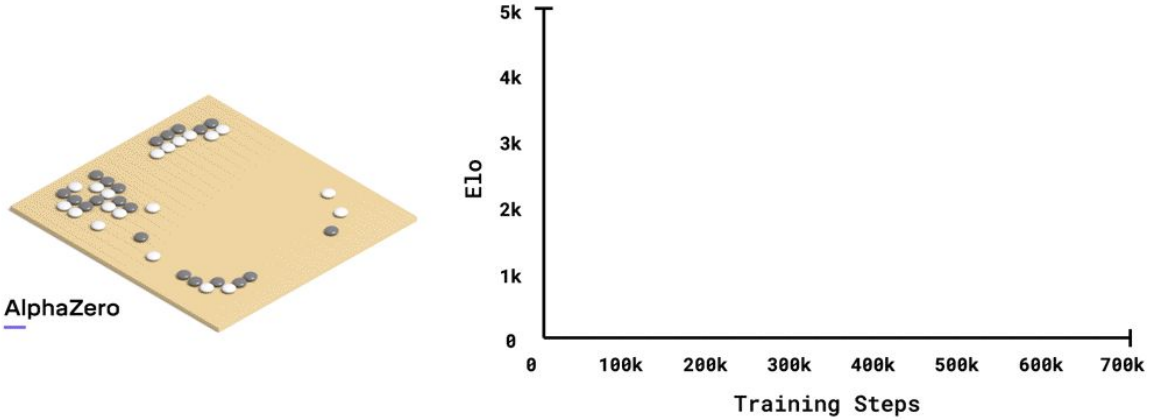


AlphaGo Zero

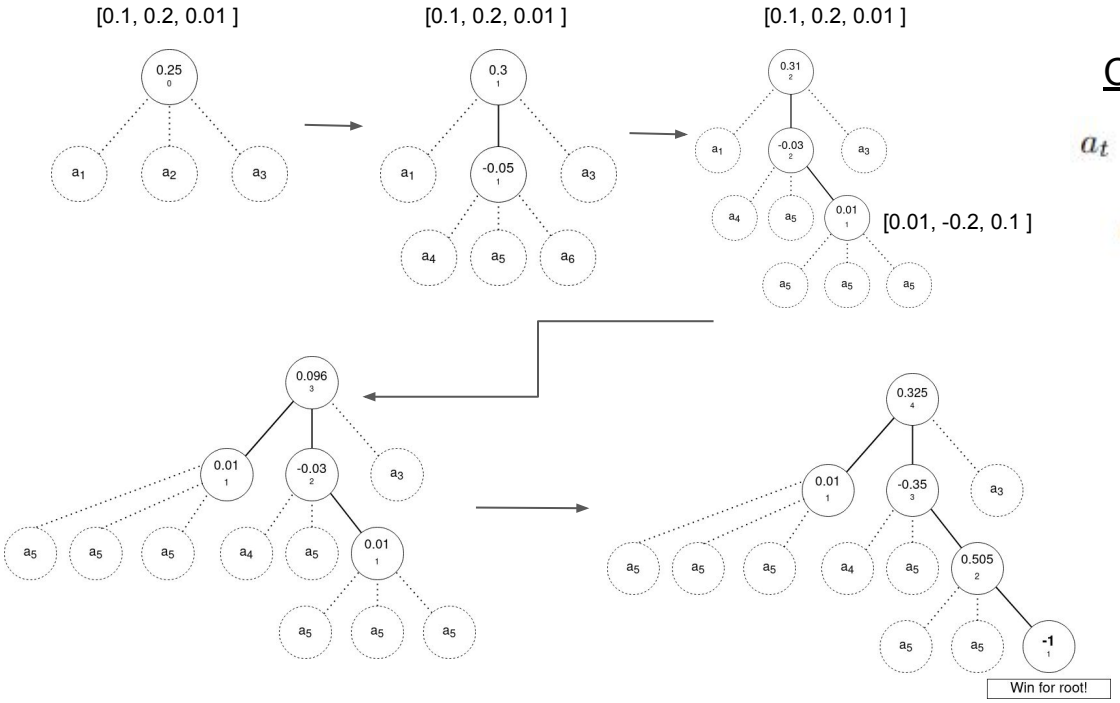
- ❏ Builds upon AlphaGo, some modifications
- ❏ Learns from scratch, no processing of human expert moves
- ❏ Increased computational requirements

AlphaZero

- Generalized version of AlphaGo Zero
- Can play atari games, such as shogi, etc
- Chess is one example of another game it can master
- Image below shows AlphaZero vs “state of the art” agents, the previously best known agent



Monte Carlo Tree Search (MCTS)



Choice:

$$a_t = argmax_a(Q(s_t, a) + U(s_t, a))$$

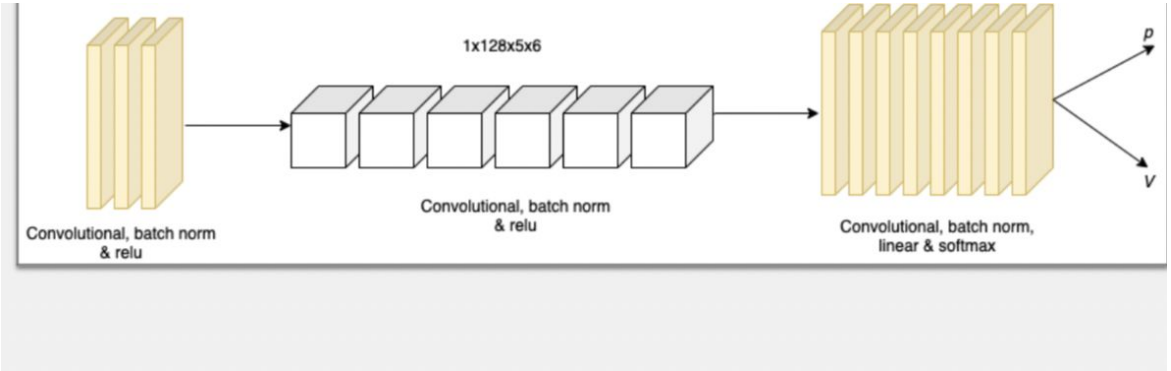
$$U(s,a) = C_{puct} \cdot P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$

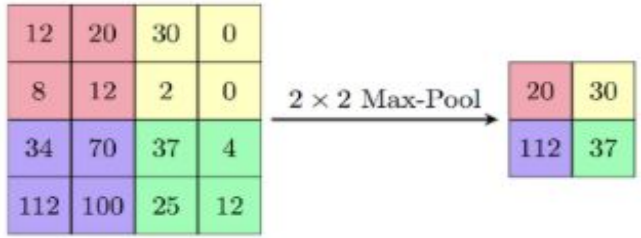
Result: Trainingdata
(state, policy)

Difference from AlphaGo: No simulation step

Neural network

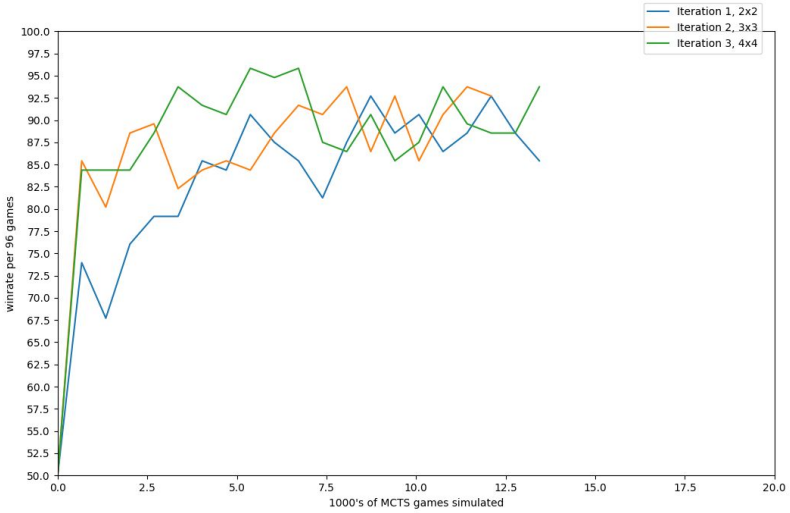
- Neural network consists of three input layers (player 1 board, player 2 board, what player to make next move) and two output layers. In between, there are a number of residual blocks
- A total of 6 residual blocks, reduced from 19 in the initial implementation
- Output is policy ***p*** and value **v** for state *s*,



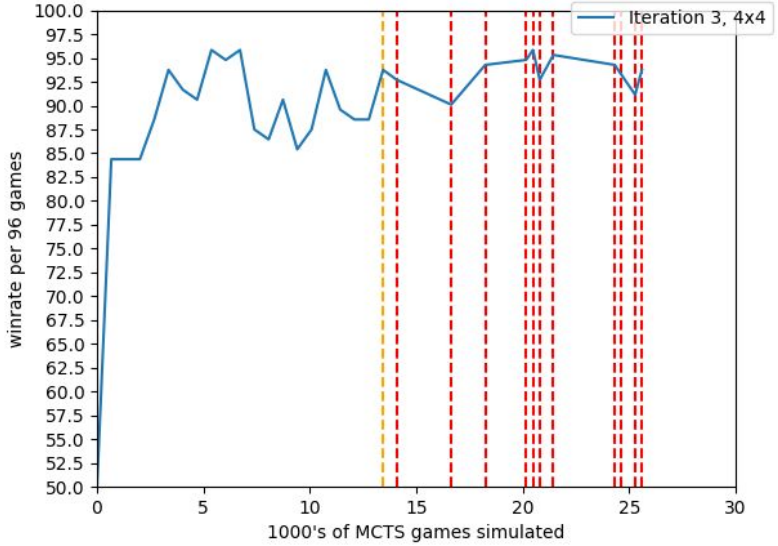


Experiments

- Trained on self-play generated games, network vs itself
- The experiments consisted of
  - First experiment was done by generating 672 games, training the network, evaluating the network versus random, continuing to generate 672 games, ….
  - Second experiment was done by building on iteration 3 that had a total of 13440 games trained on it. Here, new iterations were generated and trained by utilizing the previous iteration to generate games and then train on them. It did so until the new iteration beat the previous iteration by 80%. Then, it played against random and returned a result.



Results



Algorithm 1 Generate games and train network

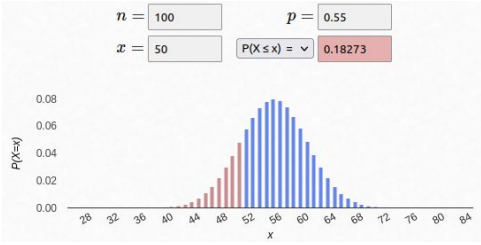
```
procedure TRAIN_NETWORK
    x ← number of self-play iterations to run
    h ← number of self-play games per iteration
    v ← number of evaluations against random
    i ← 0
    for i ← 0 to x do
        simulate h number of MCTS games
        train network on all previous games
        evaluate against random v times
```

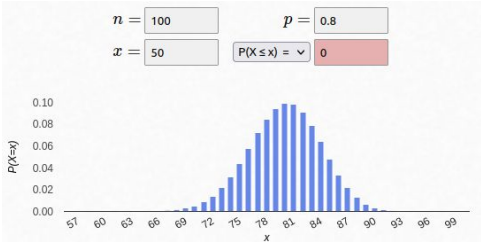
Algorithm 3 Generate self-play and compare networks

```
procedure SELF-PLAY AND EVALUATION
    iteration ← Network to start from
    total_iterations ← iteration to stop at
    num_games ← number of self-play games to play
    winner ← winner of evaluation
    for i ← iteration to total_iterations do
        create empty network of iteration i + 1
        generate num_games number of self-play games with iteration i
        train iteration i + 1 on games generated by iteration i
        winner ← Evaluate iteration i + 1 versus iteration i
        while winner ≠ i + 1 do
            generate num_games number of self-play games with iteration i
            train iteration i+1 on games generated by iteration i
            winner ← Evaluate iteration i + 1 versus iteration i
        evaluate iteration i + 1 versus random
```

Evaluation

- ❏ Actual improvements for new iteration
- ❏ More training with old data



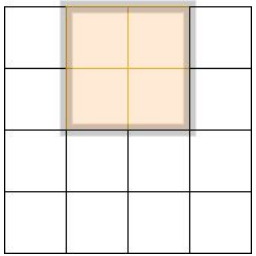


Reflection

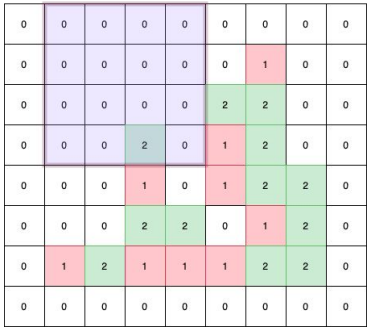
- ❏ Reinforcement learning will improve your model still, even if you are using bad data in the beginning
- ❏ Need for high computing power
- ❏ Machine learning uses a lot of parallel processing

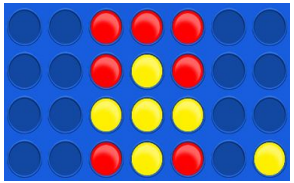
Further improvements

- ❏ Generality for doing Monte Carlo Simulations
- ❏ Mirroring board to increase training data
- ❏ Generalizing the rules of the game
- ❏ Keeping the 4x4 kernel size for connect4
- ❏ Make it more parallel to increase speed

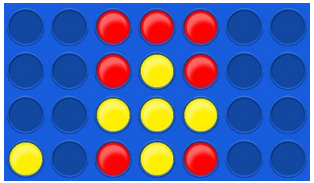


vs





+



Thanks for listening

LUND
UNIVERSITY