How AlphaGo Works

Yuu Sakaguchi
How to Play Go

Played on a 19 x 19 square grid board.

Black and white stones.

Points awarded for surrounding empty space.
Why is Go Hard to Compute?
Why is Go Hard to Compute?

Search space is huge

After the first two moves of a Chess game, there are 400 possible next moves. In Go, there are close to 130,000.

Complexity : $250^{150}$ possible sequences
Match against Lee Sedol

AlphaGo played professional Go player Lee Sedol, ranked 9-dan, one of the best players at Go in March 2016. AlphaGo won by 4 - 1.
How did AlphaGo solve it?
How did AlphaGo solve it?

Ideas

- Deep Learning
- Convolutional Neural Network
- Supervised Learning
- Reinforcement Learning
- Monte-Carlo Tree Search
How did AlphaGo solve it?

Strategies

Knowledge learned from human expert games and self-play.

Monte-Carlo search guided by policy and value networks.
Computing Go

AlphaGo sees the board as One-hot matrix.

Give a state $s$, pick the best action $a$. 
Computing Go
The hidden layers of a CNN consist of convolutional layers, pooling layers, fully connected layers and normalization layers. There are many applications such as image and video recognition, recommender systems and natural language processing.
Types of Neural Networks

1. Policy Network

Breath Reduction. Finds the probability of the next move, and reduces the action candidates.

2. Value Network

Depth Reduction. Evaluates the value of the board at each state.
## Types of Neural Networks

<table>
<thead>
<tr>
<th>Name</th>
<th>Network</th>
<th>Data Set</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_\pi P_\zeta$</td>
<td>Linear Softmax</td>
<td>8M from expert players</td>
<td>CPU 2μs</td>
</tr>
<tr>
<td>$P_\sigma P_\rho$</td>
<td>Deep Network</td>
<td>28M from expert players</td>
<td>GPU 2ms</td>
</tr>
<tr>
<td>$V_\theta$</td>
<td>Deep Network</td>
<td>30M random states from $P_\sigma$ + 160M probabilities from $P_\rho$</td>
<td>GPU 2ms</td>
</tr>
</tbody>
</table>
Types of Neural Networks

Policy Network

- Input layer: \(19 \times 19 \times 48\)
- Hidden layers: \(19 \times 19 \times k \times (12 \text{ layers})\)
- Output layer: \(19 \times 19 \ P(a|s)\)

Value Network

- Input layer: \(19 \times 19 \times 49\)
- Hidden layer: \(19 \times 19 \times 192 \times (12 \text{ layers}) + 19 \times 19 \times (1 \text{ layer}) + 256 \times (1 \text{ layer})\)
- Output layer: 1 output \(V(S)\)
### Types of Networks

#### Extended Data Table 2 | Input features for neural networks

<table>
<thead>
<tr>
<th>Feature</th>
<th># of planes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stone colour</td>
<td>3</td>
<td>Player stone / opponent stone / empty</td>
</tr>
<tr>
<td>Ones</td>
<td>1</td>
<td>A constant plane filled with 1</td>
</tr>
<tr>
<td>Turns since</td>
<td>8</td>
<td>How many turns since a move was played</td>
</tr>
<tr>
<td>Liberties</td>
<td>8</td>
<td>Number of liberties (empty adjacent points)</td>
</tr>
<tr>
<td>Capture size</td>
<td>8</td>
<td>How many opponent stones would be captured</td>
</tr>
<tr>
<td>Self-atari size</td>
<td>8</td>
<td>How many of own stones would be captured</td>
</tr>
<tr>
<td>Liberties after move</td>
<td>8</td>
<td>Number of liberties after this move is played</td>
</tr>
<tr>
<td>Ladder capture</td>
<td>1</td>
<td>Whether a move at this point is a successful ladder capture</td>
</tr>
<tr>
<td>Ladder escape</td>
<td>1</td>
<td>Whether a move at this point is a successful ladder escape</td>
</tr>
<tr>
<td>Sensibleness</td>
<td>1</td>
<td>Whether a move is legal and does not fill its own eyes</td>
</tr>
<tr>
<td>Zeros</td>
<td>1</td>
<td>A constant plane filled with 0</td>
</tr>
<tr>
<td>Player color</td>
<td>1</td>
<td>Whether current player is black</td>
</tr>
</tbody>
</table>

*Feature planes used by the policy network (all but last feature) and value network (all features).*
Types of Networks

Policy Network

Input - First hidden layer:
- 2x2 padding
- 5x5 convolutional by 5 filters
- ReLU function

n - n+1 hidden layer
- 21x21 padding
- 3x3 convolutional by 3 filters
- ReLU function

12th hidden layer - Output
- 1 output
- Different biases on each place on board
- Softmax function
Types of Networks

Value Network

Input - 12th hidden layer:
Same as policy network.

12th - 13th hidden layer
● 1x1 filter
● ReLU function

13th - 14th hidden layer
● Fully connected
● ReLU function

14th - output
● Fully connected
● tanh function
Training

Supervised learning of policy network

4 weeks on 50 GPUs using Google Cloud.

57% accuracy on test data.
Training

Reinforcement learning of policy network
1 week on 50 GPUs using Google Cloud.
80% against supervised learning.
Training

Supervised learning of value network

1 week on 50 GPUs using Google Cloud.
Monte-Carlo Tree Search
Monte-Carlo Tree Search
Monte-Carlo Tree Search: selection

P: prior probability
Q: action value

\( u(P) = \frac{P}{N} \)
Monte-Carlo Tree Search: expansion

$P_\sigma = \text{policy network}$

$P = \text{prior probability}$
Monte-Carlo Tree Search: evaluation

$V_\theta = \text{value network}$
Monte-Carlo Tree Search : rollout

$V_\theta = \text{value network}$

$r = \text{game score}$
Monte-Carlo Tree Search: backup

\[ Q = \text{action value} \]

\[ V_\theta = \text{value network} \]

\[ r = \text{game score} \]
DeepMind - Beyond AlphaGo
Questions?