Go in numbers

3,000 Years Old

40M Players

$10^{170}$ Positions
Why is Go hard for computers to play?

Game tree complexity = $b^d$

Brute force search intractable:

1. Search space is huge
2. “Impossible” for computers to evaluate who is winning
Convolutional neural network
Value network

Evaluation

Position

\[ v_\theta(s) \]
Policy network

Move probabilities

Position

\( p_{\sigma}(a|s) \)

\( \sigma \)

\( s \)
Exhaustive search
Monte-Carlo rollouts
Reducing depth with value network
Reducing depth with value network
Reducing breadth with policy network
Neural network training pipeline

- Human expert positions
- Supervised Learning policy network
- Reinforcement Learning policy network
- Self-play data
- Value network

Steps:
1. Human expert positions → Classification → Supervised Learning policy network
2. Supervised Learning policy network → Self Play → Reinforcement Learning policy network
3. Reinforcement Learning policy network → Self Play → Self-play data
4. Self-play data → Regression → Value network
Supervised learning of policy networks

**Policy network:** 12 layer convolutional neural network

**Training data:** 30M positions from human expert games (KGS 5+ dan)

**Training algorithm:** maximise likelihood by stochastic gradient descent

\[
\Delta \sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}
\]

**Training time:** 4 weeks on 50 GPUs using Google Cloud

**Results:** 57% accuracy on held out test data (state-of-the-art was 44%)
Reinforcement learning of policy networks

**Policy network:** 12 layer convolutional neural network

**Training data:** games of self-play between policy network

**Training algorithm:** maximise wins $z$ by policy gradient reinforcement learning

\[ \Delta \sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma} z \]

**Training time:** 1 week on 50 GPUs using Google Cloud

**Results:** 80% vs supervised learning. Raw network $\sim$3 amateur dan.
Reinforcement learning of value networks

**Value network:** 12 layer convolutional neural network

**Training data:** 30 million games of self-play

**Training algorithm:** minimise MSE by stochastic gradient descent

\[ \Delta \theta \propto \frac{\partial v_\theta(s)}{\partial \theta} (z - v_\theta(s)) \]

**Training time:** 1 week on 50 GPUs using Google Cloud

**Results:** First strong position evaluation function - previously thought impossible
Monte-Carlo tree search in AlphaGo: selection

\[ Q + u(P) \overset{\text{max}}{\rightarrow} Q + u(P) \]

\[ Q + u(P) \overset{\text{max}}{\rightarrow} Q + u(P) \]

\[ P \quad \text{prior probability} \]

\[ Q \quad \text{action value} \]
Monte-Carlo tree search in AlphaGo: **expansion**
Monte-Carlo tree search in AlphaGo: evaluation

\[ v_\theta(\text{state}) \quad \text{Value network} \]
Monte-Carlo tree search in AlphaGo: rollout

$v_\theta$  Value network
$r$    Game scorer
Monte-Carlo tree search in AlphaGo: backup

\[ Q \quad \text{Action value} \]
\[ v_\theta \quad \text{Value network} \]
\[ r \quad \text{Game scorer} \]
At last — a computer program that can beat a champion Go player PAGE 404

ALL SYSTEMS GO
Evaluating Nature AlphaGo against computers

494/495 against computer opponents

>75% winning rate with 4 stone handicap

Even stronger using distributed machines
Evaluating Nature AlphaGo against humans

**Fan Hui** (2p): European Champion 2013 - 2016

Match was played in October 2015

AlphaGo won the match 5-0

First program ever to beat a professional on a full size 19x19 in an even game
Seoul AlphaGo: Improvements

- Improved value network
- Improved policy network
- Improved search
- Improved hardware (TPU vs GPU)
Evaluating Seoul AlphaGo against computers

Beats Nature AlphaGo with 3 to 4 stones handicap

CAUTION: ratings based on self-play results

Beats Nature with 3 to 4 stones handicap
AlphaGo vs Lee Sedol: Game 1
AlphaGo vs Lee Sedol: Game 2
AlphaGo vs Lee Sedol: Game 3
AlphaGo vs Lee Sedol: Game 4
AlphaGo vs Lee Sedol: Game 5
<table>
<thead>
<tr>
<th>Deep Blue</th>
<th>AlphaGo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Handcrafted chess knowledge</td>
<td>Knowledge learned from expert games and self-play</td>
</tr>
<tr>
<td>Alpha-beta search guided by heuristic evaluation function</td>
<td>Monte-Carlo search guided by policy and value networks</td>
</tr>
<tr>
<td>200 million positions / second</td>
<td>60,000 positions / second</td>
</tr>
</tbody>
</table>
What’s Next?
With thanks to: Lucas Baker, David Szepesvari, Malcolm Reynolds, Ziyu Wang, Nando De Freitas, Mike Johnson, Ilya Sutskever, Jeff Dean, Mike Marty, Sanjay Ghemawat.