AlphaZero (A general reinforcement learning algorithm that masters chess, shogi and Go through self-play)

Vojtěch Cahlík

Faculty of Information Technology, Czech Technical University in Prague

cahlivoj@fit.cvut.cz

March 2, 2021
Structure

1. Introduction
2. Background
3. Adversarial search
4. AlphaZero internals
5. Training
6. Evaluation
7. Discussion
AlphaZero is a computer program created by DeepMind that can play chess, shogi and Go

- Reached state-of-the-art performance in all of these games (upon publication in 2018)
- Learned to play entirely from self-play using no domain knowledge except the game rules

Original paper: A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play

- First released by David Silver et al. as preprint in 2017, published in Science the next year

History

- March 2016: AlphaGo won in a match of Go against world champion Lee Sedol
- October 2017: AlphaGo Zero published
  - Learned entirely from self-play
- December 2017: AlphaZero preprint released
These games are:
- Two-player
- Turn-taking
- Perfect information
  - Fully observable state (no hidden information)
- Zero-sum
  - No "win-win" outcome

At the end of the game, the final position is scored (-1 for a loss, 0 for a draw, +1 for a win)
A player typically wins by checkmating the opponent: the opponent’s king is in check and there is no way to remove it from attack.

Draw can happen in several ways, for example:
- A player has no legal moves and is not in check
- The same position occurs three times
- No capture has been made and no pawn has been moved in the previous 50 moves

It is often believed that the optimal solution to chess ("perfect game") is a draw.

Figure: A position in chess. (https://en.wikipedia.org/wiki/Chess/media/File:Scholars_mate_animation.gif)
Modern chess software

- In 1997, a computer (IBM Deep Blue) defeated a human world chess champion (Garry Kasparov) for the first time.
- Current state-of-the-art chess software significantly outperforms the best human players.

Figure: 1997 match between Deep Blue and Garry Kasparov.
Modern chess-playing software

Tournament results (TCEC) [edit]

<table>
<thead>
<tr>
<th>Season</th>
<th>Date</th>
<th>Winner</th>
<th>ver</th>
<th>Runner-Up</th>
<th>ver</th>
<th>Superfinal score</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCEC Season 1</td>
<td>Dec 2010 – Feb 2011</td>
<td>Houdini</td>
<td>1.5a</td>
<td>Rybka</td>
<td>4.0</td>
<td>+ 12 – 23 – 5</td>
</tr>
<tr>
<td>TCEC Season 2</td>
<td>Feb – Apr 2011</td>
<td>Houdini</td>
<td>1.5a</td>
<td>Rybka</td>
<td>4.1</td>
<td>+ 9 – 26 – 5</td>
</tr>
<tr>
<td>TCEC Season 3</td>
<td>Apr – May 2011</td>
<td>N/A (season not completed)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TCEC Season 4</td>
<td>Jan – May 2013</td>
<td>Houdini</td>
<td>3</td>
<td>Stockfish</td>
<td>250413</td>
<td>+ 6 – 38 – 4</td>
</tr>
<tr>
<td>TCEC Season 5</td>
<td>Aug – Dec 2013</td>
<td>Komodo</td>
<td>1142</td>
<td>Stockfish</td>
<td>191113</td>
<td>+ 10 – 30 – 8</td>
</tr>
<tr>
<td>TCEC Season 6</td>
<td>Feb – May 2014</td>
<td>Stockfish</td>
<td>170514</td>
<td>Komodo</td>
<td>7x</td>
<td>+ 13 – 45 – 6</td>
</tr>
<tr>
<td>TCEC Season 7</td>
<td>Sep – Dec 2014</td>
<td>Komodo</td>
<td>1333</td>
<td>Stockfish</td>
<td>141214</td>
<td>+ 7 – 53 – 4</td>
</tr>
<tr>
<td>TCEC Season 8</td>
<td>Aug – Nov 2015</td>
<td>Komodo</td>
<td>9.3x</td>
<td>Stockfish</td>
<td>021115</td>
<td>+ 9 – 89 – 2</td>
</tr>
<tr>
<td>TCEC Season 9</td>
<td>May – Dec 2016</td>
<td>Stockfish</td>
<td>8</td>
<td>Houdini</td>
<td>5</td>
<td>+ 17 – 75 – 8</td>
</tr>
<tr>
<td>TCEC Season 10</td>
<td>Oct – Dec 2017</td>
<td>Houdini</td>
<td>6.03</td>
<td>Komodo</td>
<td>1970.00</td>
<td>+ 15 – 76 – 9</td>
</tr>
<tr>
<td>TCEC Season 11</td>
<td>Jan – Apr 2018</td>
<td>Stockfish</td>
<td>260318</td>
<td>Houdini</td>
<td>6.03</td>
<td>+ 20 – 78 – 2</td>
</tr>
<tr>
<td>TCEC Season 12</td>
<td>Apr – Jul 2018</td>
<td>Stockfish</td>
<td>180614</td>
<td>Komodo</td>
<td>12.1.1</td>
<td>+ 29 – 62 – 9</td>
</tr>
<tr>
<td>TCEC Season 13</td>
<td>Aug – Nov 2018</td>
<td>Stockfish</td>
<td>18102108</td>
<td>Komodo</td>
<td>2155.00</td>
<td>+ 16 – 78 – 6</td>
</tr>
<tr>
<td>TCEC Season 14</td>
<td>Nov 2018 – Feb 2019</td>
<td>Stockfish</td>
<td>190203</td>
<td>LCZero</td>
<td>v0.20.2-32930</td>
<td>+ 10 – 81 – 9</td>
</tr>
<tr>
<td>TCEC Season 15</td>
<td>Mar – May 2019</td>
<td>LCZero</td>
<td>v0.21.1-n74t0.78.610</td>
<td>Stockfish</td>
<td>19050918</td>
<td>+ 14 – 79 – 7</td>
</tr>
<tr>
<td>TCEC Season 16</td>
<td>Jul – Oct 2019</td>
<td>Stockfish</td>
<td>19092522</td>
<td>AlloStain</td>
<td>v0.5-dav_7b41f8c-n11</td>
<td>+ 14 – 81 – 5</td>
</tr>
<tr>
<td>TCEC Season 17</td>
<td>Jan – Apr 2020</td>
<td>LCZero</td>
<td>v0.24-sv-160-3010</td>
<td>Stockfish</td>
<td>20200407</td>
<td>+ 17 – 71 – 12</td>
</tr>
<tr>
<td>TCEC Season 18</td>
<td>May 2020 – Jul 2020</td>
<td>Stockfish</td>
<td>202006170741</td>
<td>LCZero</td>
<td>v0.25.1-svjo-t60-3972-mlh</td>
<td>+ 23 – 61 – 16</td>
</tr>
<tr>
<td>TCEC Season 19</td>
<td>Aug 2020 – Oct 2020</td>
<td>Stockfish</td>
<td>202009282242_nn-baeb9ef2d183</td>
<td>LCZero</td>
<td>v0.26.3-rct1_T60.SV.JH.92.190</td>
<td>+ 18 – 73 – 9</td>
</tr>
<tr>
<td>TCEC Season 20</td>
<td>Dec 2020 – Feb 2021</td>
<td>Stockfish</td>
<td>20210113</td>
<td>LCZero</td>
<td>0.27.0d-Tips-dje-magic_JH.94-100</td>
<td>+ 14 – 78 – 8</td>
</tr>
</tbody>
</table>

Figure: Top Chess Engine Championship results. (https://en.wikipedia.org/wiki/TopChessEngineChampionship)
Go

- Rules (simplified):
  - Players take turns in which they either place a stone on the board, or pass
  - If a group of stones is completely surrounded by opposing stones, it is removed from the board
  - After both players consecutively pass, the player with larger surrounded territory wins

- More complex than chess ($10^{43}$ vs. $10^{170}$ legal positions)

Figure: A position in Go. (https://en.wikipedia.org/wiki/Go_(game)/media/File:Fineart_vs_Galaxy.gif)
Modern Go-playing software

- Computers played Go at only amateur level until AlphaGo defeated human world champion Lee Sedol in a five-game match in 2016
- Final score was 4-1 for AlphaGo

Figure: 2016 match between AlphaGo and Lee Sedol. (https://blog.google/technology/ai/alphagos-ultimate-challenge/)
Shogi

- Japanese variant of chess
- More complex than chess
  - Larger board, wider variety of pieces
  - Captured pieces switch sides and may be returned to the board

*Figure*: Starting position in Shogi. (https://en.wikipedia.org/wiki/Shogi/media/File:Shogiban.png)
Heuristic alpha-beta tree search

- Based on the minimax search algorithm
  - One player aims to minimize the final score, the other player aims to maximize it
  - It is assumed that the opponent plays optimally
- Alpha-Beta pruning is used to reduce the size of the search tree
- Because of exponential complexity, the search is typically cut off early and a heuristic evaluation function is applied
Heuristic alpha-beta tree search

Figure: Explanatory diagram of heuristic minimax search. (http://zackmdavis.net/blog/wp-content/uploads/2019/05/game_tree.png)
Heuristic alpha-beta tree search - SOTA techniques

- Heuristic alpha-beta tree search is used e.g. by the Stockfish program
- Many advanced techniques are typically used to speed up the search, such as:
  - A fine-tuned heuristic evaluation function (hand-selected "features", such as material values of remaining pieces, are combined using a weighted linear function)
  - Move ordering (enables more efficient pruning)
  - Transposition tables (addresses the problem of duplicated states)
  - Quiescence (search is not cut off at complicated positions)
  - Databases of initial and endgame moves

- A single mistake of the heuristic evaluation function can be fatal, as it propagates to the root state of the search tree
Branching factor

- Branching factor of chess is about 35
  - Heuristic alpha-beta search to sufficient depth is possible
- Branching factor of Go is about 250
  - Heuristic alpha-beta search doesn’t work, as the explored tree is too shallow
- Claude Shannon, 1950:
  - Techniques such as heuristic alpha-beta search follow the **Type A strategy:**
    - “Explore a wide but shallow portion of the search tree”
  - The alternative is to follow the **Type B strategy:**
    - “Explore a deep but narrow portion of the search tree”
    - Moves that appear to be bad are ignored, promising subtrees are explored to the maximal possible depth
Monte Carlo Tree Search (MCTS)

- Instead of using a heuristic evaluation function, the value of a state is estimated as the average outcome of simulations of complete games from that state.
- The search tree is built while balancing two factors:
  - **Exploration**: We choose states that have had few playouts
  - **Exploitation**: We choose states that usually lead to victory
- Playouts are simulated by "playout policy"
Monte Carlo Tree Search (MCTS)

Figure: Explanatory diagram of Monte Carlo tree search. (TODO source)
Monte Carlo Tree Search (MCTS)

- The basic idea of Monte Carlo tree search (simulating moves, observing the outcome and using it to determine which moves are good) is, in essence, reinforcement learning.
- **Single mistakes (e.g. of playout policy) are typically averaged out over many playouts**
  - This favors the use of a (neural) heuristic
  - MCTS, along with an artificial neural network, is the basis of AlphaZero
Training procedure overview

- AlphaZero learns from self-play
- When it needs to decide what move to make, it runs a (slightly modified) MCTS from the current game state
- Similarly as before, AlphaZero’s MCTS balances two factors:
  - **Exploration**: We choose a state with low visit count
  - **Exploitation**: We choose a move/state with high move probability and high value of state
  - The move probabilities and the value of state are determined by a deep neural network
- The output of the MCTS is a vector of move probabilities, which is:
  - Immediately used for selecting the current move, and
  - Stored for subsequent update of the parameters of the neural network
- After we reach a terminal position of the game, the outcome of the game and the stored move probabilities of every state are used to train the neural network
AlphaZero’s neural network

- Input: encoded game state
- Two outputs: move probabilities, value of state
- Training is performed by gradient descent on a loss function that sums:
  - Cross-entropy losses of the move probabilities,
  - The mean squared error of the value of state, and
  - A regularization term
AlphaZero’s neural network: input encoding

- The input to the neural network is an encoded game state
- This encoding is a tensor of size $N \times N \times (MT + L)$:
  - $M$ planes of size $N \times N$, each plane indicating the positions of figures of one piece type
  - This is repeated $T$ times - once for each previous step of the game
  - There are also $L$ additional constant-valued planes denoting special features, such as:
    - Player’s color
    - Legality of special moves (i.e. castling in chess)
AlphaZero's neural network: architecture

- Input -> "body" -> two "heads" (one for outputting the move probabilities, the other for outputting the state value)
- Body: 19 residual convolutional blocks
  - 256 filters of kernel size 3x3 with stride 1
  - ReLU and batch normalization are utilized
- Move probability and state value heads are each only 2 layers deep
Inference procedure (playing)

- Same as the training procedure, only without updating the parameters of the neural network
- For a given time limit, at each turn, AlphaZero always "thinks" (runs MCTS) for 1/20th of the remaining time
5,000 TPUs (1st gen.) used to generate self-play games, 16 TPUs (2nd gen.) used to train the neural network

Durations of training:
- 9 hours in chess
  - First outperformed previous SOTA after 4 hours
- 12 hours in shogi
  - First outperformed previous SOTA after 4 hours
- 13 days in Go
  - First outperformed AlphaGo after 4 hours
## Training details

<table>
<thead>
<tr>
<th></th>
<th>Chess</th>
<th>Shogi</th>
<th>Go</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mini-batches</td>
<td>700k</td>
<td>700k</td>
<td>700k</td>
</tr>
<tr>
<td>Training Time</td>
<td>9h</td>
<td>12h</td>
<td>13d</td>
</tr>
<tr>
<td>Training Games</td>
<td>44 million</td>
<td>24 million</td>
<td>140 million</td>
</tr>
<tr>
<td>Thinking Time</td>
<td>800 sims</td>
<td>800 sims</td>
<td>800 sims</td>
</tr>
<tr>
<td></td>
<td>~ 40 ms</td>
<td>~ 80 ms</td>
<td>~ 200 ms</td>
</tr>
</tbody>
</table>

*Figure:* Selected statistics of AlphaZero training. (TODO source)
Training details

**Figure:** Progress of AlphaZero’s performance during training (Elo ratings). (TODO source)
Tournament evaluation

- Time controls: 3 hours of thinking time, with 15 additional seconds for each move
- 1000 games were played against every state-of-the-art program, which were:
  - AlphaGo Zero (in Go),
  - Stockfish (in chess), and
  - Elmo (in shogi)
- AlphaZero reached the new SOTA in all three games
Figure: Tournament evaluation of AlphaZero against previous SOTA programs. (TODO source)
Tournament evaluation

Figure: Scalability of AlphaZero with thinking time. (TODO source)
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Background</th>
<th>Adversarial search</th>
<th>AlphaZero internals</th>
<th>Training</th>
<th>Evaluation</th>
<th>Discussion</th>
</tr>
</thead>
</table>

Questions/Discussion