MuZero, AlphaZero Policy Target, Gumbel-Max, GumbelZero

Milan Straka
The MuZero algorithm extends the AlphaZero by a trained model, alleviating the requirement for a known MDP dynamics. It is evaluated both on board games and on the Atari domain.

At each time-step \( t \), for each of \( 1 \leq k \leq K \) steps, a model \( \mu_\theta \) with parameters \( \theta \), conditioned on past observations \( o_1, \ldots, o_t \) and future actions \( a_{t+1}, \ldots, a_{t+k} \), predicts three future quantities:

- the policy \( p_t^k \approx \pi(a_{t+k+1} \mid o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k}) \),
- the value function \( v_t^k \approx \mathbb{E}[u_{t+k+1} + \gamma u_{t+k+2} + \ldots \mid o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k}] \),
- the immediate reward \( r_t^k \approx u_{t+k} \),

where \( u_i \) are the observed rewards and \( \pi \) is the behaviour policy.
At each time-step $t$ (omitted from now on for simplicity), the model is composed of three components: a **representation** function, a **dynamics** function, and a **prediction** function.

- The dynamics function, $(r^k, s^k) \leftarrow g_\theta(s^{k-1}, a^k)$, simulates the MDP dynamics and predicts an immediate reward $r^k$ and an internal state $s^k$. The internal state has no explicit semantics, its only goal is to accurately predict rewards, values, and policies.
- The prediction function, $(p^k, v^k) \leftarrow f_\theta(s^k)$, computes the policy and the value function, similarly as in AlphaZero.
- The representation function, $s^0 \leftarrow h_\theta(o_1, \ldots, o_t)$, generates an internal state encoding the past observations.
Figure 1 of "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
MuZero – MCTS

The MCTS algorithm is very similar to the one used in AlphaZero, only the trained model is used. It produces a policy $\pi_t$ and a value estimate $\nu_t$.

- All actions, including the invalid ones, are allowed at any time, except at the root, where the invalid actions (available from the current state) are disallowed.
- No states are considered terminal during the search.
- During the backup phase, we consider a general discounted bootstrapped return

$$G_k = \sum_{t=0}^{l-k-1} \gamma^t r_{k+1+t} + \gamma^{l-k} \nu_l.$$ 

- Furthermore, the expected return is generally unbounded. Therefore, MuZero normalize the Q-value estimates to $[0, 1]$ range by using the minimum and maximum the values observed in the search tree until now:

$$\bar{Q}(s, a) = \frac{Q(s, a) - \min_{s', a' \in \text{Tree}} Q(s', a')} {\max_{s', a' \in \text{Tree}} Q(s', a') - \min_{s', a' \in \text{Tree}} Q(s', a')}.$$
MuZero – Action Selection

To select a move, we employ a MCTS algorithm and then sample an action from the obtained policy, $a_{t+1} \sim \pi_t$.

For games, the same strategy of sampling the actions $a_t$ as in AlphaZero is used. In the Atari domain, the actions are sampled according to the visit counts for the whole episode, but with a given temperature $T$:

$$
\pi(a|s) = \frac{N(s, a)^{1/T}}{\sum_b N(s, b)^{1/T}},
$$

where $T$ is decayed during training – for first 500k steps it is 1, for the next 250k steps it is 0.5 and for the last 250k steps it is 0.25.

While for the board games 800 simulations are used during MCTS, only 50 are used for Atari.

In case of Atari, the replay buffer consists of 125k sequences of 200 actions.
During training, we utilize a sequence of $K$ moves. We estimate the return using bootstrapping as $z_t = u_{t+1} + \gamma u_{t+2} + \ldots + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n}$. The values $K = 5$ and $n = 10$ are used in the paper, with batch size 2048 for the board games and 1024 for Atari.

The loss is then composed of the following components:

$$
\mathcal{L}_t(\theta) = \sum_{k=0}^{K} \mathcal{L}^r(u_{t+k}, r^k_t) + \mathcal{L}^v(z_{t+k}, v^k_t) + \mathcal{L}^p(\pi_{t+k}, p^k_t) + c\|\theta\|^2.
$$

Note that in Atari, rewards are scaled by $\text{sign}(x)(\sqrt{|x| + 1} - 1) + \varepsilon x$ for $\varepsilon = 10^{-3}$, and authors utilize a cross-entropy loss with 601 categories for values $-300, \ldots, 300$, which they claim to be more stable (this can be considered distributional RL).

Furthermore, in Atari the discount factor $\gamma = 0.997$ is used, and the replay buffer elements are sampled according to prioritized replay with priority $\propto |\nu - z|^\alpha$, and importance sampling with exponent $\beta$ is used to account for changing the sampling distribution ($\alpha = \beta = 1$ is used).
Model

\[
\begin{align*}
    s^0 &= h_\theta(o_1, \ldots, o_t) \\
    r^k, s^k &= g_\theta(s^{k-1}, a^k) \\
    p^k, v^k &= f_\theta(s^k)
\end{align*}
\]

Search

\[
\begin{align*}
    \nu_t, \pi_t &= MCTS(s^0_t, \mu_\theta) \\
    a_t &\sim \pi_t
\end{align*}
\]
Learning Rule

\[ \boldsymbol{p}_t^k, \boldsymbol{v}_t^k, \boldsymbol{r}_t^k = \mu_\theta(o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k}) \]

\[ z_t = \begin{cases} u_T \\ u_{t+1} + \gamma u_{t+2} + \ldots + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n} \end{cases} \text{ for general MDPs} \]

\[ \mathcal{L}_t(\theta) = \sum_{k=0}^{K} \mathcal{L}_t^r(u_{t+k}, r_t^k) + \mathcal{L}_t^v(z_{t+k}, v_t^k) + \mathcal{L}_t^p(\pi_{t+k}, p_t^k) + c\|\theta\|^2 \]

Losses

\[ \mathcal{L}_t^r(u, r) = \begin{cases} 0 \quad \text{for games} \\ -\varphi(u)^T \log \varphi(r) \quad \text{for general MDPs} \end{cases} \]

\[ \mathcal{L}_t^v(z, q) = \begin{cases} (z - q)^2 \quad \text{for games} \\ -\varphi(z)^T \log \varphi(q) \quad \text{for general MDPs} \end{cases} \]

\[ \mathcal{L}_t^p(\pi, p) = -\pi^T \log p \]
MuZero – Evaluation

Figure 2 of “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.
MuZero is optimized for greater sample efficiency. It revisits past trajectories by re-running the MCTS using the network with the latest parameters, notably

- using the fresh policy as target in 80% of the training updates, and
- always using the fresh $v^k \leftarrow f_\theta(s^k)$ in the bootstrapped target $z_t$.

Some hyperparameters were changed too – 2.0 samples were drawn per state instead of 0.1, the value loss was weighted down to 0.25, and the $n$-step return was reduced to $n = 5$. 

Table 1 of “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Median</th>
<th>Mean</th>
<th>Env. Frames</th>
<th>Training Time</th>
<th>Training Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ape-X [18]</td>
<td>434.1%</td>
<td>1695.6%</td>
<td>22.8B</td>
<td>5 days</td>
<td>8.64M</td>
</tr>
<tr>
<td>R2D2 [21]</td>
<td>1920.6%</td>
<td>4024.9%</td>
<td>37.5B</td>
<td>5 days</td>
<td>2.16M</td>
</tr>
<tr>
<td><em>MuZero</em></td>
<td><strong>2041.1%</strong></td>
<td><strong>4999.2%</strong></td>
<td><strong>20.0B</strong></td>
<td><strong>12 hours</strong></td>
<td><strong>1M</strong></td>
</tr>
<tr>
<td>IMPALA [9]</td>
<td>191.8%</td>
<td>957.6%</td>
<td>200M</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Rainbow [17]</td>
<td>231.1%</td>
<td>--</td>
<td>200M</td>
<td>10 days</td>
<td>--</td>
</tr>
<tr>
<td>UNREAL$^a$ [19]</td>
<td>250%$^a$</td>
<td>880%$^a$</td>
<td>250M</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>LASER [36]</td>
<td>431%</td>
<td>--</td>
<td>200M</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td><em>MuZero</em> Reanalyze*</td>
<td><strong>731.1%</strong></td>
<td><strong>2168.9%</strong></td>
<td><strong>200M</strong></td>
<td><strong>12 hours</strong></td>
<td><strong>1M</strong></td>
</tr>
</tbody>
</table>
Figure 3 of "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
Figure S3 of "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
### MuZero – Detailed Atari Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>alien</td>
<td>227.75</td>
<td>7,127.80</td>
<td>616.90</td>
<td>40,805.00</td>
<td>229,496.90</td>
<td>741,812.63</td>
<td>10,747.5 %</td>
</tr>
<tr>
<td>amidar</td>
<td>5.77</td>
<td>1,719.53</td>
<td>74.30</td>
<td>8,659.00</td>
<td>29,321.40</td>
<td>28,634.39</td>
<td>1,670.5 %</td>
</tr>
<tr>
<td>assault</td>
<td>222.39</td>
<td>742.00</td>
<td>527.20</td>
<td>24,559.00</td>
<td>108,197.00</td>
<td>143,972.03</td>
<td>27,664.9 %</td>
</tr>
<tr>
<td>asterix</td>
<td>210.00</td>
<td>8,503.33</td>
<td>1,128.30</td>
<td>313,305.00</td>
<td>999,153.30</td>
<td>998,425.00</td>
<td>12,036.4 %</td>
</tr>
<tr>
<td>asteroids</td>
<td>719.10</td>
<td>47,388.67</td>
<td>793.60</td>
<td>155,495.00</td>
<td>357,867.70</td>
<td>678,558.64</td>
<td>1,452.4 %</td>
</tr>
<tr>
<td>atlantis</td>
<td>12,850.00</td>
<td>29,028.13</td>
<td>20,992.50</td>
<td>94,498.00</td>
<td>1,620,764.00</td>
<td>1,674,767.20</td>
<td>10,272.6 %</td>
</tr>
<tr>
<td>bank heist</td>
<td>14.20</td>
<td>753.13</td>
<td>34.20</td>
<td>1,716.00</td>
<td>24,235.90</td>
<td>1,278.98</td>
<td>171.2 %</td>
</tr>
<tr>
<td>battle zone</td>
<td>2,360.00</td>
<td>37,187.50</td>
<td>4,031.20</td>
<td>98,895.00</td>
<td>751,880.00</td>
<td>848,623.00</td>
<td>2,429.9 %</td>
</tr>
<tr>
<td>beam rider</td>
<td>363.88</td>
<td>16,926.53</td>
<td>621.60</td>
<td>63,305.00</td>
<td>188,257.40</td>
<td>454,993.53</td>
<td>2,744.9 %</td>
</tr>
<tr>
<td>berzerk</td>
<td>123.65</td>
<td>2,630.42</td>
<td>-</td>
<td>57,197.00</td>
<td>53,318.70</td>
<td>85,932.60</td>
<td>3,423.1 %</td>
</tr>
<tr>
<td>bowling</td>
<td>23.11</td>
<td>160.73</td>
<td>30.00</td>
<td>18.00</td>
<td>219.50</td>
<td>260.13</td>
<td>172.2 %</td>
</tr>
<tr>
<td>boxing</td>
<td>0.05</td>
<td>12.06</td>
<td>7.80</td>
<td>100.00</td>
<td>98.50</td>
<td>100.00</td>
<td>832.2 %</td>
</tr>
<tr>
<td>breakout</td>
<td>1.72</td>
<td>30.47</td>
<td>16.40</td>
<td>801.00</td>
<td>837.70</td>
<td>864.00</td>
<td>2,999.2 %</td>
</tr>
<tr>
<td>centipede</td>
<td>2,090.87</td>
<td>12,017.04</td>
<td>-</td>
<td>12,974.00</td>
<td>599,140.30</td>
<td>1,159,049.27</td>
<td>11,655.6 %</td>
</tr>
<tr>
<td>chopper command</td>
<td>811.00</td>
<td>7,387.80</td>
<td>979.40</td>
<td>721,851.00</td>
<td>986,652.00</td>
<td>991,039.70</td>
<td>15,056.4 %</td>
</tr>
<tr>
<td>crazy climber</td>
<td>10,780.50</td>
<td>35,829.41</td>
<td>62,583.60</td>
<td>320,426.00</td>
<td>366,690.70</td>
<td>458,315.40</td>
<td>1,786.6 %</td>
</tr>
<tr>
<td>defender</td>
<td>2,874.50</td>
<td>18,688.89</td>
<td>-</td>
<td>411,944.00</td>
<td>665,792.00</td>
<td>839,642.95</td>
<td>5,291.2 %</td>
</tr>
<tr>
<td>demon attack</td>
<td>152.07</td>
<td>1,971.00</td>
<td>208.10</td>
<td>133,086.00</td>
<td>140,002.30</td>
<td>143,964.26</td>
<td>7,906.4 %</td>
</tr>
<tr>
<td>double dunk</td>
<td>-18.55</td>
<td>-16.40</td>
<td>-</td>
<td>24.00</td>
<td>23.70</td>
<td>23.94</td>
<td>1,976.3 %</td>
</tr>
<tr>
<td>enduro</td>
<td>0.00</td>
<td>860.53</td>
<td>-</td>
<td>217.70</td>
<td>2,372.70</td>
<td>2,382.44</td>
<td>276.9 %</td>
</tr>
<tr>
<td>fishing derby</td>
<td>-91.71</td>
<td>-38.80</td>
<td>-90.70</td>
<td>44.00</td>
<td>85.80</td>
<td>91.16</td>
<td>345.6 %</td>
</tr>
<tr>
<td>freeway</td>
<td>0.01</td>
<td>29.60</td>
<td>16.70</td>
<td>34.00</td>
<td>32.50</td>
<td>33.03</td>
<td>111.6 %</td>
</tr>
<tr>
<td>frostbite</td>
<td>65.20</td>
<td>4,334.67</td>
<td>236.90</td>
<td>9,329.00</td>
<td>315,456.40</td>
<td>631,378.53</td>
<td>14,786.7 %</td>
</tr>
<tr>
<td>gopher</td>
<td>257.60</td>
<td>2,412.50</td>
<td>596.80</td>
<td>120,301.00</td>
<td>124,776.30</td>
<td>130,345.58</td>
<td>6,036.8 %</td>
</tr>
<tr>
<td>gravitar</td>
<td>173.00</td>
<td>3,351.43</td>
<td>173.40</td>
<td>1,599.00</td>
<td>15,680.70</td>
<td>6,682.70</td>
<td>204.8 %</td>
</tr>
<tr>
<td>hero</td>
<td>1,026.97</td>
<td>30,826.38</td>
<td>2,656.60</td>
<td>31,656.00</td>
<td>39,537.10</td>
<td>49,244.11</td>
<td>161.8 %</td>
</tr>
<tr>
<td>ice hockey</td>
<td>-11.15</td>
<td>0.88</td>
<td>-11.60</td>
<td>33.00</td>
<td>79.30</td>
<td>67.04</td>
<td>650.0 %</td>
</tr>
<tr>
<td>jamesbond</td>
<td>29.00</td>
<td>302.80</td>
<td>100.50</td>
<td>21,323.00</td>
<td>25,354.00</td>
<td>41,063.25</td>
<td>14,986.9 %</td>
</tr>
<tr>
<td>kangaroo</td>
<td>52.00</td>
<td>3,035.00</td>
<td>51.20</td>
<td>1,416.00</td>
<td>14,130.70</td>
<td>16,763.60</td>
<td>560.2 %</td>
</tr>
</tbody>
</table>

### Table S1 of "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
## MuZero – Detailed Atari Results

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>krull</td>
<td>1,598.05</td>
<td>2,665.53</td>
<td>2,204.80</td>
<td>11,741.00</td>
<td>218,448.10</td>
<td>269,358.27</td>
<td>25,083.4 %</td>
</tr>
<tr>
<td>kung fu master</td>
<td>258.50</td>
<td>22,736.25</td>
<td>14,862.50</td>
<td>97,830.00</td>
<td>233,413.30</td>
<td>204,824.00</td>
<td>910.1 %</td>
</tr>
<tr>
<td>montezuma revenge</td>
<td>0.00</td>
<td>4,753.33</td>
<td>-</td>
<td>2,500.00</td>
<td>2,061.30</td>
<td>0.00</td>
<td>0.0 %</td>
</tr>
<tr>
<td>ms pacman</td>
<td>307.30</td>
<td>6,951.60</td>
<td>1,480.00</td>
<td>11,255.00</td>
<td>42,281.70</td>
<td>243,401.10</td>
<td>3,658.7 %</td>
</tr>
<tr>
<td>name this game</td>
<td>2,292.35</td>
<td>8,049.00</td>
<td>2,420.70</td>
<td>25,783.00</td>
<td>58,182.70</td>
<td>157,177.85</td>
<td>2,690.5 %</td>
</tr>
<tr>
<td>phoenix</td>
<td>761.40</td>
<td>7,242.60</td>
<td>-</td>
<td>224,491.00</td>
<td>864,020.00</td>
<td>955,137.84</td>
<td>14,725.3 %</td>
</tr>
<tr>
<td>pitfall</td>
<td>-229.44</td>
<td>6,463.69</td>
<td>-</td>
<td>-1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>3.4 %</td>
</tr>
<tr>
<td>pong</td>
<td>-20.71</td>
<td>14.59</td>
<td>12.80</td>
<td>21.00</td>
<td>21.00</td>
<td>21.00</td>
<td>118.2 %</td>
</tr>
<tr>
<td>private eye</td>
<td>24.94</td>
<td>69,571.27</td>
<td>35.00</td>
<td>50.00</td>
<td>5,322.70</td>
<td>15,299.98</td>
<td>22.0 %</td>
</tr>
<tr>
<td>qbert</td>
<td>163.88</td>
<td>13,455.00</td>
<td>1,288.80</td>
<td>302,391.00</td>
<td>408,850.00</td>
<td>72,276.00</td>
<td>542.6 %</td>
</tr>
<tr>
<td>riverraid</td>
<td>1,338.50</td>
<td>17,118.00</td>
<td>1,957.80</td>
<td>63,864.00</td>
<td>45,632.10</td>
<td>323,417.18</td>
<td>2,041.1 %</td>
</tr>
<tr>
<td>road runner</td>
<td>11.50</td>
<td>7,845.00</td>
<td>5,640.60</td>
<td>222,235.00</td>
<td>599,246.70</td>
<td>613,411.80</td>
<td>7,830.5 %</td>
</tr>
<tr>
<td>robotank</td>
<td>2.16</td>
<td>11.94</td>
<td>-</td>
<td>74.00</td>
<td>100.40</td>
<td>131.13</td>
<td>1,318.7 %</td>
</tr>
<tr>
<td>seaquest</td>
<td>68.40</td>
<td>42,054.71</td>
<td>683.30</td>
<td>392,952.00</td>
<td>999,996.70</td>
<td>999,976.52</td>
<td>2,381.5 %</td>
</tr>
<tr>
<td>skiing</td>
<td>-17,098.09</td>
<td>-4,336.93</td>
<td>-10,790.00</td>
<td>-30,021.70</td>
<td>-29,968.36</td>
<td>-100.9 %</td>
<td>-100.9 %</td>
</tr>
<tr>
<td>solaris</td>
<td>1,236.30</td>
<td>32,362.67</td>
<td>-2,893.00</td>
<td>3,787.20</td>
<td>56.62</td>
<td>-10.6 %</td>
<td></td>
</tr>
<tr>
<td>space invaders</td>
<td>148.03</td>
<td>1,668.67</td>
<td>-54,681.00</td>
<td>43,223.40</td>
<td>74,335.30</td>
<td>4,878.87</td>
<td></td>
</tr>
<tr>
<td>star gunner</td>
<td>664.00</td>
<td>10,250.00</td>
<td>-434,343.00</td>
<td>717,344.00</td>
<td>549,271.70</td>
<td>5,723.00</td>
<td></td>
</tr>
<tr>
<td>surround</td>
<td>-9.99</td>
<td>6.53</td>
<td>-7.00</td>
<td>9.90</td>
<td>9.99</td>
<td>120.9 %</td>
<td></td>
</tr>
<tr>
<td>tennis</td>
<td>-23.84</td>
<td>-8.27</td>
<td>-24.00</td>
<td>-0.10</td>
<td>0.00</td>
<td>153.1 %</td>
<td></td>
</tr>
<tr>
<td>time pilot</td>
<td>3,568.00</td>
<td>5,229.10</td>
<td>-87,085.00</td>
<td>445,377.30</td>
<td>476,763.90</td>
<td>28,486.9 %</td>
<td></td>
</tr>
<tr>
<td>tutankham</td>
<td>11.43</td>
<td>167.59</td>
<td>-273.00</td>
<td>395.30</td>
<td>491.48</td>
<td>307.4 %</td>
<td></td>
</tr>
<tr>
<td>up n down</td>
<td>533.40</td>
<td>11,693.23</td>
<td>3,350.30</td>
<td>401,884.00</td>
<td>589,226.90</td>
<td>715,545.61</td>
<td>6,407.0 %</td>
</tr>
<tr>
<td>venture</td>
<td>0.00</td>
<td>1,187.50</td>
<td>-1,813.00</td>
<td>1,970.70</td>
<td>0.40</td>
<td>0.0 %</td>
<td></td>
</tr>
<tr>
<td>video pinball</td>
<td>0.00</td>
<td>17,667.90</td>
<td>-565,163.00</td>
<td>999,383.20</td>
<td>981,791.88</td>
<td>5,556.9 %</td>
<td></td>
</tr>
<tr>
<td>wizard of wor</td>
<td>563.50</td>
<td>4,756.52</td>
<td>-46,204.00</td>
<td>144,362.70</td>
<td>197,126.00</td>
<td>4,687.9 %</td>
<td></td>
</tr>
<tr>
<td>yars revenge</td>
<td>3,092.91</td>
<td>54,576.93</td>
<td>5,664.30</td>
<td>148,595.00</td>
<td>995,048.40</td>
<td>553,311.46</td>
<td>1,068.7 %</td>
</tr>
<tr>
<td>zaxxon</td>
<td>32.50</td>
<td>9,173.30</td>
<td>-42,286.00</td>
<td>224,910.70</td>
<td>725,853.90</td>
<td>7,940.5 %</td>
<td></td>
</tr>
</tbody>
</table>

**# best** | 0 | 5 | 0 | 5 | 13 | 37

*Table S1 of “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.*
Recall that in AlphaZero, actions are selected according to a variant of PUCT algorithm:

\[ a^* = \arg \max_a \left( Q(s, a) + C(s)P(s, a) \frac{\sqrt{N(s)}}{1 + N(s, a)} \right), \]

with a slightly time-increasing exploration rate \( C(s) = \log \left( \frac{1 + N(s) + 19625}{19625} \right) + 1.25 \approx 1.25. \)

The paper Jean-Bastien Grill et al.: Monte-Carlo Tree Search as Regularized Policy Optimization, the authors have shown how to interpret this algorithm as a regularized policy optimization.
Policy optimization is usually an iterative procedure, which in every step improves a current policy $\pi_{\theta_0}$ according to

$$\pi_{\theta'} \overset{\text{def}}{=} \arg\max_{\pi \in \mathcal{S}} q^T_{\pi_{\theta_0}} y - R(y, \pi_{\theta_0}),$$

where $\mathcal{S}$ is a $|\mathcal{A}|$-dimensional simplex and $R : \mathcal{S}^2 \to \mathbb{R}$ is an optional (usually convex) regularization term.

- with $R = 0$, the above reduces to policy iteration (used for example in DQN);
- with $R = 0$, if the policy is updated using a single gradient step, the algorithm reduces to policy gradient;
- when $R(y, \pi_{\theta_0}) = -H(y)$, we recover the Soft Actor Critic objective;
- for $R(y, \pi_{\theta_0}) = D_{KL}(\pi_{\theta_0} \| y)$ we get an analogue of the TRPO objective, which motivated PPO;
- the MPO algorithm (which we did not discuss) employs $R(y, \pi_{\theta_0}) = D_{KL}(y \| \pi_{\theta_0})$. 

NPFL139, Lecture 12  MuZero  μ0Model  μ0MCTS  μ0Training  AlphaZero Policy Target  Gumbel-Max  GumbelZero
Let us define the empirical visit distribution $\hat{\pi}$ as

$$\hat{\pi}(a|s) \overset{\text{def}}{=} \frac{1 + N(s, a)}{|A| + \sum_b N(s, b)}.$$  

The added plus ones makes the following analysis easier, but are not strictly necessary. We also define the multiplier $\lambda_N$ as

$$\lambda_N(s) \overset{\text{def}}{=} C(s) \cdot \frac{\sqrt{\sum_b N(s, b)}}{|A| + \sum_b N(s, b)}.$$  

With these definitions, we can rewrite the AlphaZero action selection to

$$a^* = \arg \max_a \left( Q(s, a) + \lambda_N \cdot \frac{\pi_{\theta}(a|s)}{\hat{\pi}(a|s)} \right).$$
AlphaZero as Regularized Policy Optimization

\[ a^* = \text{arg max}_a \left( Q(s, a) + \lambda N \cdot \frac{\pi_\theta(a|s)}{\hat{\pi}(a|s)} \right) \]

For notational simplicity, we will represent \( Q(s, a) \) as a vector \( q \), where \( q_a = Q(s, a) \), and similarly the policies as \( \pi_\theta, \hat{\pi} \).

Furthermore, for two vectors \( a, b \), let \( \frac{a}{b} \) denote element-wise division with \( \left( \frac{a}{b} \right)_i \overset{\text{def}}{=} \frac{a_i}{b_i} \).

With this notation, the action selection can be succinctly written as

\[ a^* = \text{arg max}_a \left( q + \lambda N \frac{\pi_\theta}{\hat{\pi}} \right). \]
Let $\bar{\pi}$ be the solution of the following objective:

$$\bar{\pi} \stackrel{\text{def}}{=} \arg \max_{y \in S} \left( q^T y - \lambda_N D_{KL}(\pi_\theta \parallel y) \right).$$

It can be shown that this solution can be computed explicitly as

$$\bar{\pi} = \lambda_N \frac{\pi_\theta}{\alpha - q},$$

where $\alpha \in \mathbb{R}$ is set such that the result is a proper distribution.

- Note that $\alpha \geq \max_{b \in A} \left( q_b + \lambda_N \pi_\theta(b) \right)$, because $\bar{\pi}(a)$ must be at most 1.
- Furthermore, $\alpha \leq \max_{b \in A} (q_b) + \lambda_N$, because we need $\sum_a \bar{\pi}(a) = 1$ and we combine $\sum_a \frac{\lambda_N \pi_\theta(a)}{\max_b(q_b) + \lambda_N - q_a} \leq \sum_a \frac{\lambda_N \pi_\theta(a)}{\lambda_N} = 1$ with the fact that $\sum_a \frac{\lambda_N \pi_\theta(a)}{\alpha - q_a}$ is a decreasing function of $\alpha \geq \max_b q_b$.

Note the $\lambda_N \approx 1/\sqrt{N}$ decreasing the regularization for increasing number of simulations.
The action $a^*$ selected by the AlphaZero algorithm fulfills

$$a^* = \arg \max_a \left( \frac{\partial}{\partial N(s, a)} \left( q^T \hat{\pi} - \lambda N D_{KL}(\pi_\theta \| \hat{\pi}) \right) \right).$$

In other words, $\hat{\pi}$ “tracks” $\bar{\pi}$.

Furthermore, it can be also shown that for the selected action $a^*$,

$$\hat{\pi}(a^*|s) \leq \bar{\pi}(a^*|s),$$

until in the limit, the two distributions coincide.
The $\bar{\pi}$ can be used in the AlphaZero algorithm in several ways:

- **Act**: the action in self-play games could be sampled according to $\bar{\pi}(\cdot | s_{\text{root}})$ instead of $\hat{\pi}$;
- **Search**: during search, we could sample the actions stochastically according to $\bar{\pi}$ instead of the PUCT rule;
- **Learn**: we could use $\bar{\pi}$ as the target policy during training instead of $\hat{\pi}$;
- **All**: all of the above.
Figure 1. Comparison of the score (median score over 3 seeds) of MuZero (red: using $\hat{\pi}$) and ALL (blue: using $\bar{\pi}$) after 100k learner steps as a function of $N_{\text{sim}}$ on Cheetah Run of the Control Suite.

Figure 5. Comparison of the median score over 3 seeds of MuZero (red) and ALL (blue) at 4 (dotted) and 50 (solid line) simulations per step on Cheetah Run (Control Suite).
Figure 2. Comparison of median scores of MuZero (red) and ALL (blue) at $N_{\text{sim}} = 5$ (dotted line) and $N_{\text{sim}} = 50$ (solid line) simulations per step on Ms Pacman (Atari). Averaged across 8 seeds.

Figure 3. Comparison of median score (solid lines) over 6 seeds of MuZero and ALL on four Atari games with $N_{\text{sim}} = 50$. The shaded areas correspond to the range of the best and worst seed. ALL (blue) performs consistently better than MuZero (red).
Figure 4. Ablation study at 5 and 50 simulations per step on Ms Pacman (Atari); average across 8 seeds.

Figure 4 of “Monte-Carlo Tree Search as Regularized Policy Optimization”, https://arxiv.org/abs/2007.12509
Let $z$ be a categorical variable with class probabilities $p = (p_1, p_2, \ldots, p_K)$.

The Gumbel-Max trick (based on a 1954 theorem from E. J. Gumbel) states that we can draw samples $z \sim p$ using

$$z = \text{one-hot} \left( \arg \max_i (g_i + \log p_i) \right),$$

where $g_i$ are independent samples drawn from the Gumbel$(0, 1)$ distribution.

To sample $g$ from the distribution Gumbel$(0, 1)$, we can sample $u \sim U(0, 1)$ and then compute $g = -\log(-\log u)$. 

First recall that exponential distribution $\text{Exp}(\lambda)$ has
\[
\text{PDF}(x; \lambda) = \lambda e^{-\lambda x}, \quad \text{CDF}(x; \lambda) = 1 - e^{-\lambda x}.
\]

The standard Gumbel$(0, 1)$ distribution has
\[
\text{PDF}(x) = e^{-x}e^{-x}, \quad \text{CDF}(x) = e^{-e^{-x}}.
\]

The Gumbel distribution can be used to model the distribution of maximum of a number of samples from the exponential distribution: if $\tilde{x}$ is a maximum of $N$ samples from the $\text{Exp}(1)$ distribution, we get that
\[
P(\tilde{x} - \log N \leq x) = P(\tilde{x} \leq x + \log N) = \text{CDF}_{\text{Exp}(1)}(x + \log N)^N = \left(1 - \frac{e^{-x}}{N}\right)^N,
\]
which converges to $e^{-e^{-x}} = \text{CDF}_{\text{Gumbel}(0,1)}(x)$ for $N \to \infty$. 
Gumbel-Max Trick Proof

To prove the Gumbel-Max trick, we first reformulate it slightly.

Let $l_i$ be logits of a categorical distribution (so that the class probabilities $\pi_i \propto e^{l_i}$), and let $g_i \sim \text{Gumbel}(0, 1)$. Then

$$\pi_k = P(k = \arg\max_i (g_i + l_i)).$$

We first observe that the theorem is invariant to a scalar shift of logits, so we can without loss of generality assume that $\sum_i e^{l_i} = 1$ and $\pi_i = e^{l_i}$.

For convenience, denote $u_i \overset{\text{def}}{=} g_i + l_i$.

We will use both the PDF and CDF of a Gumbel$(0, 1)$ distribution:

$$\text{PDF}(g_i) = e^{-g_i} e^{-g_i},$$
$$\text{CDF}(g_i) = e^{-e^{-g_i}}.$$
To finish the proof, we compute

\[
P\left(k = \arg \max_i (g_i + l_i) \right) = P(u_k \geq u_i, \forall i \neq k)
\]

\[
= \int P(u_k) \prod_{i \neq k} P(u_k \geq u_i \mid u_k) \, du_k
\]

\[
= \int P(g_k \mid g_k = u_k - l_k) \prod_{i \neq k} P(g_i \leq u_k - l_i \mid u_k) \, du_k
\]

\[
= \int e^{l_k - u_k - e^{l_k - u_k}} \prod_{i \neq k} e^{-e^{l_i - u_k}} \, du_k
\]

\[
= \int \pi_k e^{-u_k - \pi_k} \prod_{i \neq k} e^{-\pi_i e^{-u_k}} \, du_k
\]

\[
= \pi_k \int e^{-u_k - e^{-u_k}} \sum_i \pi_i \, du_k
\]

\[
= \pi_k \int e^{-g_k - e^{-g_k}} \, dg_k = \pi_k \cdot 1 = \pi_k.
\]
In AlphaZero, using the MCTS visit counts as the target policy fails to improve the policy for small number of visits.

Figure 2 of “Policy improvement by planning with Gumbel”, https://openreview.net/forum?id=bERaNdoegnO

In Ivo Danihelka et al.: Policy Improvement by Planning with Gumbel, several AlphaZero/MuZero improvements are proposed; among other a different target policy, which guarantees improvement.
Let $\pi$ be a categorical distributions parametrized with $\text{logits}(a)$. Let $g \in \mathbb{R}^k$ be a vector of independent Gumbel(0, 1) random variables.

The Gumbel-Max trick states that

$$A = \arg\max_a \left( g(a) + \text{logits}(a) \right)$$

has a distribution $A \sim \pi$.

The Gumbel-Max trick can be generalized to **Gumbel-Top-k** trick, capable of producing $n$ actions without replacement by considering the top $n$ scoring actions $\text{argtop}(g + \text{logits}, n)$:

$$A_1 = \arg\max_a \left( g(a) + \text{logits}(a) \right),$$
$$A_2 = \arg\max_{a \neq A_1} \left( g(a) + \text{logits}(a) \right),$$
$$\vdots$$
$$A_n = \arg\max_{a \notin \{A_1, \ldots, A_{n-1}\}} \left( g(a) + \text{logits}(a) \right).$$
A. Guaranteed Policy Improvement

For a small number of simulations, PUCT does not guarantee policy improvement.

- Consider for example three actions with prior policy (50%, 30%, 20%) and action values (0, 0, 1).
- The PUCT rule will select the first two actions.
- However, the value function of any policy considering just the first two actions is 0, which is worse than the value function of the prior policy.

In GumbelZero, we start by sampling \( n \) actions without replacement using the Gumbel-Max trick with Gumbel noise \( g \).

Our first attempt is to define a one-hot policy selecting an action \( A_{n+1} \) such that

\[
A_{n+1} = \arg \max_{a \in \{A_1, \ldots, A_n\}} \left( g(a) + \text{logits}(a) + \sigma(q(a)) \right),
\]

where \( \sigma \) can be any monotonically increasing transformation.
Algorithm 1 Policy Improvement by Planning with Gumbel

Require: $k$: number of actions.
Require: $n \leq k$: number of simulations.
Require: $\text{logits} \in \mathbb{R}^k$: predictor logits from a policy network $\pi$.

Sample $k$ Gumbel variables:
\[(g \in \mathbb{R}^k) \sim \text{Gumbel}(0)\]

Find $n$ actions with the highest $g(a) + \text{logits}(a)$:
\[A_{\text{topn}} = \text{argtop}(g + \text{logits}, n)\]

Get $q(a)$ for each $a \in A_{\text{topn}}$ by visiting the actions.

From the $A_{\text{topn}}$ actions, find the action with the highest $g(a) + \text{logits}(a) + \sigma(q(a))$:
\[A_{n+1} = \text{argmax}_{a \in A_{\text{topn}}} (g(a) + \text{logits}(a) + \sigma(q(a)))\]

return $A_{n+1}$

Algorithm 1 of “Policy improvement by planning with Gumbel”, https://openreview.net/forum?id=bERaNdOegn0
A policy choosing the action

\[ A_{n+1} = \arg \max_{a \in \{A_1, \ldots, A_n\}} (g(a) + \text{logits}(a) + \sigma(q(a)))) , \]

guarantees policy improvement, i.e., \( \mathbb{E}[q(A_{n+1})] \geq \mathbb{E}_{a \sim \pi}[q(a)] \).

Considering a fixed Gumbel noise \( g \), we get that

\[ q(\arg \max_{a \in \{A_1, \ldots, A_n\}} g(a) + \text{logits}(a) + \sigma(q(a))) \geq q(\arg \max_{a \in \{A_1, \ldots, A_n\}} g(a) + \text{logits}(a)) , \]

- either the action chosen on both sides is the same and we get an equality, or
- the action on the left side is different, meaning it has larger \( q(a) \).

Finally, if the inequality holds for any \( g \), it holds also in expectation. With the Gumbel-Max trick transforming the expectation of the right side to sampling an action \( a \sim \pi \), we get the required \( \mathbb{E}[q(A_{n+1})] \geq \mathbb{E}_{a \sim \pi}[q(a)] \).
B. Planning on Stochastic Bandit

When we get only an estimate $\hat{q}(a)$ of the action-value function, it is probably beneficial to visit an action multiple times.

Furthermore, choosing actions in the root using a UCB-like rule is not optimal:

- UCB minimizes cumulative regret, i.e., maximizes the sum of the obtained returns;
- in the root our goal is to obtain the best possible $A_{n+1}$, i.e., maximize just $\mathbb{E}[q(A_{n+1})]$.

The authors evaluated several simple regret minimization algorithms, and chose Sequential Halving (because it was easier to tune and does not have problem-dependent parameters).
Figure 1: The number of considered actions and their visit counts $N(a)$, when using Sequential Halving with Gumbel on a $k$-armed stochastic bandit. The search uses $n = 200$ simulations and starts by sampling $m = 16$ actions without replacement. Sequential Halving divides the budget of $n$ simulations equally to $\log_2(m)$ phases. In each phase, all considered actions are visited equally often. After each phase, one half of the actions is rejected. From the original $k$ actions, only the best actions will remain.
Algorithm 2 Sequential Halving with Gumbel

**Require:** \( k \): number of actions.
**Require:** \( m \leq k \): number of actions sampled without replacement.
**Require:** \( n \): number of simulations.
**Require:** \( \logits \in \mathbb{R}^k \): predictor logits from a policy network \( \pi \).

Sample \( k \) Gumbel variables:
\[
(g \in \mathbb{R}^k) \sim \text{Gumbel}(0)
\]
Find \( m \) actions with the highest \( g(a) + \logits(a) \):
\[
A_{\text{topm}} = \text{argtop}(g + \logits, m)
\]
Use Sequential Halving with \( n \) simulations to identify the best action from the \( A_{\text{topm}} \) actions, by comparing \( g(a) + \logits(a) + \sigma(\hat{q}(a)) \).
\[
A_{n+1} = \text{arg max}_{a \in \text{Remaining}} (g(a) + \logits(a) + \sigma(\hat{q}(a)))
\]
**return** \( A_{n+1} \)

The authors utilize \( m = \min(n, 16) \), and visit each action at least once even when \( n \) is small by visiting each action \( \max \left( 1, \left\lfloor \frac{n}{\log_2 m} \right\rfloor \right) \); after \( n \) simulation, the search is always stopped.
Using a one-hot policy based on

\[
A_{n+1} = \arg \max_{a \in \{A_1, \ldots, A_n\}} \left( g(a) + \text{logits}(a) + \sigma(q(a)) \right)
\]

results in using a simple policy loss

\[
L_{\text{simple}}(\pi) = - \log \pi(A_{n+1}).
\]
However, more information from the search might be extracted by using all action-value functions $q(a)$ produced by the search.

- First, we complete the action values using

$$completedQ(a) \overset{\text{def}}{=} \begin{cases} q(a) & \text{if } N(a) > 0, \\ v_\pi & \text{otherwise.} \end{cases}$$

- Then, we define improved policy as

$$\pi' = \text{softmax} \left( \text{logits}(a) + \sigma(completedQ(a)) \right).$$

It can be again proven (appendix C of the paper) that $\pi'$ is an improved policy, so

$$\mathbb{E}_{a \sim \pi'}[q(a)] \geq \mathbb{E}_{a \sim \pi}[q(a)].$$

- A natural loss is then

$$L_{\text{completed}}(\pi) = D_{\text{KL}}(\pi' \| \pi).$$
The authors propose to use

$$\sigma(\hat{q}(a)) \overset{\text{def}}{=} (c_{\text{visit}} + \max_b N(b)) c_{\text{scale}} \hat{q}(a),$$

for $c_{\text{visit}} = 50$, $c_{\text{scale}} = 1.0$.

Furthermore, the authors propose a consistent approximation to $v_\pi$ based on a network-predicted $\hat{v}_\pi$ and the $q(a)$ of the visited actions:

$$v_{\text{mix}} \overset{\text{def}}{=} \frac{1}{1 + \sum_b N(b)} \left( \hat{v}_\pi + \left( \sum_b N(b) \right) \frac{\sum_{a, N(a) > 0} \pi(a)q(a)}{\sum_{a, N(a) > 0} \pi(a)} \right).$$

Overall, the algorithm denoted in the paper as Gumbel MuZero utilizes Sequential Halving with Gumbel and trains using the improved policy combining logits and action values completed by $v_{\text{mix}}$; otherwise it is the same as MuZero.
D. Action Selection in Non-Root Nodes

We might consider utilizing the improved policy $\pi'$ also in the non-root nodes, by for example sampling actions from it. Additionally, the authors provide a deterministic algorithm of choosing non-root actions minimizing the difference between $\pi'$ and the current visit counts:

$$a^* = \arg\min_a \sum_b \left( \pi'(b) - \frac{N(b) + [a = b]}{1 + \sum_c N(c)} \right)^2.$$

This formula can be simplified to

$$a^* = \arg\max_a \left( \pi'(a) - \frac{N(a)}{1 + \sum_b N(b)} \right).$$

When this action selection is used, the authors call the algorithm Full Gumbel MuZero.
GumbelZero, D. Action Selection in Non-Root Nodes

\[ a^* = \arg\min_a \sum_b \left( \pi'(b) - \frac{N(b) + [a = b]}{1 + \sum_c N(c)} \right)^2 \]

\[ = \arg\min_a \sum_b \left( \left( \pi'(b) - \frac{N(b)}{1 + \sum_c N(c)} \right) - \frac{[a = b]}{1 + \sum_c N(c)} \right)^2 \]

\[ = \arg\min_a \sum_b -2 \left( \pi'(b) - \frac{N(b)}{1 + \sum_c N(c)} \right) \frac{[a = b]}{1 + \sum_c N(c)} \]

\[ = \arg\min_a - \sum_b \left( \pi'(b) - \frac{N(b)}{1 + \sum_c N(c)} \right) [a = b] \]

\[ = \arg\max_a \left( \pi'(a) - \frac{N(a)}{1 + \sum_b N(b)} \right) \]
“Replacement” is a Gumbel MuZero ablation sampling actions with replacement.

“TRPO MuZero”, “MPO MuZero” use Act+Search+Learn using the previously described regularized policy with $D_{KL}(\pi || \pi_{new})$ and $D_{KL}(\pi_{new} || \pi)$ regularizer, respectively.

Figure 2: Elo on 9x9 Go, when training with $n \in \{2, 4, 16, 32, 200\}$ simulations. Evaluation uses 800 simulations. Shades denote standard errors from 2 seeds.

Figure 2 of "Policy improvement by planning with Gumbel", https://openreview.net/forum?id=bERaNdoegnO
Figure 3: Gumbel MuZero ablations on 9x9 Go. (a) Policy loss ablations, when training with \( n \in \{2, 4, 16, 200\} \) simulations. Gumbel MuZero uses the policy loss with completed Q-values. (b) Sensitivity to the number of sampled actions. Gumbel MuZero samples \( m \) actions without replacement.
Gumbel AlphaZero and MuZero

Figure 6: Detailed policy loss ablations. Gumbel MuZero uses the policy loss with Q-values completed by the $\nu_{\text{mix}}$ value estimator from Appendix D. That works better than Q-values completed by the raw value network $\hat{v}_\pi$.

*Figure 6 of “Policy improvement by planning with Gumbel”, https://openreview.net/forum?id=bERaNdoegnO*
Figure 7: A comparison of different action selections at the non-root nodes. Gumbel MuZero uses the unmodified (deterministic) MuZero action selection at non-root nodes. Full Gumbel MuZero uses the deterministic action selection from Equation 14, which we compare to stochastic sampling from $\pi'$ at non-root nodes.

Figure 7 of "Policy improvement by planning with Gumbel", https://openreview.net/forum?id=bERaNdoegnO
Figure 8: Additional Gumbel MuZero ablations on 9x9 Go. (a) Sensitivity to Q-value scaling by $c_{\text{visit}}$. (b) On the perfect-information game, Gumbel MuZero used zero Gumbel noise at evaluation. Although, evaluation with stochastic Gumbel noise is not worse. During training, MuZero and Gumbel MuZero benefit from explorative acting proportional to the visit counts.

Figure 8 of "Policy improvement by planning with Gumbel", https://openreview.net/forum?id=bERaNdcegnO
Figure 4: Large-scale experiments with \( n = 400 \) simulations per move. (a) Elo on 19x19 Go, when training MuZero. (b) Elo on chess, when training AlphaZero.

*Figure 4 of "Policy improvement by planning with Gumbel", https://openreview.net/forum?id=bERaNdoegnO*
Figure 5: Atari results. (a) Mean return on `ms_pacman`, when training Gumbel MuZero and MuZero with \( n \in \{2, 4, 16, 18, 50\} \) simulations. MuZero fails to learn from 4 or less simulations. (b) Mean return on `beam_rider` for Gumbel MuZero with \( c_{\text{scale}} \in \{0.01, 0.1, 1, 10, 100\} \), compared to MuZero with \( n = 50 \) simulations. Shades denote standard errors from 10 seeds.

Figure 5 of “Policy improvement by planning with Gumbel”, https://openreview.net/forum?id=bERaNdoegnO