MuZero, PlaNet

Milan Straka

December 13, 2021
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 $p^k_t \approx \pi(a_{t+k+1}|o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k})$,
- the value function $v^k_t \approx \mathbb{E}[u_{t+k+1} + \gamma u_{t+k+2} + \ldots | o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k}]$,
- the immediate reward $r^k_t \approx u_{t+k}$,

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, $r^k, s^k = g_\theta(s^{k-1}, a^k)$, 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 $\pi^k, v^k = f_\theta(s^k)$, computes the policy and value function, similarly as in AlphaZero.
- The representation function, $s^0 = h_\theta(o_1, \ldots, o_t)$, generates an internal state encoding the past observations.
Figure 1 of the paper "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
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 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 normalizes the Q-value estimates to $[0, 1]$ range by using minimum and maximum 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 the obtained policy, \( a_{t+1} \sim \pi_t \).

For games, the same strategy of sampling the actions \( a_t \) is used. In the Atari domain, the actions are sampled according to 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.

The loss is then composed of the following components:

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

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.

Furthermore in Atari, the discount factor $\gamma = 0.997$ is used and the replay buffer elements are sampled according to prioritized replay and importance sampling is used to account for changing the sampling distribution.
MuZero

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

\[ p_t^k, v_t^k, r_t^k = \mu_\theta(o_1, \ldots, o_t, a_{t+1}, \ldots, a_{t+k}) \]

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

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

Losses

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

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

\[ L^p(\pi, p) = -\pi^T \log p \]
Figure 2 of the paper "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
MuZero – Atari Results

<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 Reanalyze</em></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>

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

MuZero Reanalyze is optimized for greater sample efficiency. It revisits past trajectories using the network with the latest parameters (using the fresh policy in 80% of the training steps).
Figure 3 of the paper “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.
Figure S3 of the paper "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model" by Julian Schrittwieser et al.
### Table S1 of the paper “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.

<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><strong>29,321.40</strong></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><strong>143,972.03</strong></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><strong>999,153.30</strong></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><strong>678,558.64</strong></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>944,498.00</td>
<td>1,620,764.00</td>
<td><strong>1,674,767.20</strong></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><strong>24,235.90</strong></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><strong>848,623.00</strong></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><strong>454,993.53</strong></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><strong>85,932.60</strong></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><strong>260.13</strong></td>
<td>172.2 %</td>
</tr>
<tr>
<td>boxing</td>
<td>0.05</td>
<td>12.06</td>
<td>7.80</td>
<td><strong>100.00</strong></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><strong>864.00</strong></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><strong>1,159,049.27</strong></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><strong>991,039.70</strong></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><strong>458,315.40</strong></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><strong>839,642.95</strong></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><strong>143,964.26</strong></td>
<td>7,906.4 %</td>
</tr>
<tr>
<td>double dunk</td>
<td>-18.55</td>
<td>-16.40</td>
<td>-</td>
<td><strong>24.00</strong></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>2,177.00</td>
<td>2,372.70</td>
<td><strong>2,382.44</strong></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><strong>91.16</strong></td>
<td>345.6 %</td>
</tr>
<tr>
<td>freeway</td>
<td>0.01</td>
<td>29.60</td>
<td>16.70</td>
<td><strong>34.00</strong></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><strong>631,378.53</strong></td>
<td>14,878.7 %</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><strong>130,345.58</strong></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><strong>15,680.70</strong></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><strong>49,244.11</strong></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><strong>79.30</strong></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><strong>41,063.25</strong></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><strong>16,763.60</strong></td>
<td>560.2 %</td>
</tr>
</tbody>
</table>

# best

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>0</th>
<th>5</th>
<th>13</th>
<th>37</th>
</tr>
</thead>
</table>

Table S1 of the paper “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><strong>269,358.27</strong></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><strong>233,413.30</strong></td>
<td>204,824.00</td>
<td>910.1 %</td>
</tr>
<tr>
<td>montezuma revenge</td>
<td>0.00</td>
<td><strong>4,753.33</strong></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><strong>243,401.10</strong></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><strong>157,177.85</strong></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><strong>955,137.84</strong></td>
<td>14,725.3 %</td>
</tr>
<tr>
<td>pitfall</td>
<td>-229.44</td>
<td><strong>6,463.69</strong></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><strong>21.00</strong></td>
<td>21.00</td>
<td>21.00</td>
<td>118.2 %</td>
</tr>
<tr>
<td>private eye</td>
<td>24.94</td>
<td><strong>69,571.27</strong></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><strong>408,850.00</strong></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><strong>323,417.18</strong></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><strong>613,411.80</strong></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><strong>131.13</strong></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><strong>999,996.70</strong></td>
<td>999,976.52</td>
<td>2,381.5 %</td>
</tr>
<tr>
<td>skiing</td>
<td>-17,098.09</td>
<td><strong>4,336.93</strong></td>
<td>-</td>
<td>-10,790.00</td>
<td>-30,021.70</td>
<td>-29,968.36</td>
<td>-100.9 %</td>
</tr>
<tr>
<td>solaris</td>
<td>1,236.30</td>
<td><strong>12,326.67</strong></td>
<td>-</td>
<td>2,893.00</td>
<td>3,787.20</td>
<td>56.62</td>
<td>-10.6 %</td>
</tr>
<tr>
<td>space invaders</td>
<td>148.03</td>
<td>1,668.67</td>
<td>-</td>
<td>54,681.00</td>
<td>43,223.40</td>
<td><strong>74,335.30</strong></td>
<td>4,878.7 %</td>
</tr>
<tr>
<td>star gunner</td>
<td>664.00</td>
<td>10,250.00</td>
<td>-</td>
<td>434,343.00</td>
<td><strong>717,344.00</strong></td>
<td>549,271.70</td>
<td>5,723.0 %</td>
</tr>
<tr>
<td>surround</td>
<td>-9.99</td>
<td>6.53</td>
<td>-</td>
<td>7.00</td>
<td>9.90</td>
<td><strong>9.99</strong></td>
<td>120.9 %</td>
</tr>
<tr>
<td>tennis</td>
<td>-23.84</td>
<td>-8.27</td>
<td>-</td>
<td><strong>24.00</strong></td>
<td>-0.10</td>
<td>0.00</td>
<td>153.1 %</td>
</tr>
<tr>
<td>time pilot</td>
<td>3,568.00</td>
<td>5,229.10</td>
<td>-</td>
<td>87,085.00</td>
<td>445,377.30</td>
<td><strong>476,763.90</strong></td>
<td>28,486.9 %</td>
</tr>
<tr>
<td>tutankham</td>
<td>11.43</td>
<td>167.59</td>
<td>-</td>
<td>273.00</td>
<td>395.30</td>
<td><strong>491.48</strong></td>
<td>307.4 %</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><strong>715,545.61</strong></td>
<td>6,407.0 %</td>
</tr>
<tr>
<td>venture</td>
<td>0.00</td>
<td>1,187.50</td>
<td>-</td>
<td>1,813.00</td>
<td><strong>1,970.70</strong></td>
<td>0.40</td>
<td>0.0 %</td>
</tr>
<tr>
<td>video pinball</td>
<td>0.00</td>
<td>17,667.90</td>
<td>-</td>
<td>565,163.00</td>
<td><strong>999,383.20</strong></td>
<td>981,791.88</td>
<td>5,556.9 %</td>
</tr>
<tr>
<td>wizard of wor</td>
<td>563.50</td>
<td>4,756.52</td>
<td>-</td>
<td>46,204.00</td>
<td>144,362.70</td>
<td><strong>197,126.00</strong></td>
<td>4,687.9 %</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><strong>995,048.40</strong></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>-</td>
<td>42,286.00</td>
<td>224,910.70</td>
<td><strong>725,853.90</strong></td>
<td>7,940.5 %</td>
</tr>
</tbody>
</table>

# best

|          | 0   | 5   | 0   | 5   | 13  | 37  |

Table S1 of the paper “Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model” by Julian Schrittwieser et al.
In Nov 2018, an interesting paper from D. Hafner et al. proposed a **Deep Planning Network (PlaNet)**, which is a model-based agent that learns the MDP dynamics from pixels and then chooses actions using a CEM planner using a compact latent space.

The PlaNet is evaluated on selected tasks from the DeepMind control suite.

---

*Figure 1 of “Learning Latent Dynamics for Planning from Pixels”, https://arxiv.org/abs/1811.04551*
The partially observable MDPs are considered in PlaNet, that follow the stochastic dynamics:

- **transition function:** \( s_t \sim p(s_t | s_{t-1}, a_{t-1}) \),
- **observation function:** \( o_t \sim p(o_t | s_t) \),
- **reward function:** \( r_t \sim p(r_t | s_t) \),
- **policy:** \( a_t \sim p(a_t | o_{\leq t}, a_{<t}) \).

The main goal is to train the first three – the transition function, the observation function and the reward function.
Because an untrained agent will most likely not cover all needed environment states, we need to iteratively collect new experience and train the model. The authors propose $S = 5$, $C = 100$, $B = 50$, $L = 50$, $R$ between 2 and 8.

For planning, CEM algorithm capable of solving all tasks with a true model is used; $H = 12$, $I = 10$, $J = 1000$, $K = 100$. 

---

**Algorithm 1:** Deep Planning Network (PlaNet)

**Input:**
- $R$ Action repeat $p(s_t, a_{t-1}, \alpha_{t-1})$ Transition model
- $S$ Seed episodes $p(a_t | s_t)$ Observation model
- $C$ Collect interval $p(r_t | s_t)$ Reward model
- $B$ Batch size $q(s_t | o_{<t}, \alpha_{<t})$ Encoder
- $L$ Chunk length $\epsilon(t)$ Exploration noise
- $\alpha$ Learning rate

1. Initialize dataset $D$ with $S$ random seed episodes.
2. Initialize model parameters $\theta$ randomly.

while not converged do
   // Model fitting
   for update step $s = 1, C$ do
     Draw sequence chunks $\{(a_t, a_t, r_t)_{i=1}^{L+B} \} \sim D$ uniformly at random from the dataset.
     Compute loss $L(\theta)$ from Equation 3.
     Update model parameters $\theta \leftarrow \theta - \alpha \nabla_{\theta} L(\theta)$.

   // Data collection
   $o_1 \leftarrow$ env.reset()
   for time step $t = 1, \left[ \frac{T}{R} \right] \ do$
     Infer belief over current state $q(s_t | o_{<t}, a_{<t})$ from the history.
     $a_t \leftarrow$ planner $q(s_t | o_{<t}, a_{<t}, p)$, see Algorithm 2 in the appendix for details.
     Add exploration noise $\epsilon \sim p(\epsilon)$ to the action.
     for action repeat $k = 1, R$ do
       $r_t, o_{t+1} \leftarrow$ env.step($a_t$)
       $r_t, o_{t+1} \leftarrow$ env.step($a_t$)
     $D \leftarrow D \cup \{(a_t, a_t, r_t)_{i=1}^{T} \} \}$

**Algorithm 2:** Latent planning with CEM

**Input:**
- $H$ Planning horizon distance $q(s_t | o_{<t}, a_{<t})$ Current state belief
- $I$ Optimization iterations $p(a_t | s_{t-1}, a_{t-1})$ Transition model
- $J$ Candidates per iteration $p(r_t | s_t)$ Reward model
- $K$ Number of top candidates to fit

1. Initialize factorized belief over action sequences $q(a_{t+H}) \leftarrow \text{Normal}(0, 1)$.
2. for optimization iteration $i = 1..I$ do
   // Evaluate $J$ action sequences from the current belief.
   for candidate action sequence $j = 1..J$ do
     $a^{(j)}_{t+H} \sim q(a_{t+H})$
     $s^{(j)}_{t+H+1} \sim q(s_t | o_{t+1}, a_{t+H})$
     $R^{(j)} = \sum_{t+H}^T \mathbb{E}[p(r_t | s^{(j)}_t)]$
   // Re-fit belief to the $K$ best action sequences.
   $K \leftarrow \text{argsort}(\{ R^{(j)} \}_{j=1}^{J})$
   $\mu_{t+H} = \frac{1}{K} \sum_{k} (a^{(k)}_{t+H} - \mu_{t+H})$
   $\sigma_{t+H} = \sqrt{\frac{1}{K} \sum_{k} (a^{(k)}_{t+H} - \mu_{t+H})}$
   $q(a_{t+H}) \leftarrow \text{Normal}(\mu_{t+H}, \sigma_{t+H})$
3. return first action mean $\mu_t$.

---


First let us consider a typical latent-space model, consisting of

transition function: \( s_t \sim p(s_t | s_{t-1}, a_{t-1}) \),
observation function: \( o_t \sim p(o_t | s_t) \),
reward function: \( r_t \sim p(r_t | s_t) \).

The transition model is Gaussian with mean and variance predicted by a network, the observation model is Gaussian with identity covariance and mean predicted by a deconvolutional network, and the reward model is a scalar Gaussian with unit variance and mean predicted by a neural network.

To train such a model, we turn to variational inference and use an encoder

\[
q(s_{1:T} | o_{1:T}, a_{1:T}) = \prod_{t=1}^{T} q(s_t | s_{t-1}, a_{t-1}, o_t),
\]

which is a Gaussian with mean and variance predicted by a convolutional neural network.
PlaNet – Training Objective

Using the encoder, we obtain the following variational lower bound on the log-likelihood of the observations (for rewards the bound is analogous):

\[
\log p(o_{1:T} | a_{1:T}) = \log \int \prod_{t} p(s_t | s_{t-1}, a_{t-1}) p(o_t | s_t) \, ds_{1:T}
\]

\[
\geq \sum_{t=1}^{T} \left( \mathbb{E}_{q(s_t | o_{\leq t}, a_{<t})} \log p(o_t | s_t) - \mathbb{E}_{q(s_{t-1} | o_{\leq t-1}, a_{<t-1})} D_{KL} \left( q(s_t | o_{\leq t}, a_{<t}) \| p(s_t | s_{t-1}, a_{t-1}) \right) \right).
\]

We evaluate the expectations using a single sample and use the reparameterization trick to allow backpropagation through the sampling.
To derive the training objective, we employ importance sampling and the Jensen’s inequality:

\[
\log p(o_{1:T} | a_{1:T})
\]

\[
= \log \mathbb{E}_p(s_{1:T} | a_{1:T}) \prod_{t=1}^{T} p(o_t | s_t)
\]

\[
= \log \mathbb{E}_q(s_{1:T} | o_{1:T}, a_{1:T}) \prod_{t=1}^{T} p(o_t | s_t) p(s_t | s_{t-1}, a_{t-1}) / q(s_t | o_{\leq t}, a_{< t})
\]

\[
\geq \mathbb{E}_q(s_{1:T} | o_{1:T}, a_{1:T}) \sum_{t=1}^{T} \log p(o_t | s_t) + \log p(s_t | s_{t-1}, a_{t-1}) - \log q(s_t | o_{\leq t}, a_{< t})
\]

\[
= \sum_{t=1}^{T} \left( \mathbb{E}_q(s_t | o_{\leq t}, a_{< t}) \log p(o_t | s_t) - \mathbb{E}_q(s_{t-1} | o_{\leq t-1}, a_{< t-1}) D_{KL} \left( q(s_t | o_{\leq t}, a_{< t}) \| p(s_t | s_{t-1}, a_{t-1}) \right) \right).
\]
The purely stochastic transitions nevertheless struggle to store information for multiple timesteps. Therefore, the authors propose to include a deterministic path to the model, obtaining the **recurrent state-space model (RSSM)**:

**Figure 2 of “Learning Latent Dynamics for Planning from Pixels”, https://arxiv.org/abs/1811.04551**

- **Deterministic state model**: \( h_t = f(h_{t-1}, s_{t-1}, a_{t-1}) \),
- **Stochastic state function**: \( s_t \sim p(s_t|h_t) \),
- **Observation function**: \( o_t \sim p(o_t|h_t, s_t) \),
- **Reward function**: \( r_t \sim p(r_t|h_t, s_t) \),
- **Encoder**: \( q_t \sim q(s_t|h_t, o_t) \).
Table 1: Comparison of PlaNet to the model-free algorithms A3C and D4PG reported by Tassa et al. (2018). The training curves for these are shown as orange lines in Figure 4 and as solid green lines in Figure 6 in their paper. From these, we estimate the number of episodes that D4PG takes to achieve the final performance of PlaNet to estimate the data efficiency gain. We further include CEM planning ($H = 12, I = 10, J = 1000, K = 100$) with the true simulator instead of learned dynamics as an estimated upper bound on performance. Numbers indicate mean final performance over 5 seeds and 10 trajectories.

<table>
<thead>
<tr>
<th>Method</th>
<th>Modality</th>
<th>Episodes</th>
<th>Cartpole Swing Up</th>
<th>Reacher Easy</th>
<th>Cheetah Run</th>
<th>Finger Spin</th>
<th>Cup Catch</th>
<th>Walker Walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3C</td>
<td>proprioceptive</td>
<td>100,000</td>
<td>558</td>
<td>285</td>
<td>214</td>
<td>129</td>
<td>105</td>
<td>311</td>
</tr>
<tr>
<td>D4PG</td>
<td>pixels</td>
<td>100,000</td>
<td>862</td>
<td>967</td>
<td>524</td>
<td>985</td>
<td>980</td>
<td>968</td>
</tr>
<tr>
<td>PlaNet (ours)</td>
<td>pixels</td>
<td>1,000</td>
<td>821</td>
<td>832</td>
<td>662</td>
<td>700</td>
<td>930</td>
<td>951</td>
</tr>
<tr>
<td>CEM + true simulator</td>
<td>simulator state</td>
<td>0</td>
<td>850</td>
<td>964</td>
<td>656</td>
<td>825</td>
<td>993</td>
<td>994</td>
</tr>
<tr>
<td>Data efficiency gain PlaNet over D4PG (factor)</td>
<td></td>
<td>250</td>
<td>40</td>
<td>500+</td>
<td>300</td>
<td>100</td>
<td>90</td>
<td></td>
</tr>
</tbody>
</table>

*Table 1 of “Learning Latent Dynamics for Planning from Pixels”, https://arxiv.org/abs/1811.04551*
PlaNet – Ablations

Figure 4 of “Learning Latent Dynamics for Planning from Pixels”, https://arxiv.org/abs/1811.04551

- PlaNet (RSSM)
- Stochastic (SSM)
- Deterministic (GRU)
- D4PG (100k episodes)
- A3C (100k episodes, proprio)
Figure 5 of "Learning Latent Dynamics for Planning from Pixels", https://arxiv.org/abs/1811.04551