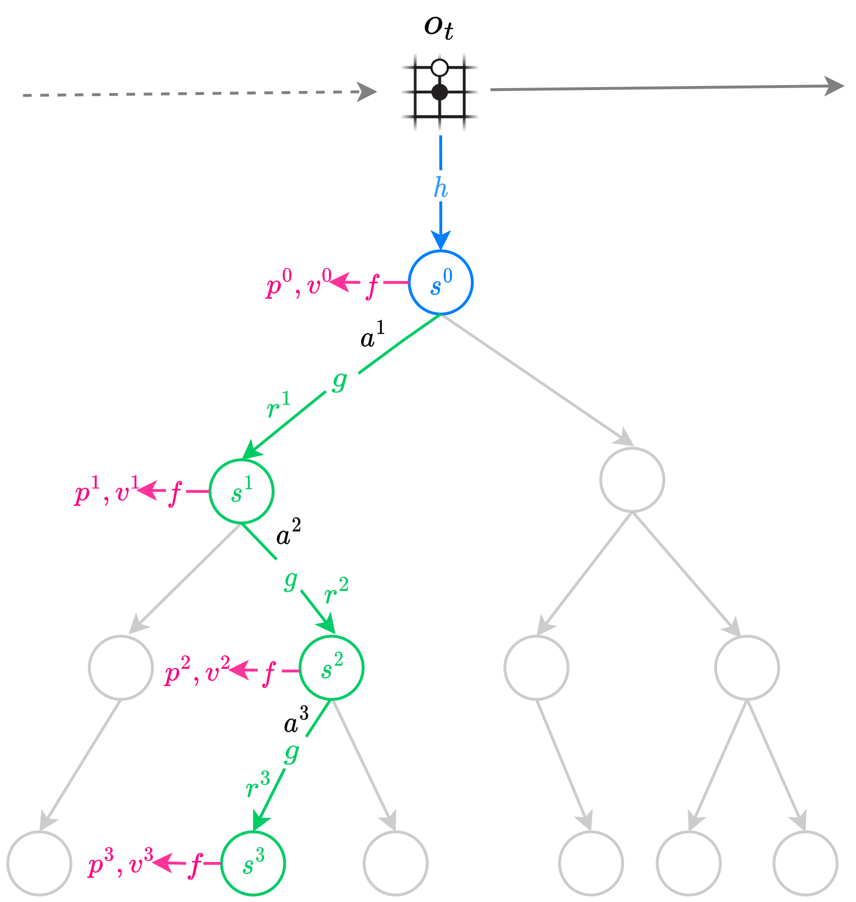
MuZero: Mastering Atari, Go, Chess and Shogi by Planninng with a Learned Model

A. Planing

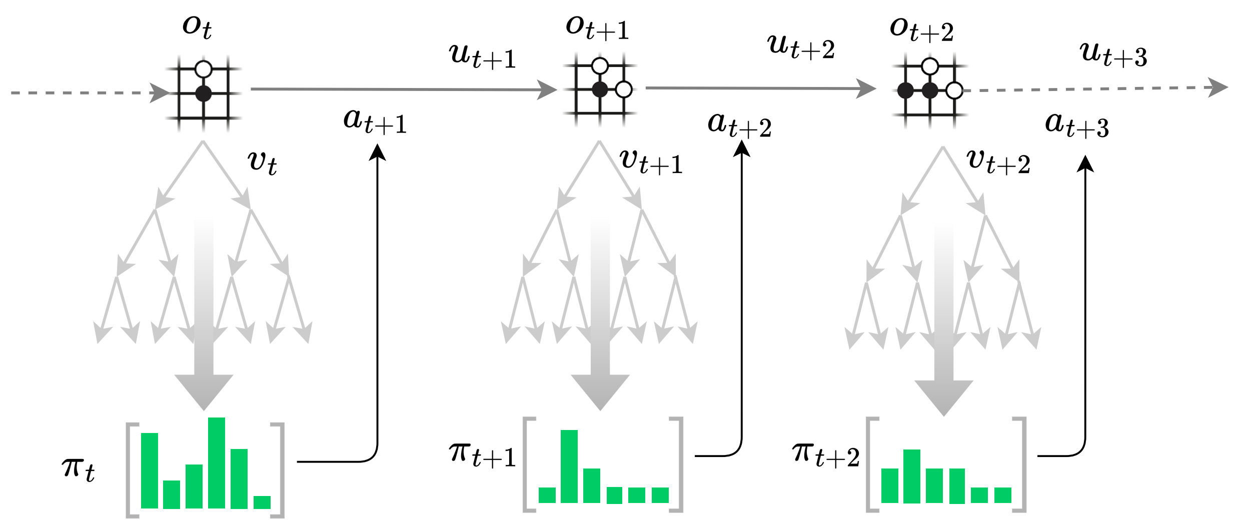


Selection:
In each node, agent select action a^k accordingto the UCB score:
$a^k = \arg\max_a \left[Q(s,a) + P(s,a)\cdot \frac{\sqrt{\sum_b N(s,b)}}{1+N(s,a)}\left(c_1 + \log\left(\frac{\sum_b N(s,b)+c_2+1}{c_2}\right)\right)\right]$

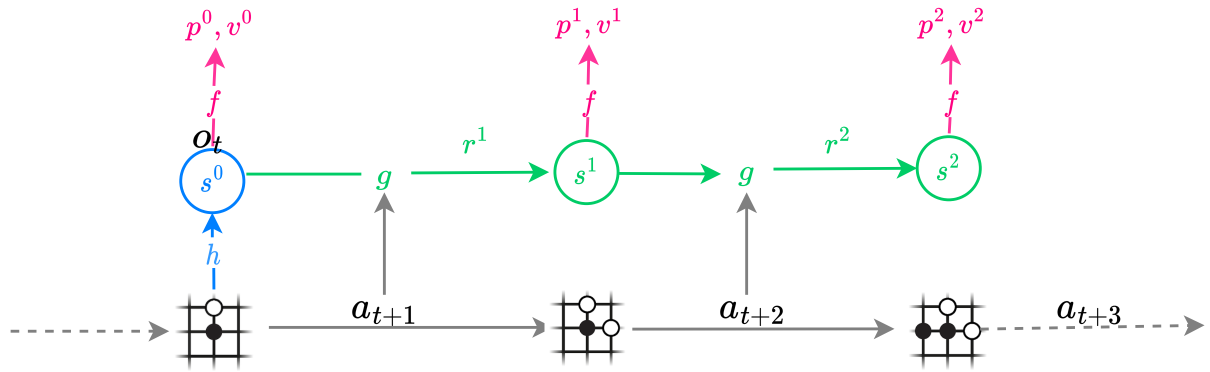
Expansion:
At the leaf node (i.e. final timestep l), the reward and hidden state are computed by the dynamics $r^l, s^l = g(^{l-1}, a^l)$ and stored in the corresponding tables. The policy and value are computed by the prediction function, $p^l, v^l = f(s^l)$.
A new node, corresponding to state s^l is added to the search tree. Each edge (s^l, a) from the newly expanded node is initialized to $\{N(s^l,a)=0, Q(s^l,a)=0, P(s^l,a)=\mathbf{p}^l\}$

Backup:
For $k = l \ldots 1$, we update the statistics for each edge (s^{k-1}, a^k) in the simulation path as follows,$Q(s^{k-1},a^k) := \frac{N(s^{k-1},a^k)\cdot Q(s^{k-1},a^k)+G^k}{N(s^{k-1},a^k)+1}$,
$N(s^{k-1},a^k) := N(s^{k-1},a^k)+1$,
where, in hypothetical step k, we utilize the $l-k$ bootstapped estimate Q value:
$G^k = \sum_{\tau=0}^{l-1-k}\gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l$

B. Acting



C. Training



D. Loss

$$l_t(\theta) = \sum_{k=0}^{K} l^r\left(u_{t+k}, r_t^k\right) + l^v\left(z_{t+k}, v_t^k\right) + l^p\left(\pi_{t+k}, \mathbf{p}_t^k\right) + c\|\theta\|^2$$

where, u_{t+k} is the observaed reward, π_{t+k} is the MCTS serched policy,z_{t+k} is the bootstrapped n-step target:
$z_{t+k} = u_{t+k+1} + \gamma u_{t+k+2} + \ldots + \gamma^{n-1} u_{t+k+n} + \gamma^n v_{t+k+n}$,
v_{t+k+n} is the MCTS serched value.
In Atari, l^r and l^v is cross-entopy loss, while in board games, l^v is MSE loss and there is no l^r due to no intermediate reward. l^p is cross-entopy loss for both.

NOTE： g is the dynamic network (MLP, without RNN), value rescale, categorical distribution for reward and value in Atari, Reanalyze