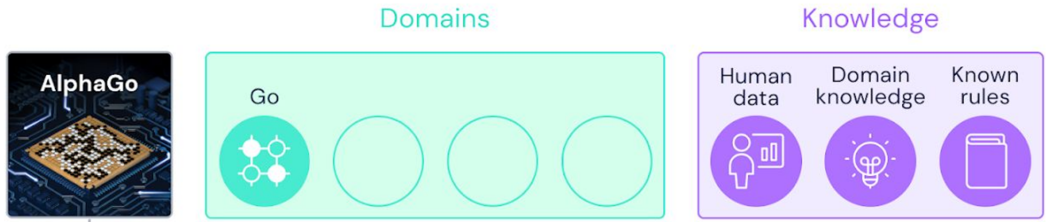
MuZero

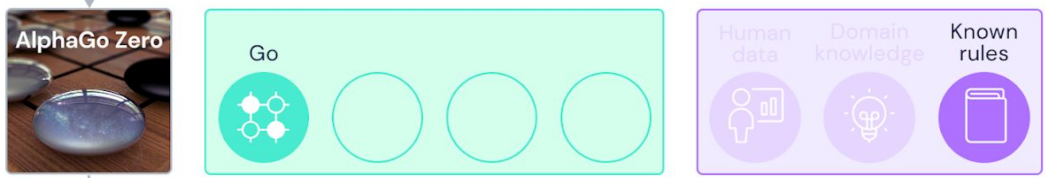
Omkar Ranadive

Schrittwieser, Julian, et al. "Mastering atari, go, chess and shogi by planning with a learned model." *Nature* 588.7839 (2020)

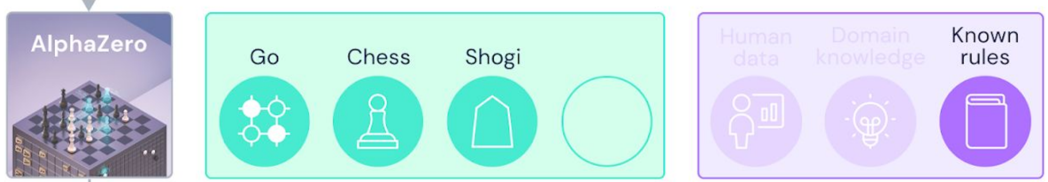
Northwestern



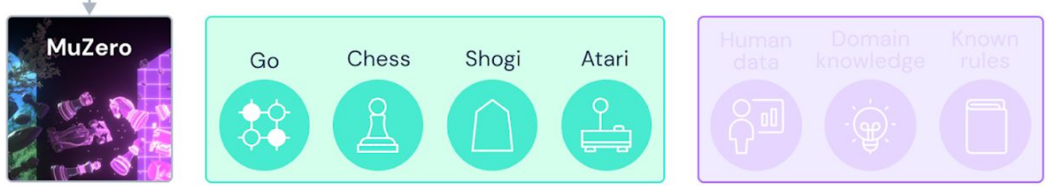
AlphaGo becomes the first program to master Go using neural networks and tree search
(Jan 2016, Nature)



AlphaGo Zero learns to play completely on its own, without human knowledge
(Oct 2017, Nature)



AlphaZero masters three perfect information games using a single algorithm for all games
(Dec 2018, Science)



MuZero learns the rules of the game, allowing it to also master environments with unknown dynamics.
(Dec 2020, Nature)

How is MuZero different?

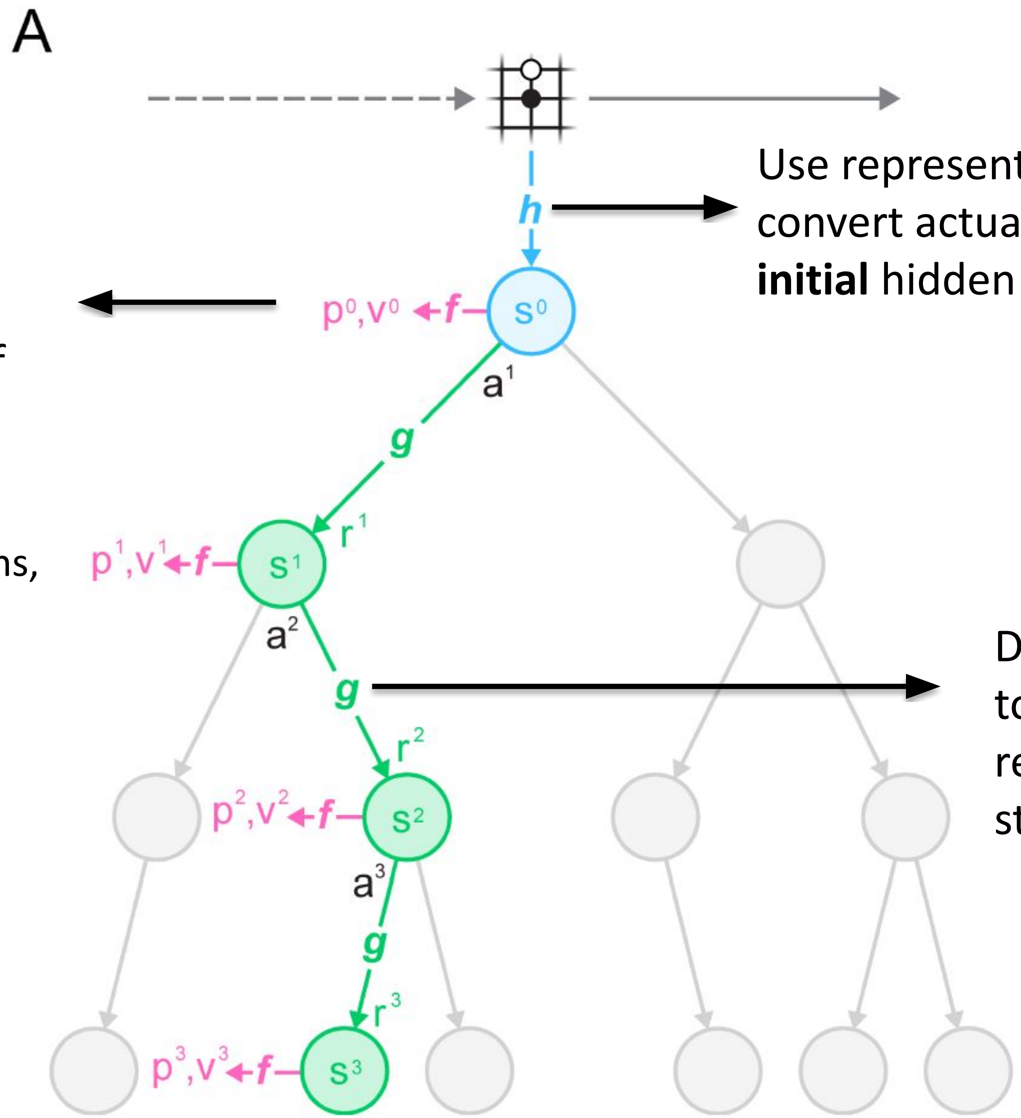
- Planning algorithms require knowledge of environment dynamics
- Model based RL tries to solve this by learning the environment
- Model based RL doesn’t work for complex environments – Ex Atari
- Model free does great in complex environments but doesn’t work well in cases which require planning – Ex – Go, Shogi

How is MuZero different?

MuZero learns a model of the environment and combines planning with it

Main idea: Instead of learning a complex model, only learn those aspects which are relevant for planning i.e., value, policy, reward

The MuZero Algorithm



Use representation function h to convert actual observation to **initial** hidden state s^0

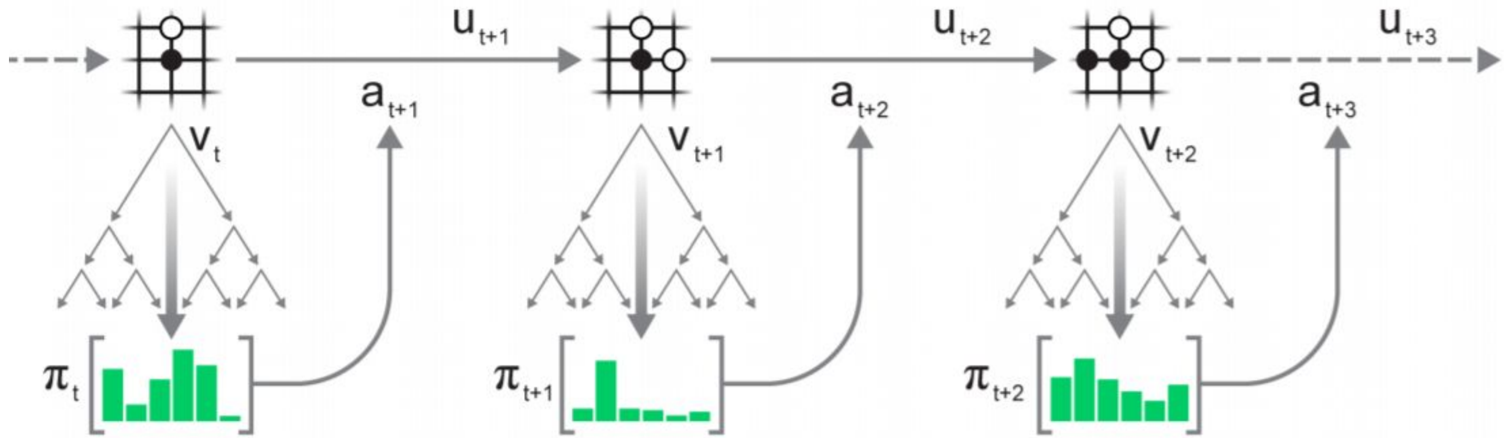
Calculate policy and value function using prediction function f

Where policy = distribution over actions, Value function = expected reward

Dynamics function to get immediate reward and next state

Key idea: Tree search is being done over those hidden states s^k and is guided by those different functions (neural networks)

Monte Carlo Tree Search (MCTS)



The actual action a is taken based on the search policy pi (not the same as function p)

Monte Carlo Tree Search (MCTS)

Comes from value function v

$$a^k = \arg\max_a \left[Q(s,a) + P(s,a) \cdot \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)} \left(c_1 + \log\left(\frac{\sum_b N(s,b) + c_2 + 1}{c_2} \right) \right) \right]$$

This prior comes from the policy function p

Visitation count

- Basically, the search is guided by these values predicted by neural networks
- This information is maintained for each (s,a) pair, i.e., edge in a tree
- 800 simulations for board games and 50 simulations for Atari games in each search

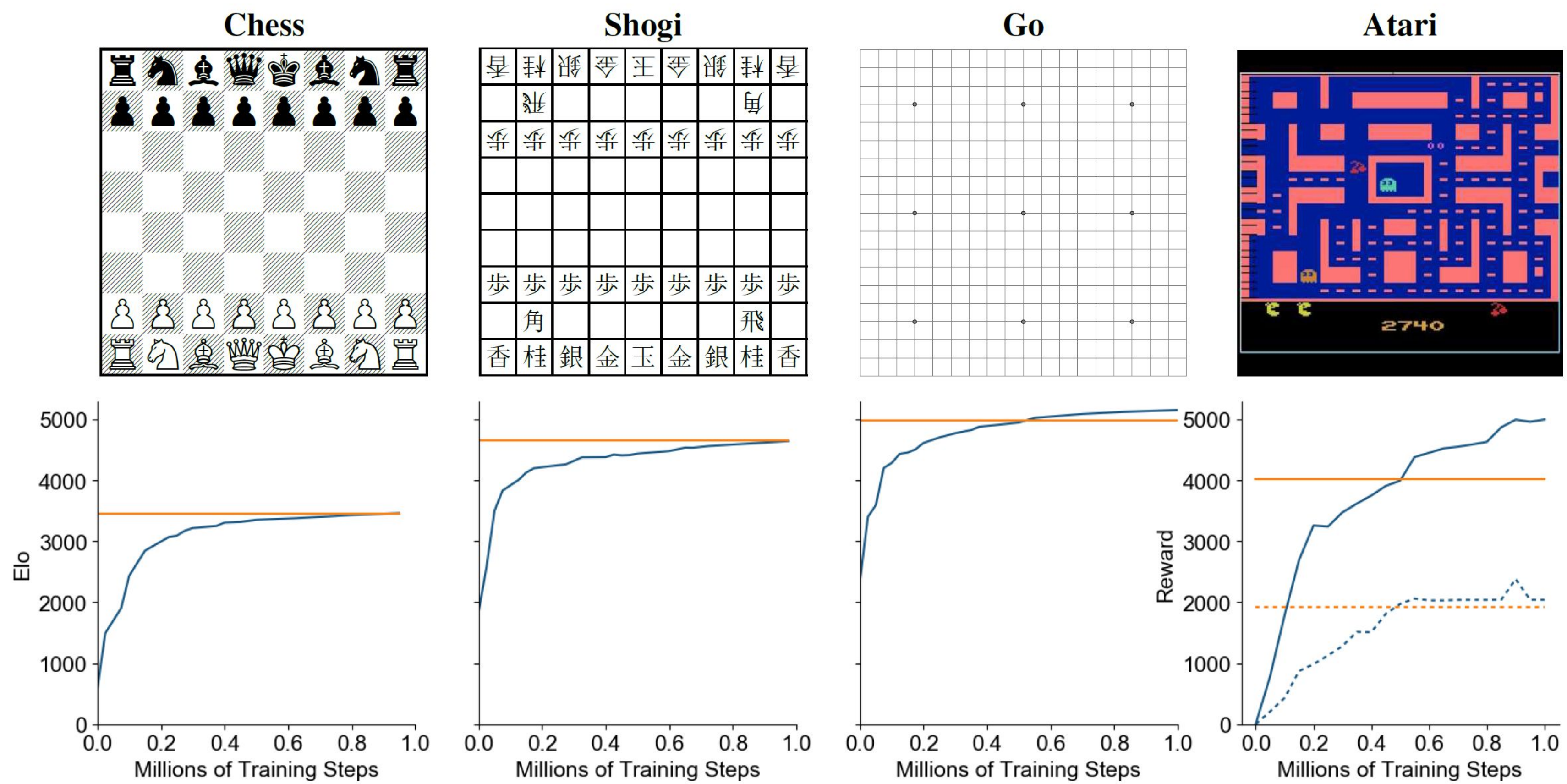
Loss function

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

Minimize immediate reward and expected reward

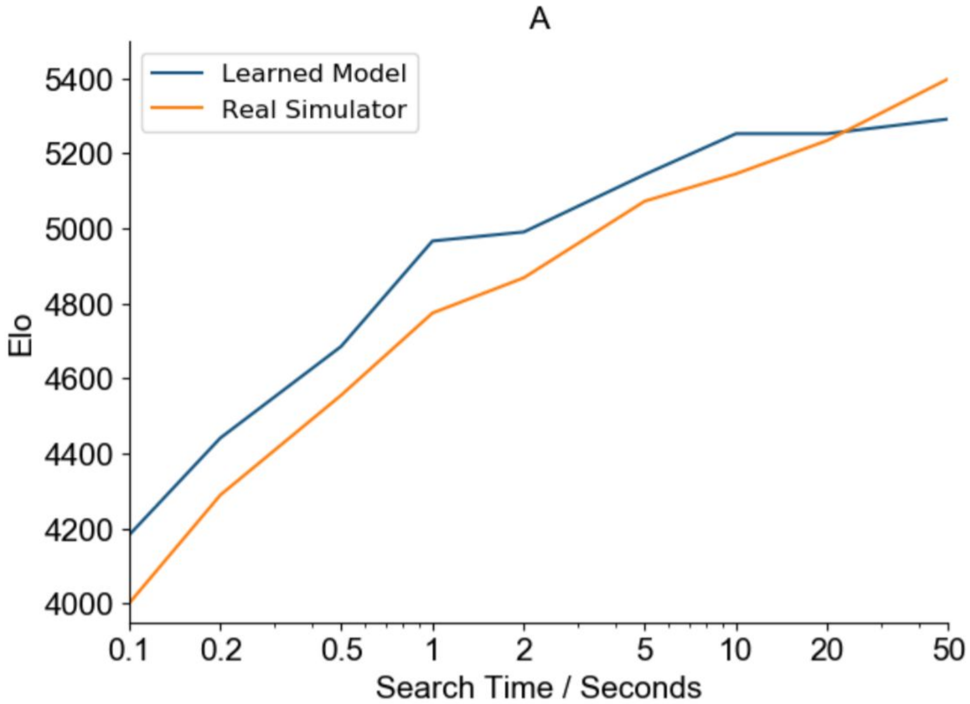
Minimize difference between predicted policy p and search policy (MCTS) - So they make each other better (as MCTS is guided by p)

Results



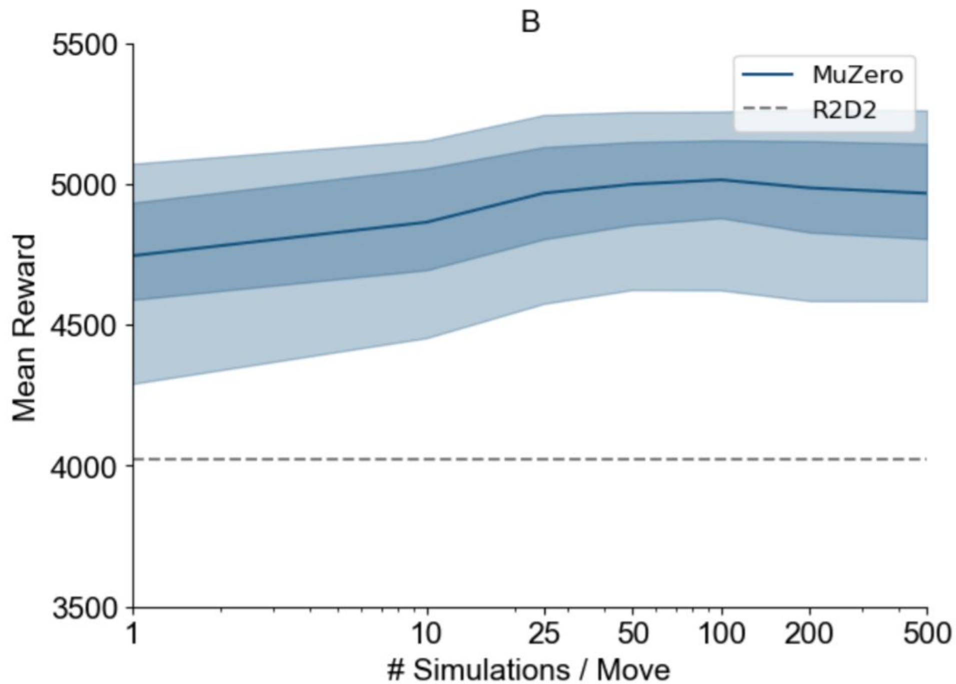
Chess
Shogi
Go
Atari
Elo
Reward
0
1000
2000
3000
4000
5000
0.0
0.2
0.4
0.6
0.8
1.0
Millions of Training Steps

Results



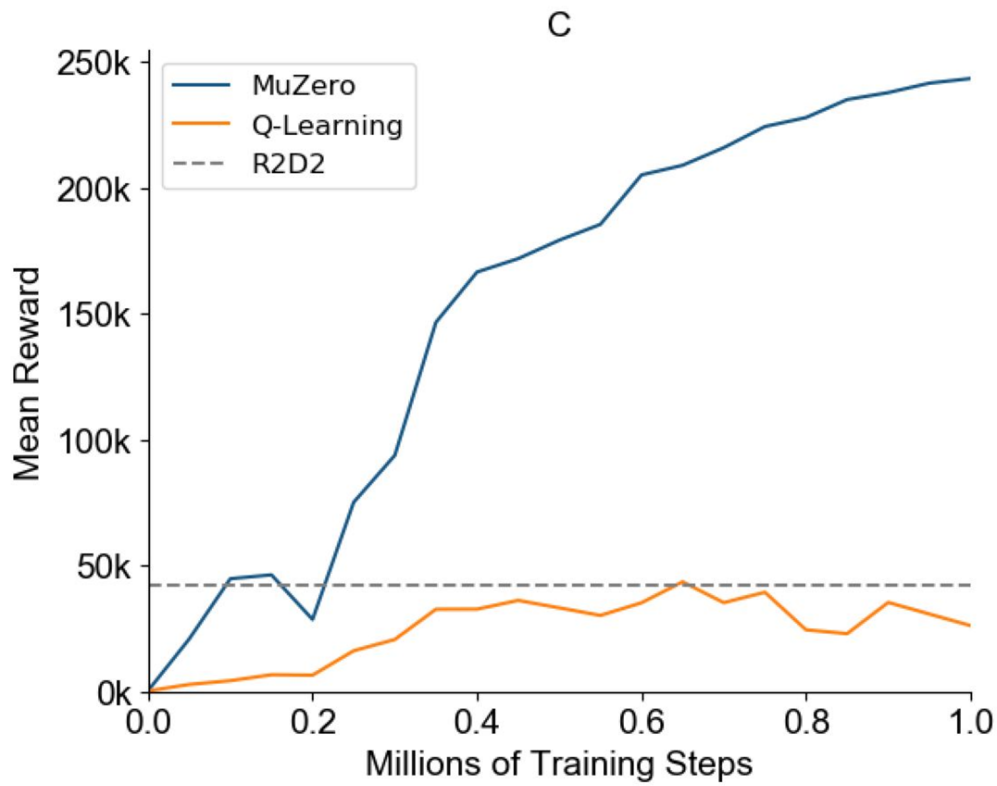
MuZero trained on 0.1s (800 simulations) can search deeper up to two orders of magnitude larger

Results



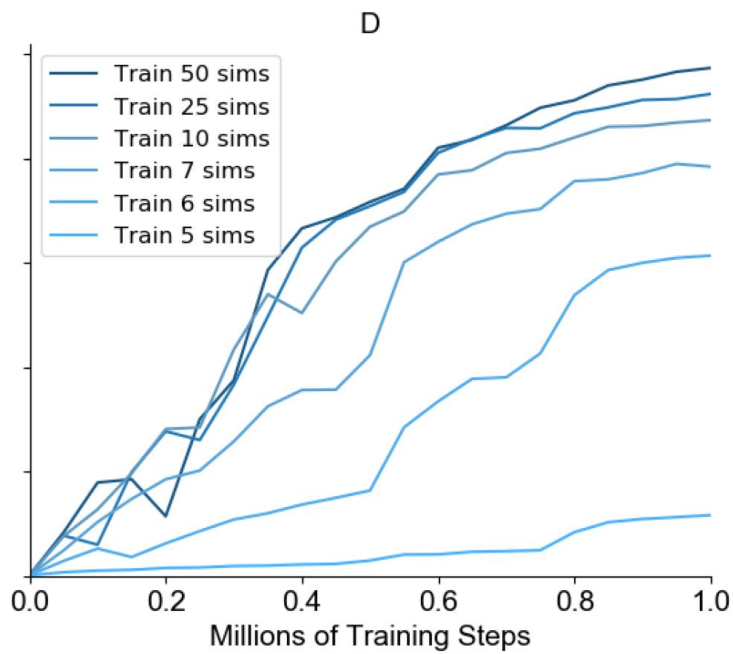
- Increasing simulations doesn't improve score by much in Atari games (plateau at 100)
- Also, the score is high even with single simulation (policy absorbs the planning)

Results



Search based planning of MuZero peforms much better and learns faster than its Q-learning counterpart

Results



- Trained with different simulation values and evaluated at 50 simulations
- Interesting result – Even when trained on 6 sims (sims < actions (8) in Ms. Pacman game) it performs very well