

# Reinforcement Learning

---

## Exercise 11: MCTS + MBRL

22.07.2025

Alexander Mattick

# Exercise Sheet 8

MCTS



# 4 Views on RL

## Why and How

---

### Dynamic Programming

- Table of State-Actions
- Recursion
- Try to find fixed point

Guarantees:  
Tabular: Easy  
Function Approx:  
None/Good luck

### Probabilistic Inference

- Try to find a distribution that represents optimal choices
- KL constraints to ensure stable information gain
- Naturally a variational inference problem

Guarantees:  
Sufficiently powerful models will eventually converge

### Decision Theory

- Find optimal action amongst set
- Maximize utility/minimize Regret

Guarantees:  
Convergence is well understood + exact guarantees are possible!

### Model Based

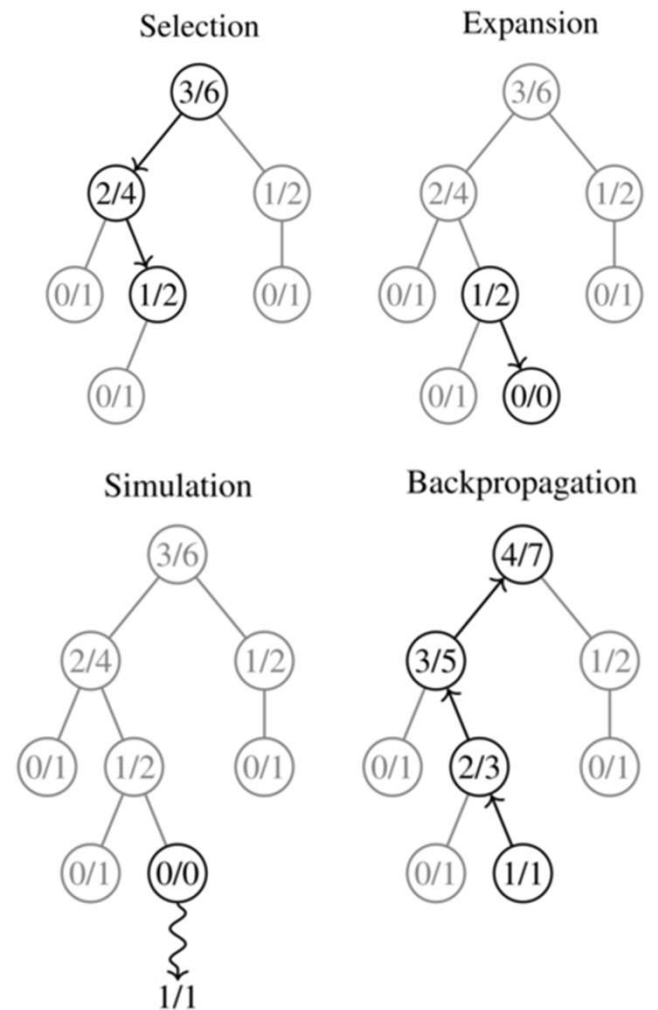
- Build a simulator
- Solve a planning problem using the simulator

Guarantees:  
Depending on Model:  
Convergence + exact guarantees are possible!

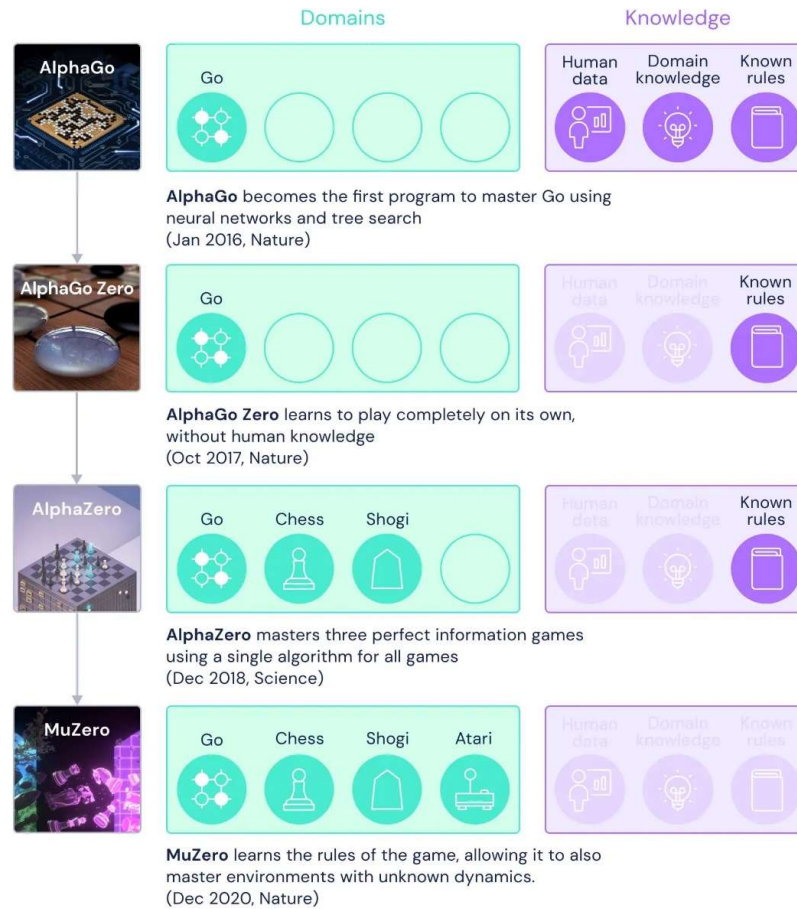
# Monte Carlo Tree Search

- Heuristic search algorithm using random sampling for (deterministic) problems
  - In our setting: Nodes are states, edges are actions
- Play many rollouts from the root node
  - **Selection:** Select successive child nodes until a leaf node is reached
  - **Expansion:** Create a new child node
  - **Simulation:** Continue with (random) actions until the terminal state
  - **Backpropagation:** Update information in the nodes on the path traversed
- Balancing exploitation and exploration during expansion via **UCT** formula

$$a = \operatorname{argmax}_i \frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$$



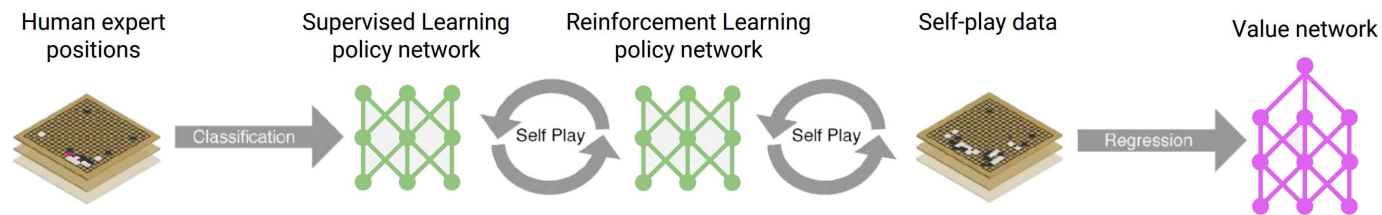
# The Evolution of AlphaGo to muZero



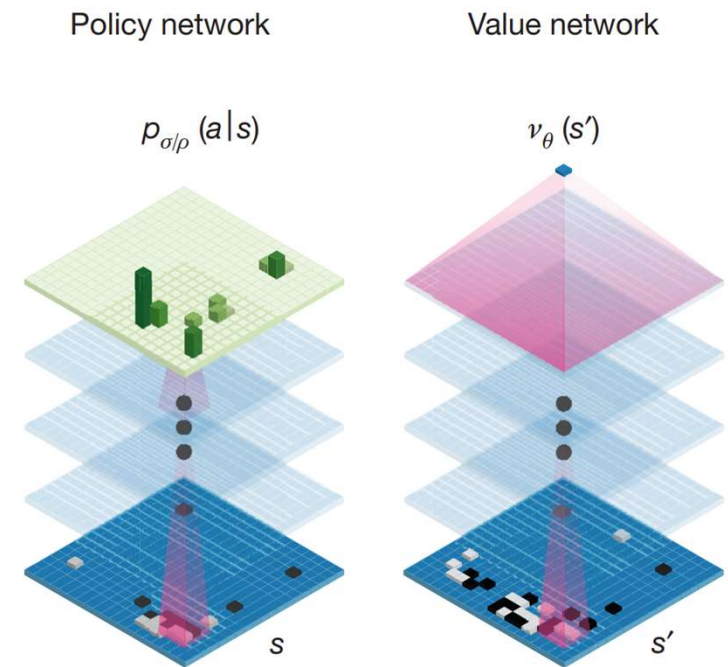
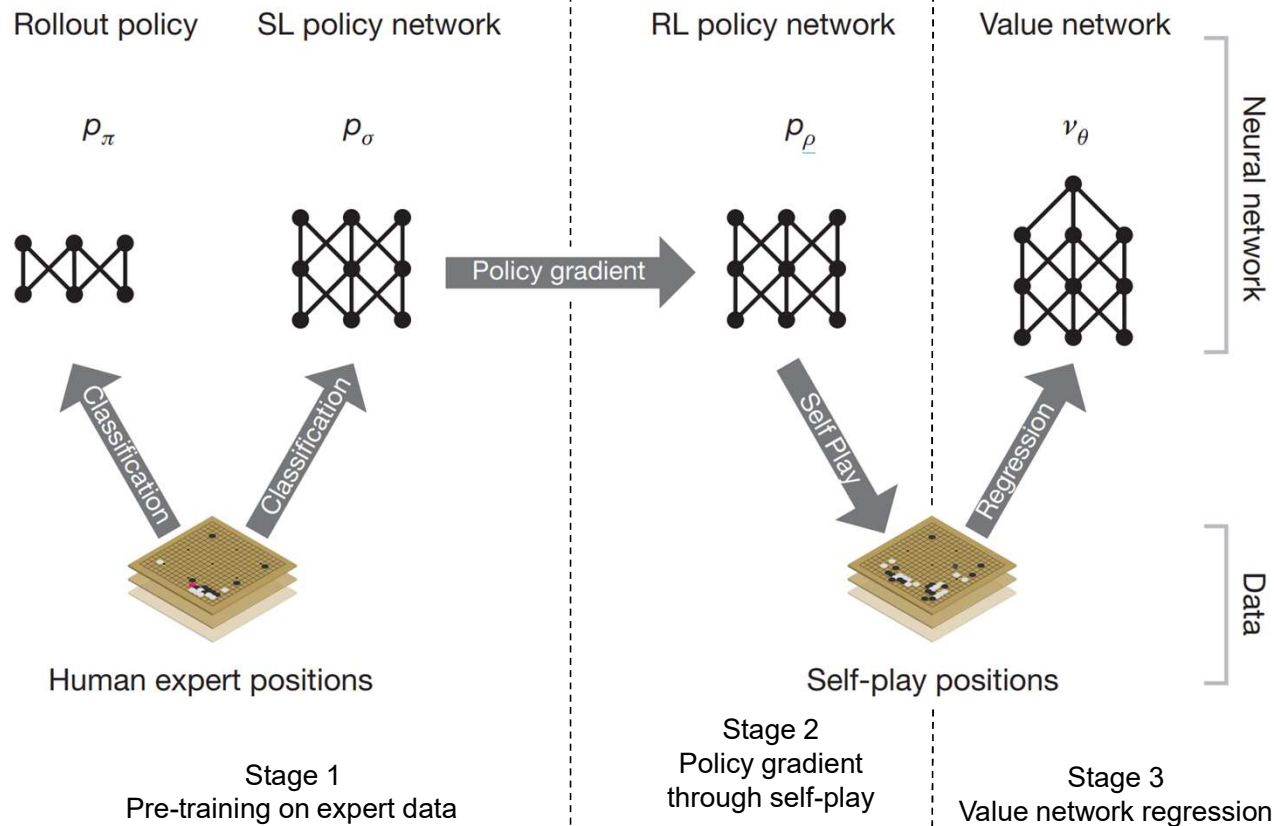
<https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules>

# AlphaGo

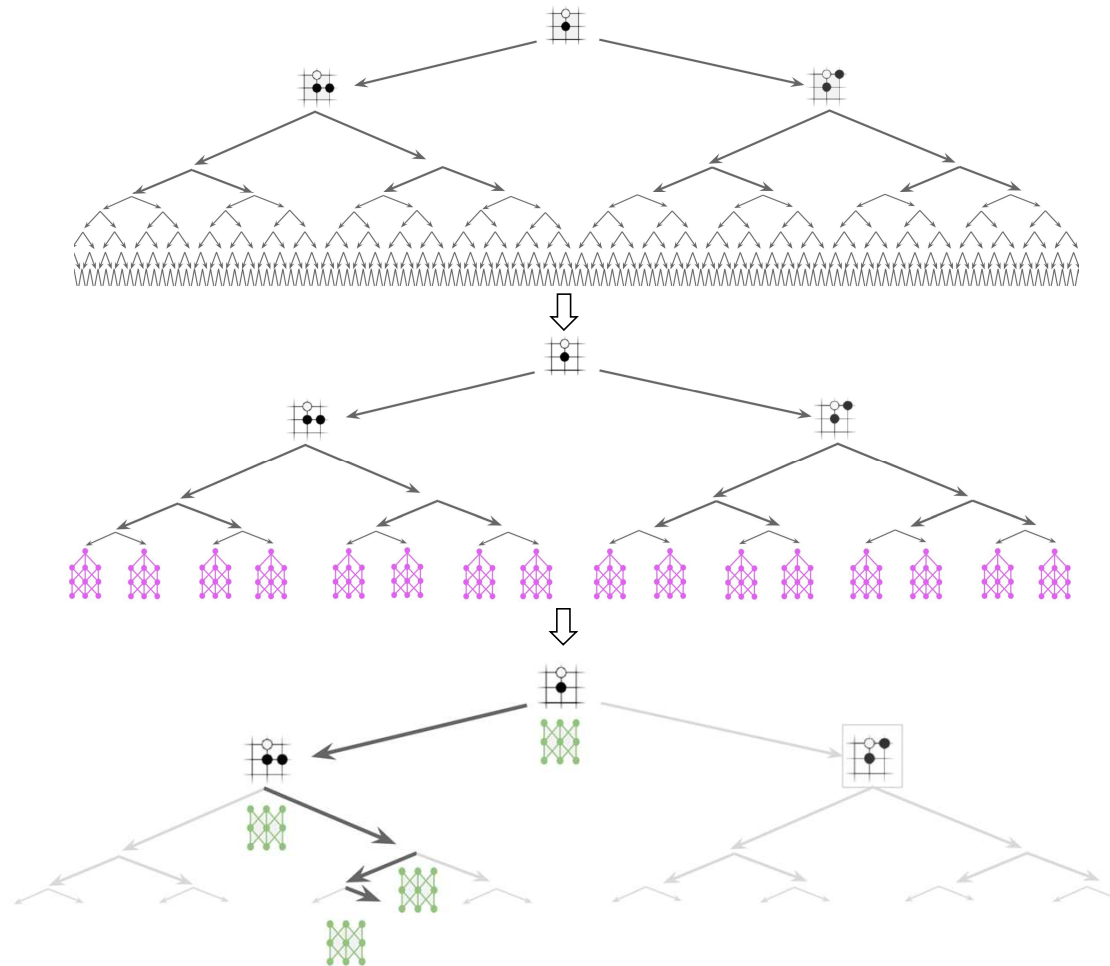
- **AlphaGo** defeated the Go champion Lee Sedol in a best-of-five tournament in 2016
- Algorithm outline
  - **Training**
    - A policy  $p(s|a)$  is trained to predict human expert moves in a data set of positions, refined via policy gradient through self-play, and training of value regressor on self-play data
  - **Deployment**
    - MCTS with policy and value network



# AlphaGo – Training



# AlphaGo – Influences on Search Complexity



Exhaustive search

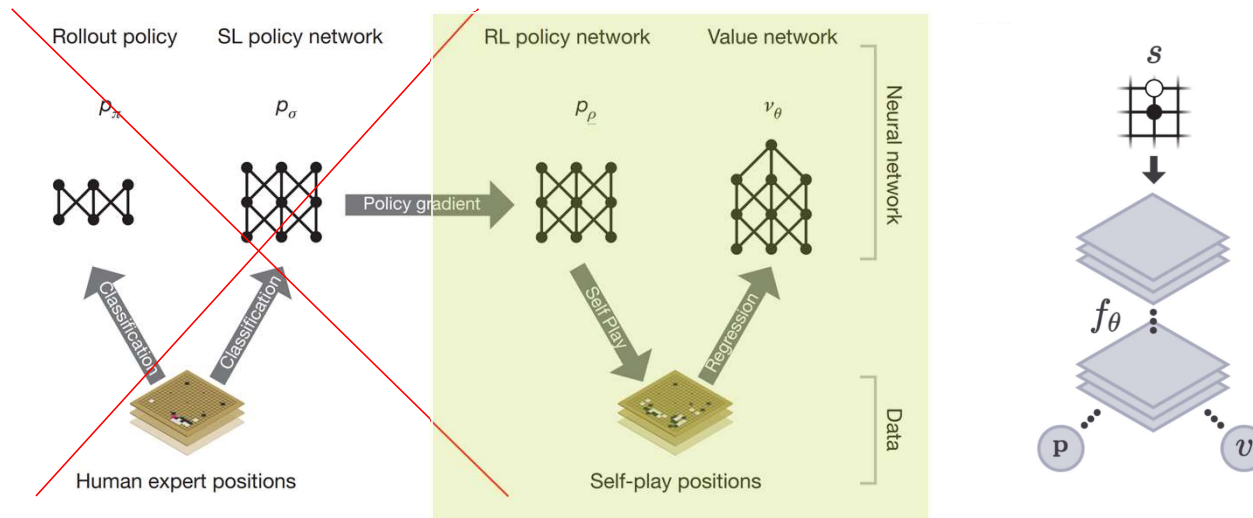
Reducing depth with value network

Reducing breath with policy network

[https://www.davidsilver.uk/wp-content/uploads/2020/03/AlphaGo-tutorial-slides\\_compressed.pdf](https://www.davidsilver.uk/wp-content/uploads/2020/03/AlphaGo-tutorial-slides_compressed.pdf)

# AlphaZero: Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm

- Published one year after AlphaGo in 2017
- Achieved superhuman level of play in the games of Chess, shogi, and Go within 24 hours of training
- Main goal: Replace handcrafted knowledge and domain-specific augmentations
  - Also: Reduction to one neural network + MCTS already during training via self-play

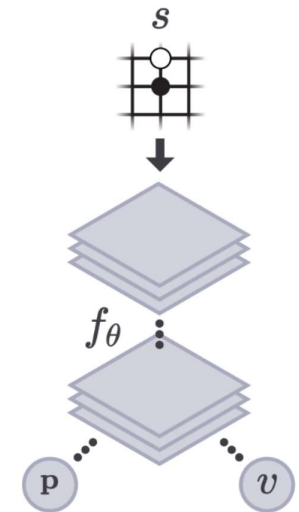


# AlphaZero

- One deep neural network  $f_{\theta}(s) = (p, v)$  with
  - move probabilities  $p = \Pr(a|s)$  and
  - value prediction  $v$  (win probability of the current player)
- “*Tabula rasa*” reinforcement learning
  - A policy plays against a past version of itself (self-play)
  - In each position, an MCTS search is executed
    - Guided by the neural network’s move probabilities  $p$
    - More robust, sophisticated policy (tree-search informed by policy network’s “best guess”)
  - Network is updated towards MCTS move probabilities (policy head) and self-play winner outcome (value head)
- “Policy iteration procedure”

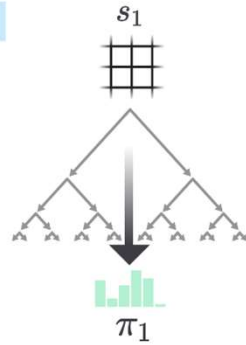
“policy evaluation”

“policy improvement”



# AlphaZero - Method

a. Self-Play



# AlphaZero - Results

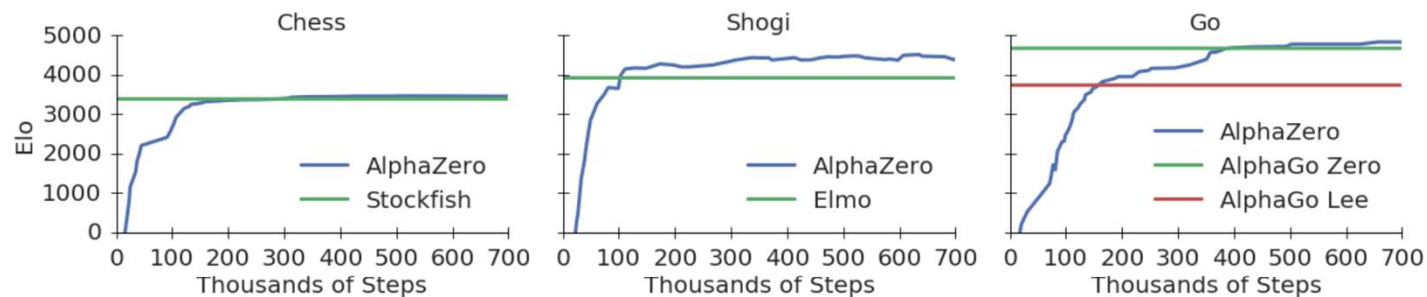


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared to 2017 CSA world-champion program *Elmo*. **c** Performance of *AlphaZero* in Go, compared to *AlphaGo Lee* and *AlphaGo Zero* (20 block / 3 day) (29).



# muZero: Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

---

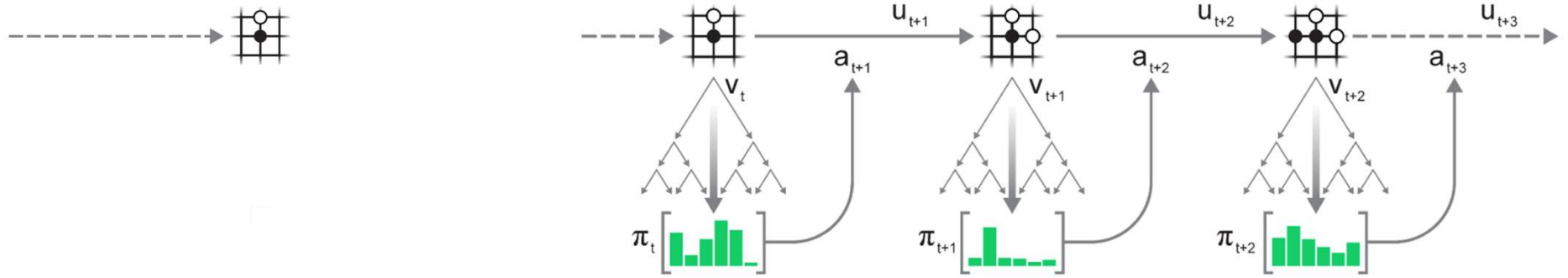
- **AlphaZero** transferred to settings without a perfect simulator
  - Remember: MCTS performs multiple rollouts, for which we must query a simulator
  - Also: **muZero** generalizes to single agent domains and with intermediate rewards settings
- Instance of **model-based RL**
- Apart from board games, achieved new state-of-the-art performance on the Atari benchmark

## muZero - Method

---

- Consists of three function approximators
  - **Dynamics function:**  $g_{\theta}(s^{k-1}, a^k) = r^k, s^k$ 
    - Recurrent process that computes, at hypothetical step  $k$ , an immediate reward  $r^k$  and internal state  $s^k$ 
      - Unlike traditional approaches to model-based RL,  $s^k$  has no semantic meaning attached
    - Deterministic
  - **Prediction function:**  $f_{\theta}(s^k) = p^k, v^k$ 
    - Analogous to AlphaGo or AlphaZero, but computed from internal state rather than “world state”
  - **Representation function:**  $h_{\theta}(o_1, \dots, o_t) = s^0$ 
    - Encodes past observations into “root” state
- Given such a model, it is possible to search over hypothetical future trajectories  $a^1, \dots, a^k$  given past observations

# muZero – Planning using the Model



## muZero - Training

---

- Compared to past methods, representation and dynamics function also must be trained
  - Place into rollout buffer:
    - All predictions, i.e.,  $s^{k+1}, r^k, p^k, v^k$
    - Actual reward  $u_{t+k}$ , value  $z_{t+k}$  and MCTS policy  $\pi_{t+k}$
  - Train end to end

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

- All experiments used 5 unrolling steps into the future

# muZero - Results

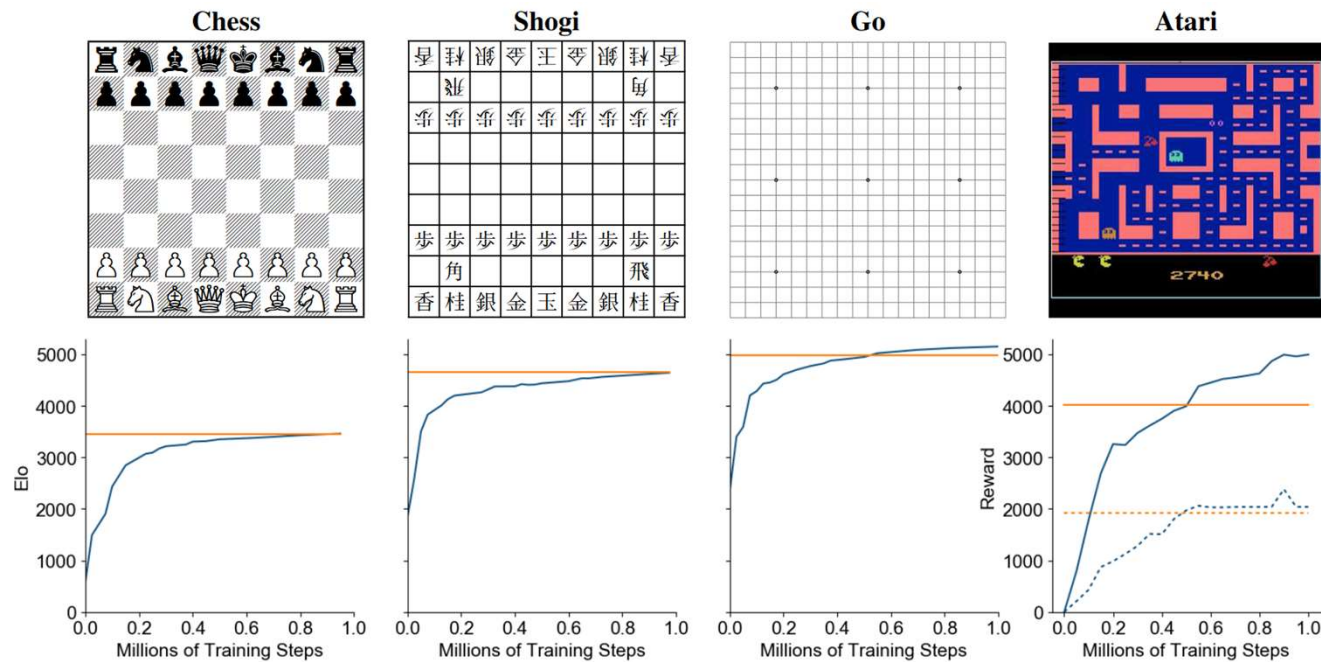


Figure 2: **Evaluation of MuZero throughout training in chess, shogi, Go and Atari.** The x-axis shows millions of training steps. For chess, shogi and Go, the y-axis shows Elo rating, established by playing games against *AlphaZero* using 800 simulations per move for both players. *MuZero*'s Elo is indicated by the blue line, *AlphaZero*'s Elo by the horizontal orange line. For Atari, mean (full line) and median (dashed line) human normalized scores across all 57 games are shown on the y-axis. The scores for R2D2 [21], (the previous state of the art in this domain, based on model-free RL) are indicated by the horizontal orange lines. Performance in Atari was evaluated using 50 simulations every fourth time-step, and then repeating the chosen action four times, as in prior work [23].

## muZero - Results

Agent	Median	Mean	Env. Frames	Training Time	Training Steps
Ape-X [18]	434.1%	1695.6%	22.8B	5 days	8.64M
R2D2 [21]	1920.6%	4024.9%	37.5B	5 days	2.16M
<i>MuZero</i>	<b>2041.1%</b>	<b>4999.2%</b>	20.0B	12 hours	1M
IMPALA [9]	191.8%	957.6%	200M	–	–
Rainbow [17]	231.1%	–	200M	10 days	–
UNREAL <sup>a</sup> [19]	250% <sup>a</sup>	880% <sup>a</sup>	250M	–	–
LASER [36]	431%	–	200M	–	–
<i>MuZero Reanalyze</i>	<b>731.1%</b>	<b>2168.9%</b>	200M	12 hours	1M

Table 1: **Comparison of *MuZero* against previous agents in Atari.** We compare separately against agents trained in large (top) and small (bottom) data settings; all agents other than *MuZero* used model-free RL techniques. Mean and median scores are given, compared to human testers. The best results are highlighted in **bold**. *MuZero* sets a new state of the art in both settings. <sup>a</sup>Hyper-parameters were tuned per game.



Fraunhofer-Institut für Integrierte  
Schaltungen IIS

**Thank you for your attention!**