Reinforcement Learning

—

Exercise 9: MCTS + MBRL

22.06.2023

Sebastian Rietsch

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS

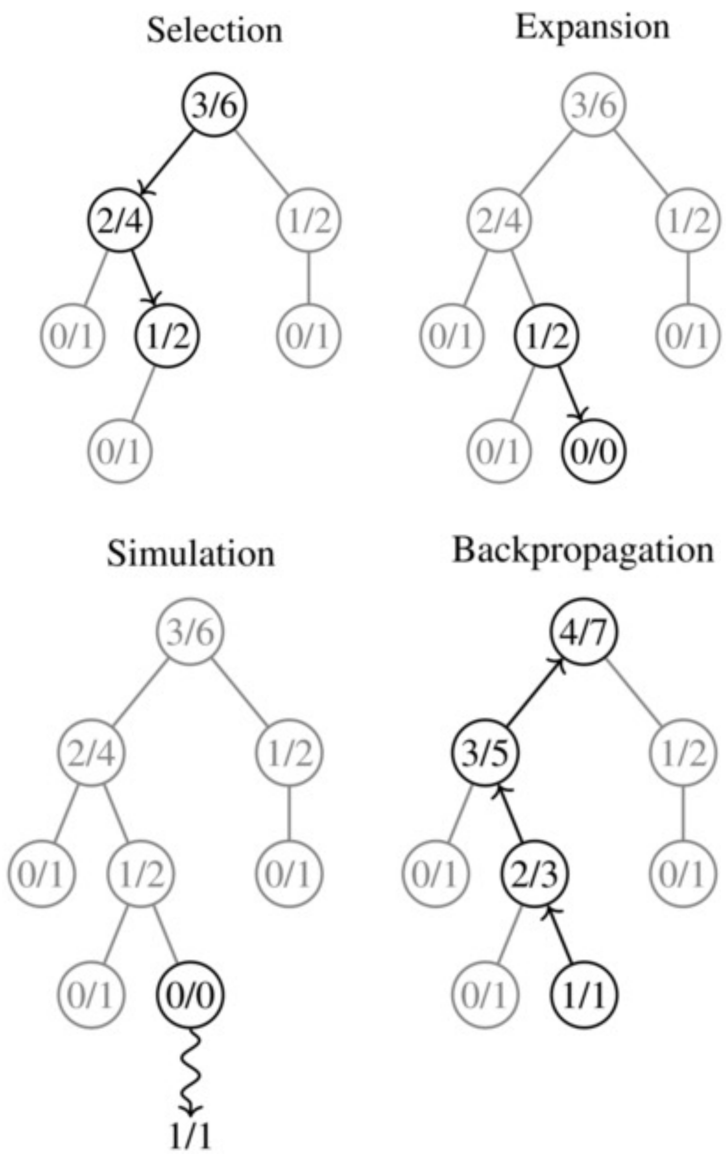
Exercise Sheet 8

MCTS

MCTS (continued)

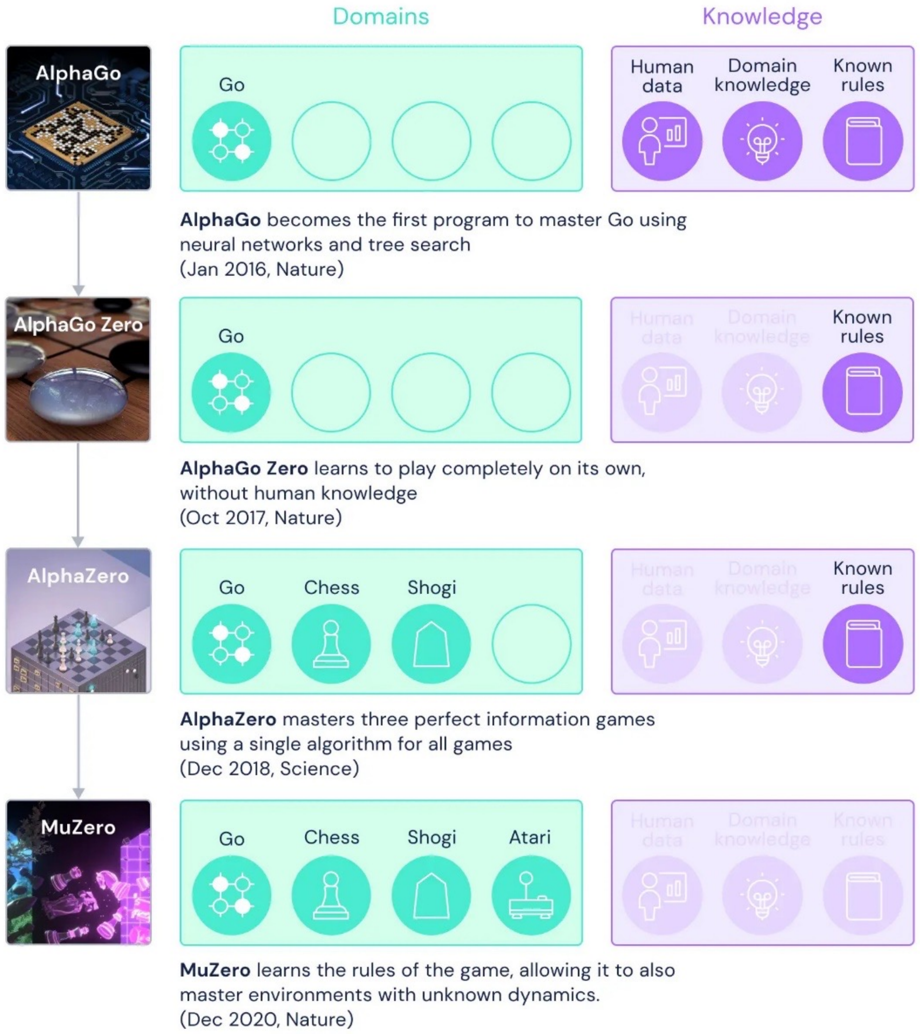
Monte Carlo Tree Search

- **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
  - $a = argmax_i \frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$



Fraunhofer IIS

The Evolution of AlphaGo to muZero

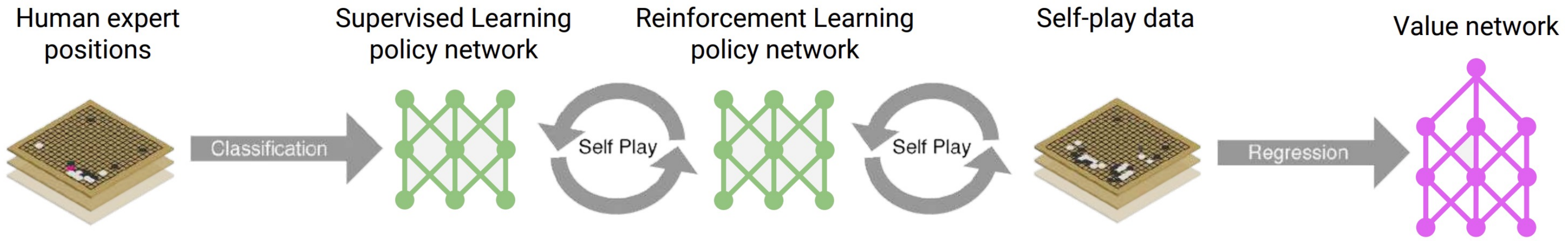


https://www.deepmind.com/blog/muzero-mastering-go-chess-shogi-and-atari-without-rules

Fraunhofer IIS

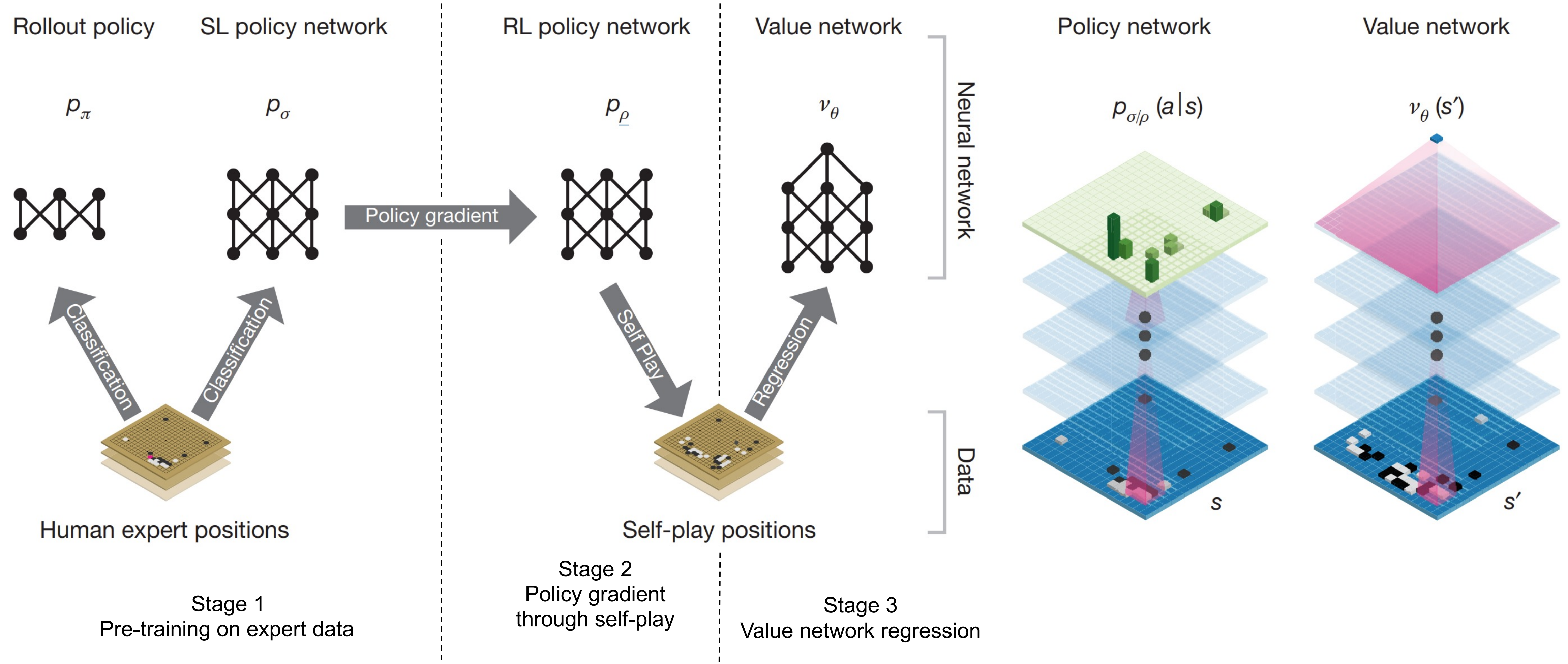
AlphaGo

- **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

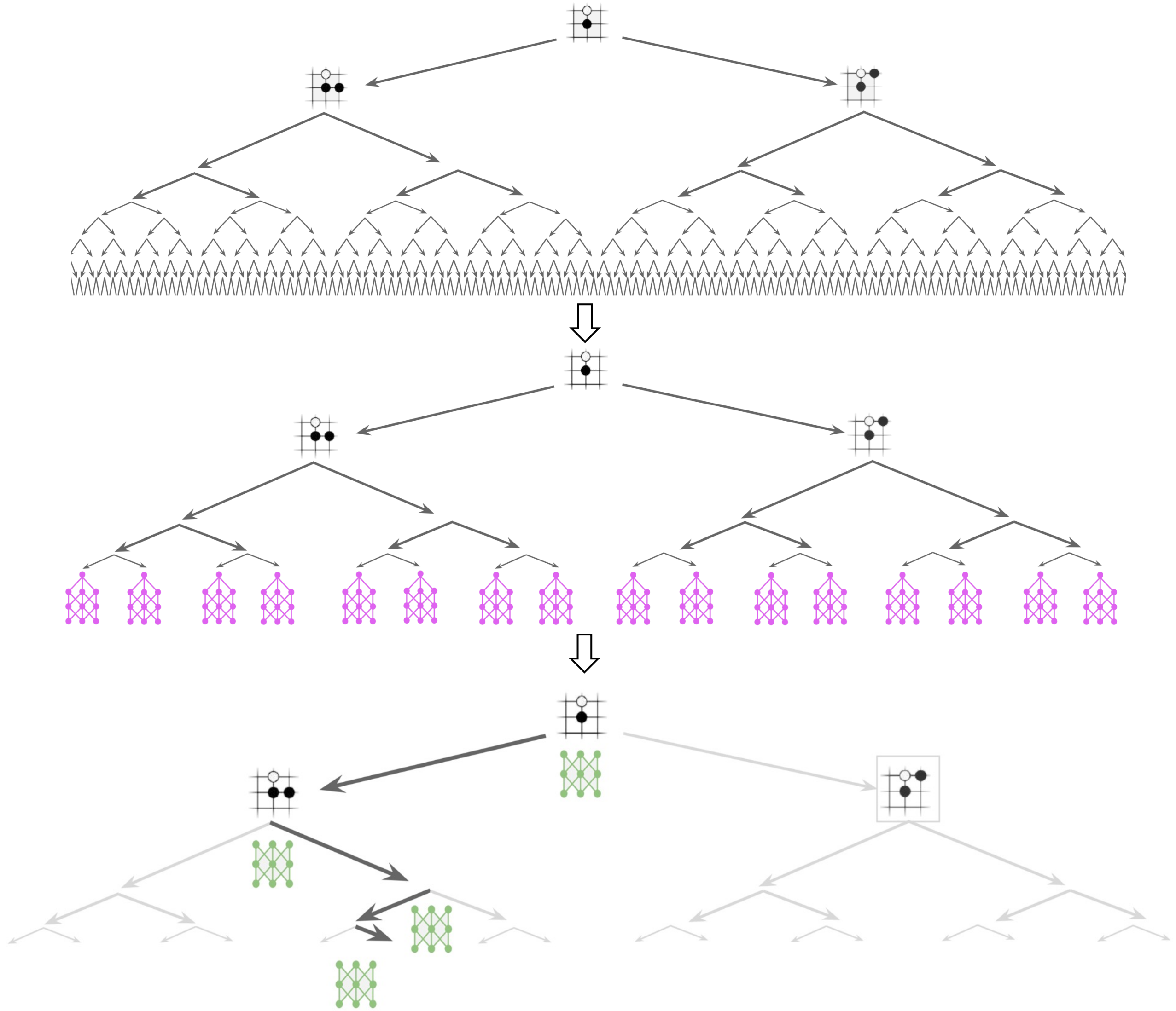


Fraunhofer IIS

AlphaGo – Training



AlphaGo – Influences on Search Complexity



Exhaustive search

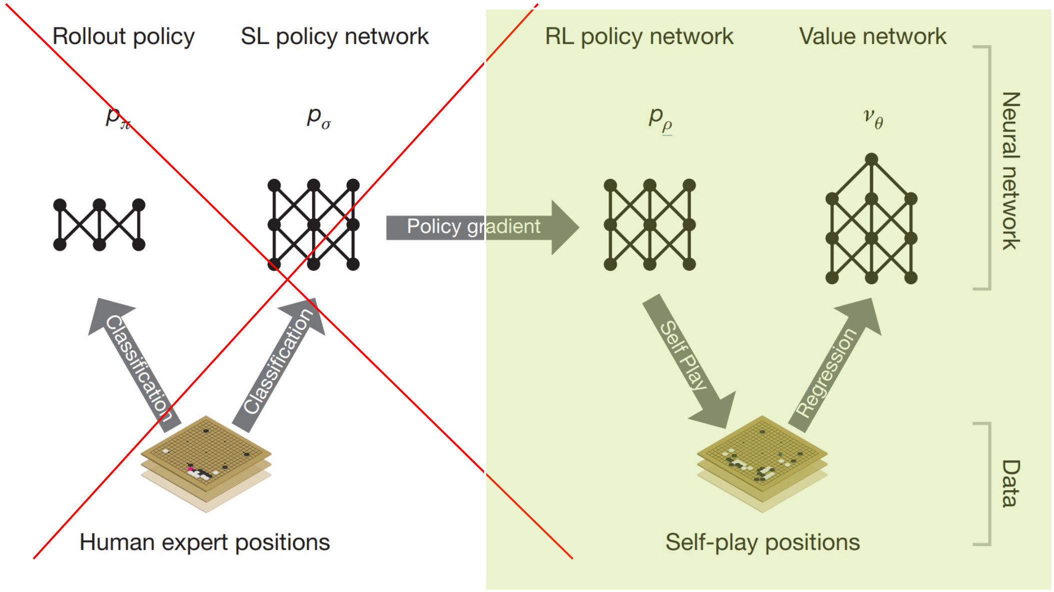
Reducing depth with value network

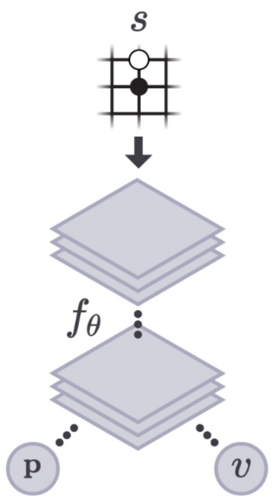
Reducing breath with policy network

https://www.davidsilver.uk/wp-content/uploads/2020/03/AlphaGo-tutorial-slides_compressed.pdf

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





Fraunhofer IIS

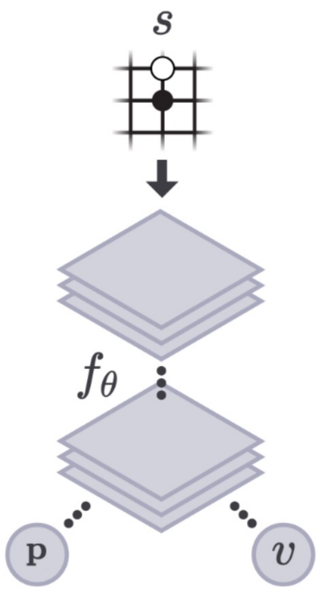
AlphaZero

- One deep neural network $f_\theta(s) = (p, v)$ with
  - move probabilities $p = \mathrm{Pr}(a|s)$ 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"

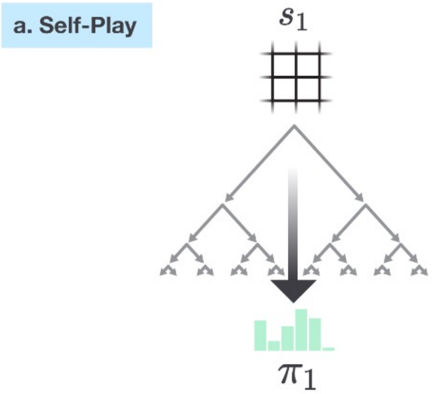
"policy evaluation"

"policy improvement"



Fraunhofer IIS

AlphaZero - Method



AlphaZero - Results

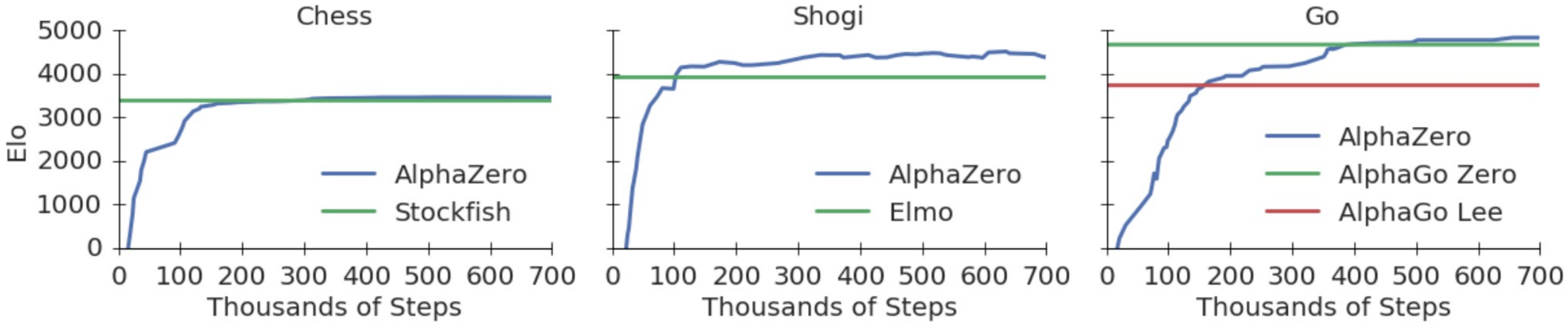


Figure 1: Training *AlphaZero* for 700,000 steps. Elo ratings were computed from evaluation games between different players when given one second per move. **a** Performance of *AlphaZero* in chess, compared to 2016 TCEC world-champion program *Stockfish*. **b** Performance of *AlphaZero* in shogi, compared 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*).

AlphaZero – Notes

- Loss function:
  - $l = (z - v)^2 - \pi^T \log(p) + c||\theta||^2$

    MSE Cross-entropy Weight regularization
- Neural network consists of
  - Single convolutional block + 19 or 39 residual blocks
  - Two separate feed-forward policy and value heads

- Actions are sampled from the MCTS policy during training, but selected greedily during deployment

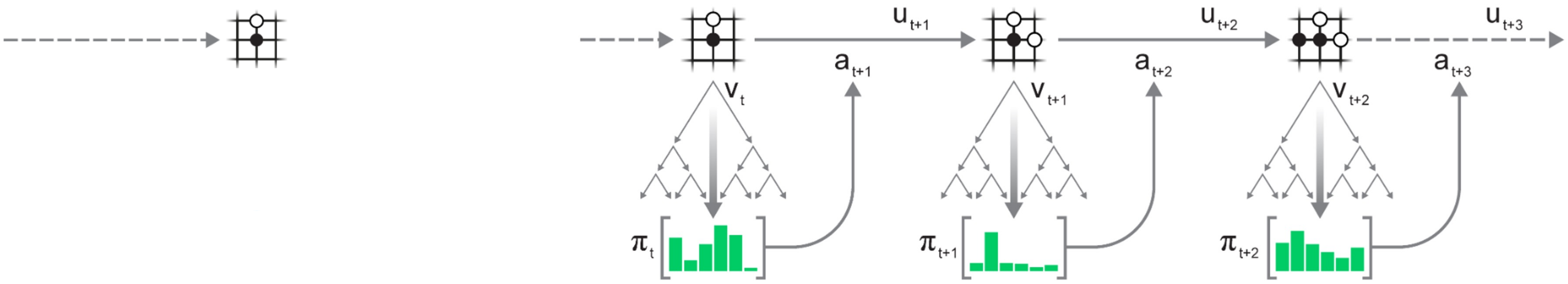
muZero: Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

- **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



Fraunhofer IIS

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 π_{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

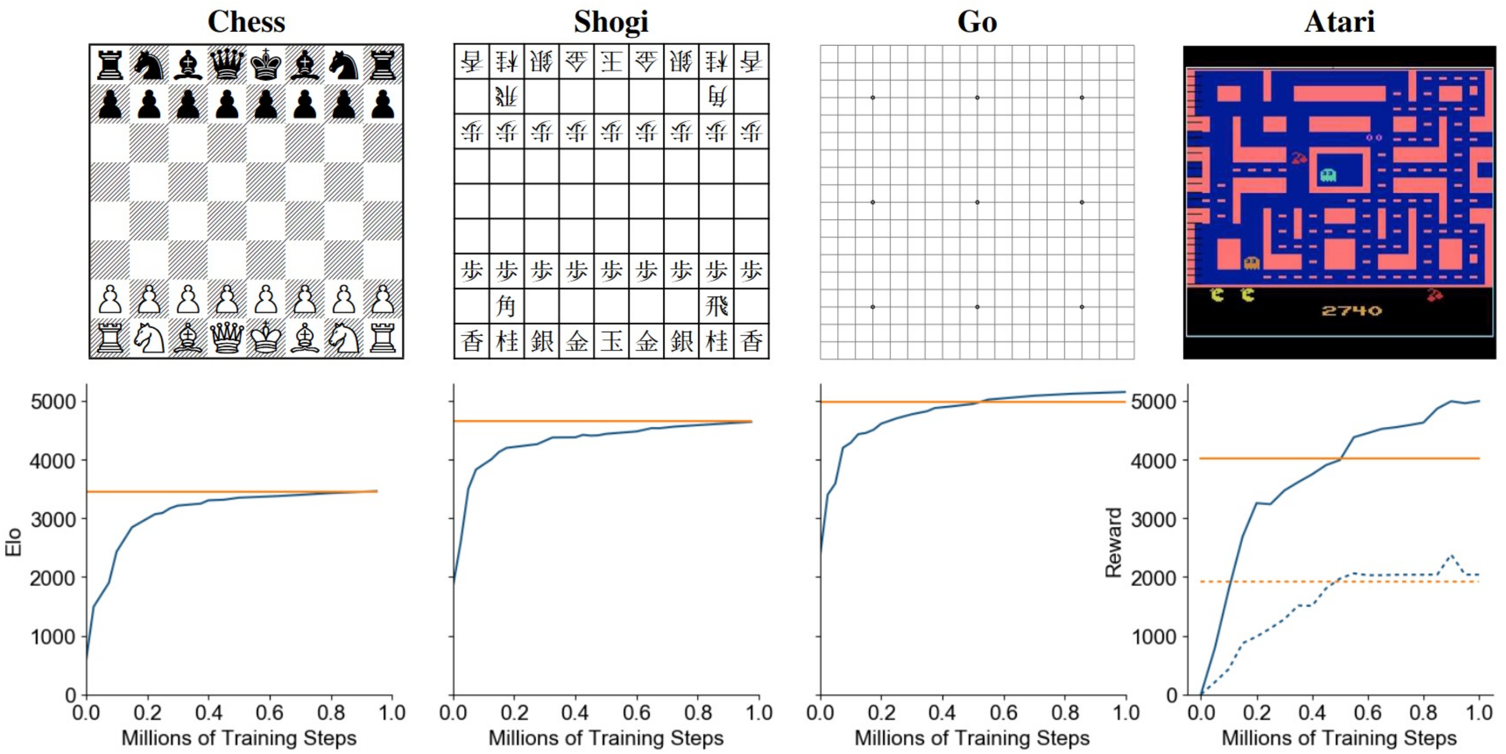


Figure 2: **Evaluation of *MuZero* throughout training in chess, shogi, Go and Atari.** The x-axis shows millions of training steps. For chess, shogi and Go, the y-axis shows Elo rating, established by playing games against *AlphaZero* using 800 simulations per move for both players. *MuZero*'s Elo is 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 [25].

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
MuZero	**2041.1%**	**4999.2%**	20.0B	12 hours	1M
IMPALA [9]	191.8%	957.6%	200M	–	–
Rainbow [17]	231.1%	–	200M	10 days	–
UNREAL[a] [19]	250%[a]	880%[a]	250M	–	–
LASER [36]	431%	–	200M	–	–
MuZero Reanalyze	**731.1%**	**2168.9%**	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. [a]Hyper-parameters were tuned per game.

Cross-Entropy Method (CEM)

Cross-Entropy Method (CEM)

Pseudocode

Initialize $\mu \in \mathbb{R}^d, \sigma \in \mathbb{R}^d$

for iteration $= 1, 2, \ldots$ **do**

    Collect n samples of $\theta_i \sim N(\mu, \text{diag}(\sigma))$

    Perform a noisy evaluation $R_i \sim \theta_i$

    Select the top $p\%$ of samples (e.g. $p = 20$), which we'll call the **elite set**

    Fit a Gaussian distribution, with diagonal covariance, to the elite set, obtaining a new μ, σ.

end for

Return the final μ.

MLSS 2016 on Deep Reinforcement Learning by John Schulman

Thank you for your attention!

Fraunhofer IIS

Fraunhofer-Institut für Integrierte Schaltungen IIS