Deep Learning

A course about theory & practice

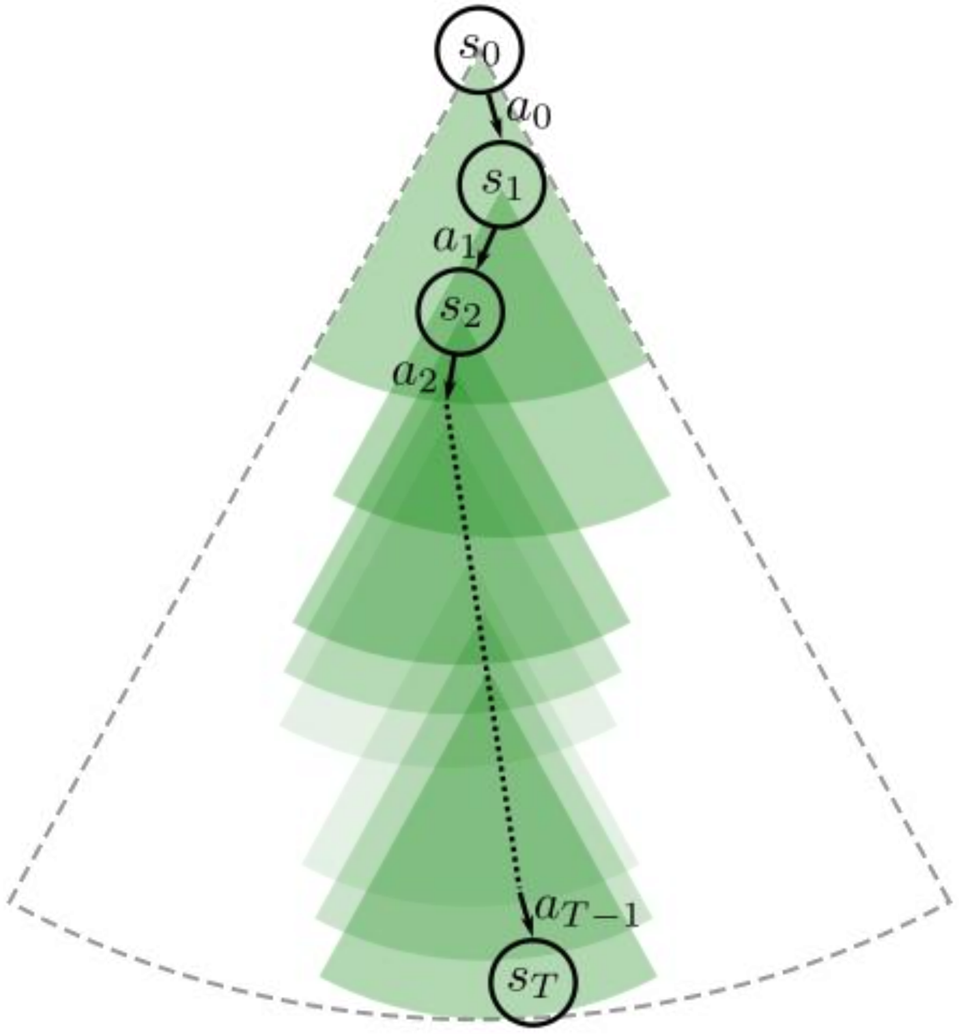
AlphaZero, MuZero

Marco Piastra

AlphaZero = MCTS + DNN

Monte Carlo Tree Search (MCTS) method

- ***MCTS*** *method:*
  - *memory* of past playouts in a single *MCTS* step
    (collected in the tree statistics)
  - *knowledge transfer* between *MCTS* steps
    (by reusing subtrees already explored)
  - optimal policy only *partially* defined
    (on actually computed states)
  - *intrinsically stochastic* policy optimization
    (the same initial state
    can give rise to different optimizations)
  - What about *knowledge transfer*
    between *MCTS episodes?*
    transferring the entire MCTS tree
    would rapidly cause its explosive growth…

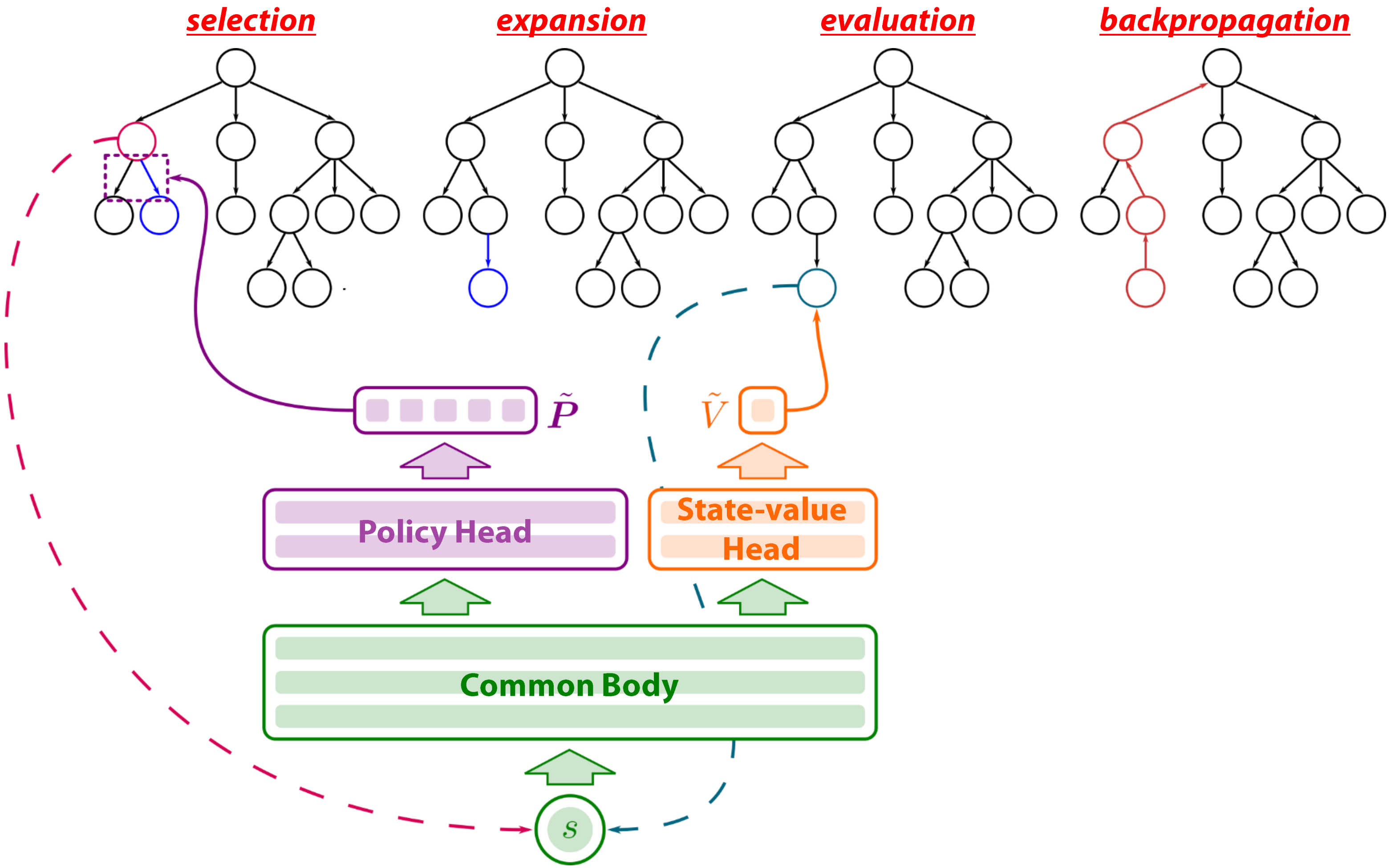


Knowledge transfer between MCTS episodes

- ***AlphaZero*** [Silver et al. 2017]
  - *Monte Carlo Tree Search (MCTS):*
    improves the policy by focusing on the most promising actions
  - *Deep Neural Network (DNN):*
    learns the improved policy and transfers it between MCTS episodes

AlphaZero

- ***AlphaZero = MCTS + DNN***

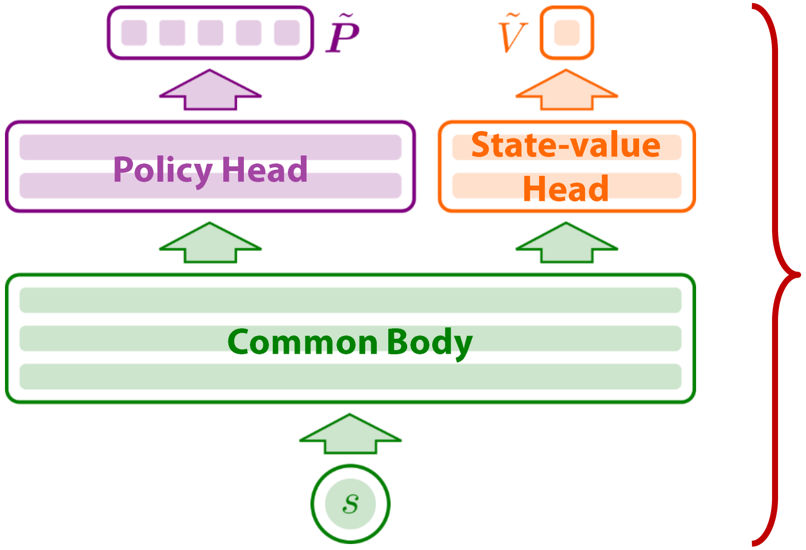


DNN in AlphaZero

- **DNN in AlphaZero**
  - <u>*input:*</u> a *state* s
  - <u>*output:*</u> a *probability distribution* $\tilde{\boldsymbol{P}}(s) := [\tilde{P}(a \mid s)]_{a \in \mathcal{A}(S)}$ — stochastic policy (a <u>vector</u> of probabilities)

    and a *state-value* $\tilde{V}(s)$
    - predicts the expected reward for state s
    - acts as an ***actor-critic*** in the <u>*training*</u> of ***parameters*** ϑ of the net

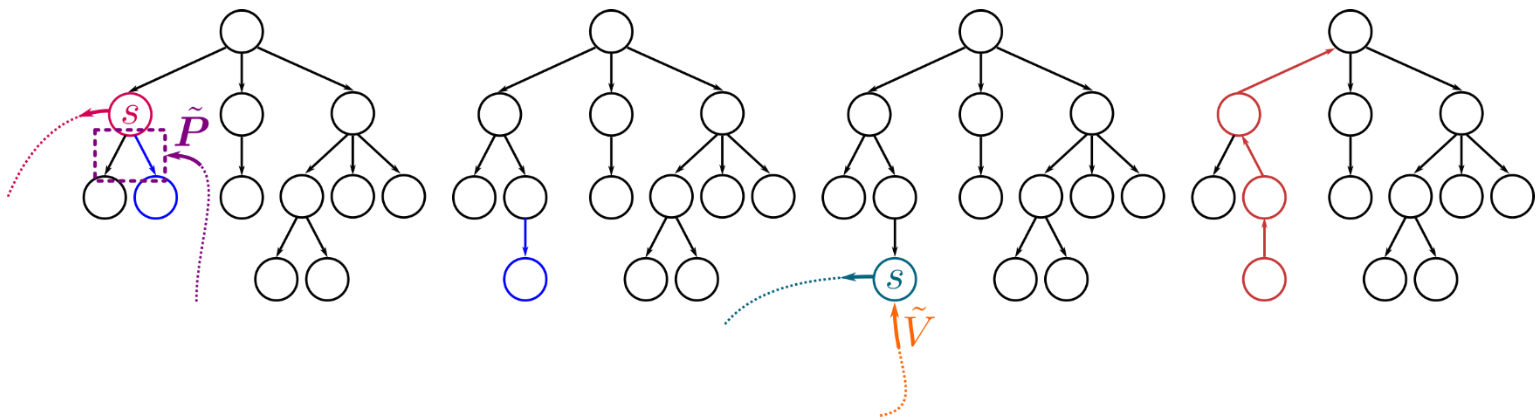
$\tilde{V}$ is compared with the *actual* reward r, which also impacts on training $\tilde{\boldsymbol{P}}$ by <u>*backpropagating*</u> through the *Common Body*



"Y" shape

MCTS step in AlphaZero

- ***MCTS step*** **in** ***AlphaZero***



- <u>*selection:*</u> *UCT* policy is replaced with ***PUCT*** *("Predictor" + UCT)*

MCTS estimation of $Q(s,a)$ for *DNN* policy

DNN policy

$$\pi^{\text{PUCT}}(s) := \underset{a}{\operatorname{argmax}} \left\{ \hat{Q}(s,a) + c(s)\tilde{P}(a \mid s) \frac{\sqrt{N(s)}}{N(s,a) + 1} \right\}$$

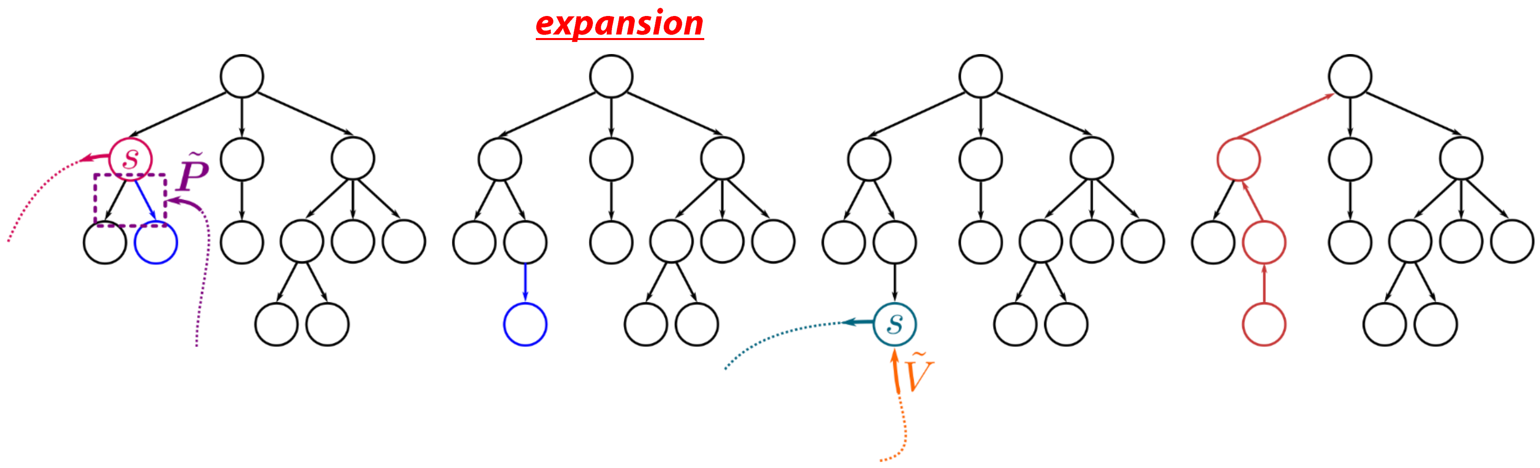
exploration rate $c(s) := \log \frac{1 + N(s) + c_{\text{base}}}{c_{\text{base}}} + c_{\text{init}}$

(slowly grows with search time)

avoids division by 0

MCTS step in AlphaZero

- ***MCTS step* in *AlphaZero***

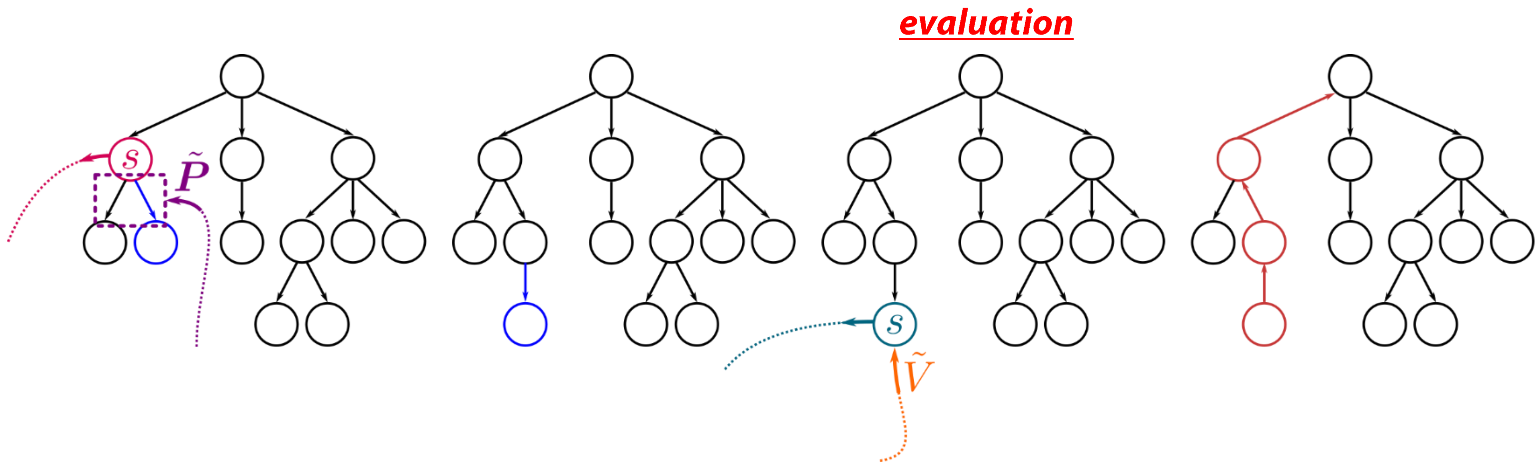


- *expansion:* initialization of the leaf new node s_L:

$$N(s_L) := 0 \quad \text{and} \quad \forall\, a \in \mathcal{A}(s_L) \qquad N(s_L, a_L) := 0, \quad \hat{Q}(s_L, a_L) := -\infty$$

MCTS step in AlphaZero

- ***MCTS step** in **AlphaZero***



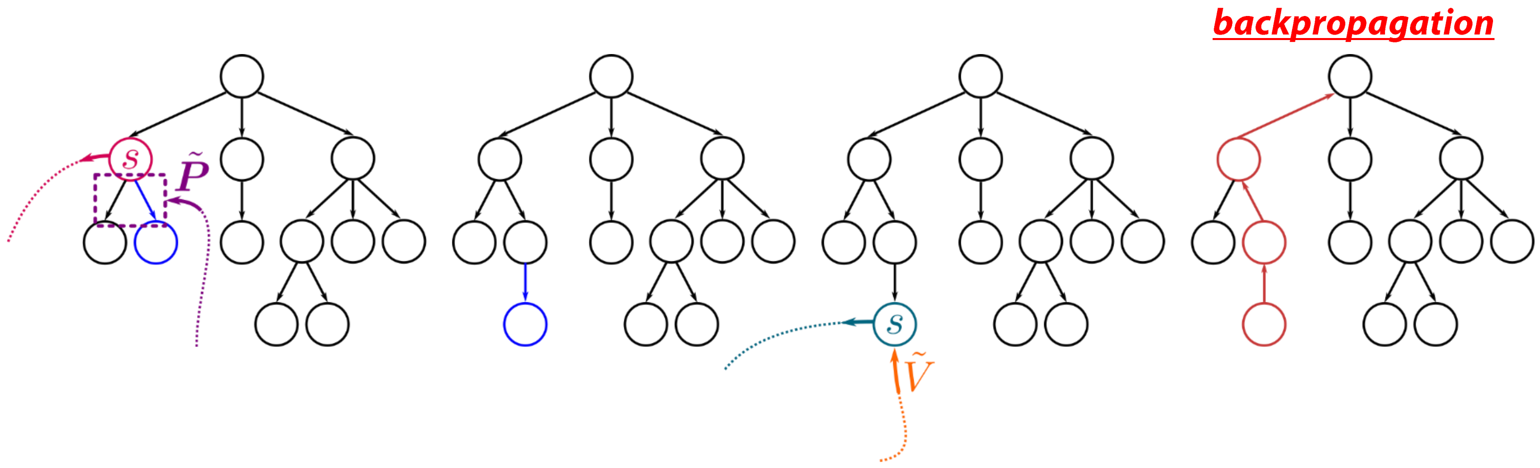
- *expansion:* initialization of the leaf new node s_L:

$$N(s_L) := 0 \quad \text{and} \quad \forall\, a \in \mathcal{A}(s_L) \qquad N(s_L, a_L) := 0, \quad \hat{Q}(s_L, a_L) := -\infty$$

- *evaluation* (in place of *simulation*): expected reward is $\tilde{V}(s_L)$

MCTS step in AlphaZero

- ***MCTS step** in **AlphaZero***



- *expansion:* initialization of the leaf new node s_L:

$$N(s_L) := 0 \quad \text{and} \quad \forall\, a \in \mathcal{A}(s_L) \qquad N(s_L, a_L) := 0, \quad \hat{Q}(s_L, a_L) := -\infty$$

- *evaluation* (in place of *simulation*): expected reward is $\tilde{V}(s_L)$
- *backpropagation:* for each state s and action a visited in selection/expansion:

$$\begin{aligned} N(s) &:= N(s) + 1, \\ N(s,a) &:= N(s,a) + 1 \end{aligned} \quad \text{and} \quad \hat{Q}(s,a) := \hat{Q}(s,a) + \frac{\tilde{V}(s_L) - \hat{Q}(s,a)}{N(s,a)}$$

MCTS step in AlphaZero: policies

- *Selection* policy: ***PUCT***

$$\pi^{\mathrm{sel}}(s) := \pi^{\mathrm{PUCT}}(s) := \operatorname*{argmax}_a \left\{ \hat{Q}(s,a) + c(s)\tilde{P}(a \mid s) \frac{\sqrt{N(s)}}{N(s,a)+1} \right\}$$

- *Output* policy:

$$\pi^{\mathrm{out}}(s) \sim \left[\hat{P}(a \mid s) := \frac{N(s,a)}{N(s)} \right]_{a \in \mathcal{A}(s)}$$

taking frequencies as probabilities
(in place of their argmax as output action)
ensures ***exploration***

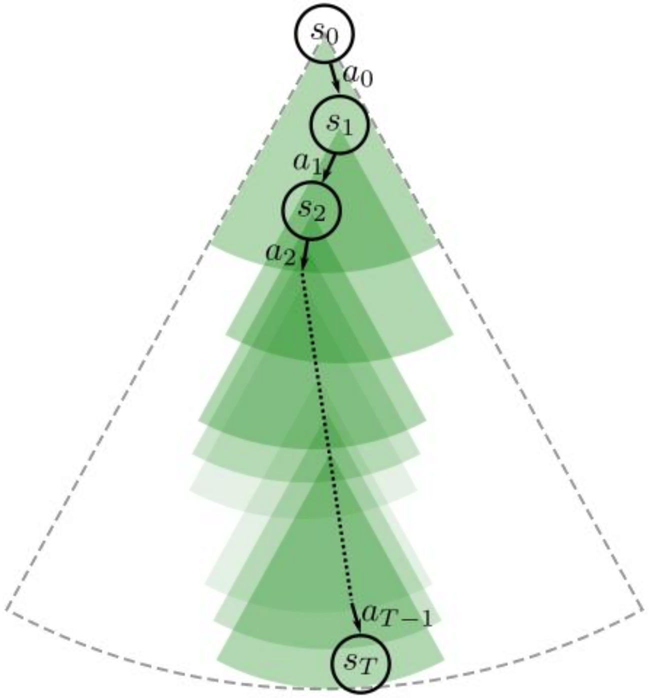
(the simulation policy does not exist anymore)

DNN training in AlphaZero

- ***Data items*** from a <u>single</u> *MCTS episode:*

After an *MCTS episode* $\tau = \langle (s_0, a_0, r_0), \ldots, (s_{T-1}, a_{T-1}, r_{T-1}), (s_T, r_T) \rangle$
with episode return

$$R(\tau) = \sum_{t=0}^{T} \gamma^t r_t$$



- for each *non-terminal* state $s_i \ (i = 0 \ldots T-1)$ in τ

$$\hat{\boldsymbol{P}}(s_i) := \left[\hat{P}(a \mid s_i) := \frac{N(s_i, a)}{N(s_i)} \right]_{a \in \mathcal{A}(s_i)}$$

vector of frequencies

- the ***output*** of episode τ is

$$D^\tau := \left\{ \langle s_i, \hat{\boldsymbol{P}}(s_i), \hat{V}_i^\tau \rangle \right\}_{i=0 \ldots T-1}$$

data item

Hold on a minute: how can this be collected?

DNN training in AlphaZero

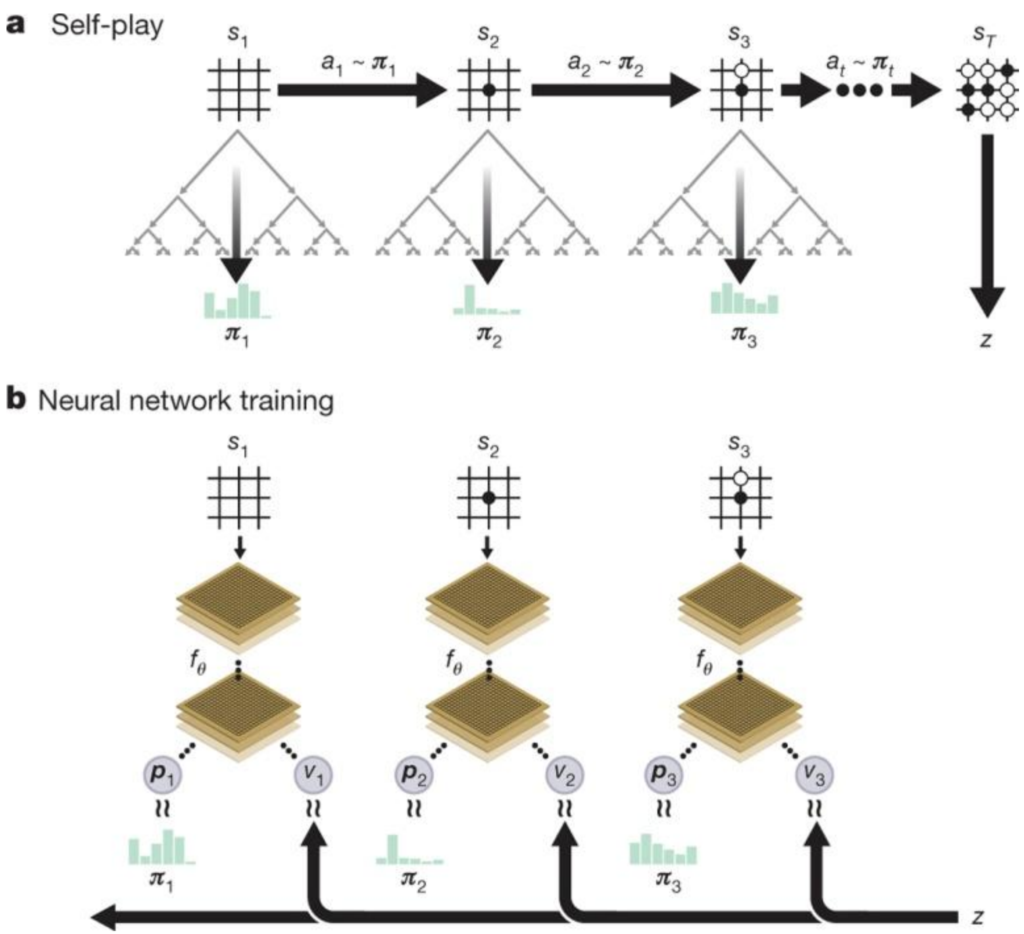
- ***State values are retrofitted***

$$D^\tau := \left\{ \underbrace{\left(s_i, \hat{\boldsymbol{P}}(s_i), \boxed{\hat{V}_i^\tau}\right)}_{\text{data item}} \right\}_{i=0\ldots T-1}$$

At the end of each episode, all rewards are known
Then:

$$V_i^\tau = \sum_{t=i}^{T} \gamma^t r_t$$

[Image from https://www.nature.com/articles/nature24270]



DNN training in AlphaZero

- ***Iteration:***

K times $\left\{ \begin{array}{l} \text{1) play one } \textit{MCTS episode } \tau \\ \text{2) collect data items } D^\tau \end{array} \right.$

3) train the parameters of the *DNN* by using as ***dataset***

$$D := \bigcup_{\tau=1}^{M} D^\tau$$

- In the limit of *infinite* iterations:

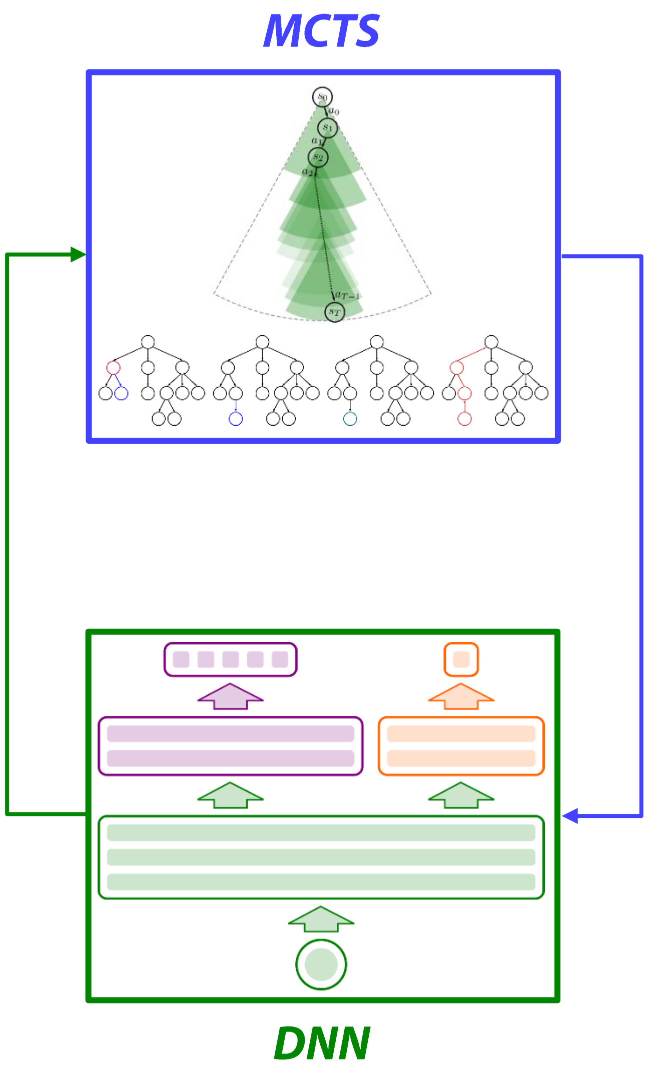
$$\pi^{\mathrm{DNN}}(s) := \operatorname*{argmax}_{a \in \mathcal{A}(s)} \tilde{P}(a \mid s) \to \pi^*(s) \quad \forall\, s$$

AlphaZero

- ***AlphaZero*:**
  - <u>*memory*</u> of past playouts in a single *MCTS* step
    (collected in the tree statistics)
  - <u>*knowledge transfer*</u> between *MCTS* steps
    (by reusing subtrees already explored)
  - <u>*knowledge transfer*</u> between *MCTS* episodes
    (provided by DNN)

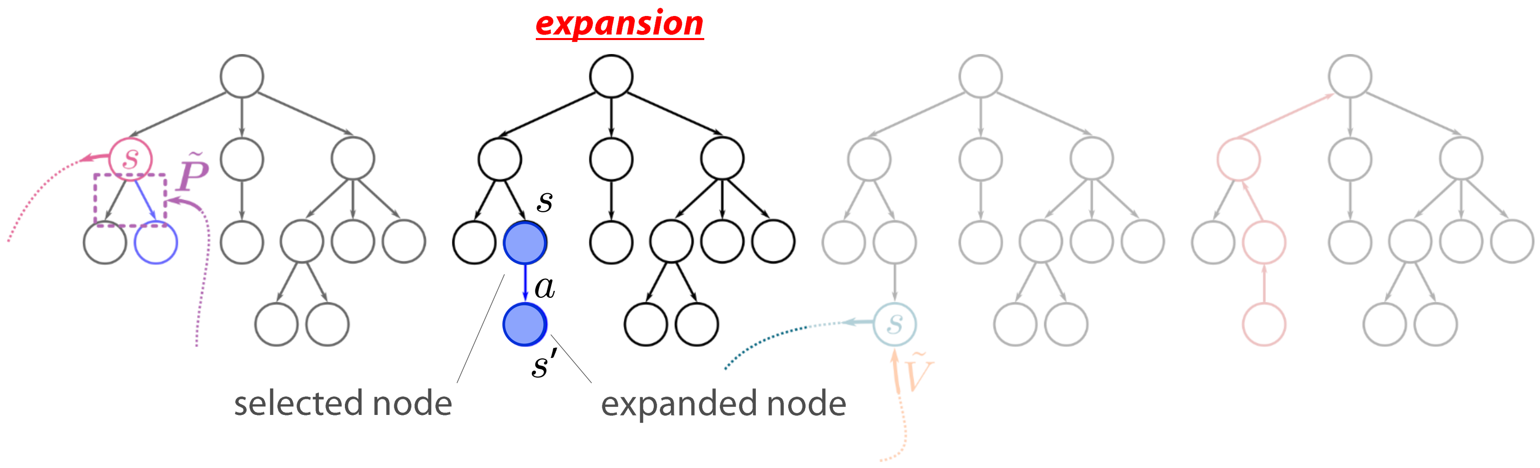
  - <u>*deterministic*</u> policy optimization
    with policy defined for all states s :

$$\pi^{\text{DNN}}(s) := \underset{a \in \mathcal{A}(s)}{\text{argmax}}\, \tilde{P}(a \mid s)$$



MuZero = model-free MCTS + DNN

Expansion step in AlphaZero



The expansion step of a selected node s entails applying action a and entering node s'

- This is a *state transition* to s', given a and s : the model of the environment need to be known
- It is *hypothetical*: it will not be reproduced in the environment
- Alternative transitions will also be considered (*the branches of the tree*)

at least to some extent

Not all environment simulators can support this

The MuZero solution

MuZero uses three different networks:

- ***representation***
  It encodes a set of observations into (an internal) state representation

$$h(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t; \theta_h) \rightarrow \boldsymbol{s}_t$$

- ***prediction***
  It returns the predicted policy (as probability distribution) and state value

$$f(\boldsymbol{s}_t; \theta_f) \rightarrow \boldsymbol{p}(\boldsymbol{s}_t), v_t$$

- ***dynamics***
  Given an (internal) state representation and one action, it predicts the next state and reward

$$g(\boldsymbol{s}_t, \boldsymbol{a}_t; \theta_g) \rightarrow \boldsymbol{s}_{t+1}, r_{t+1}$$

The MuZero solution

MuZero uses three different networks:

- ***representation***

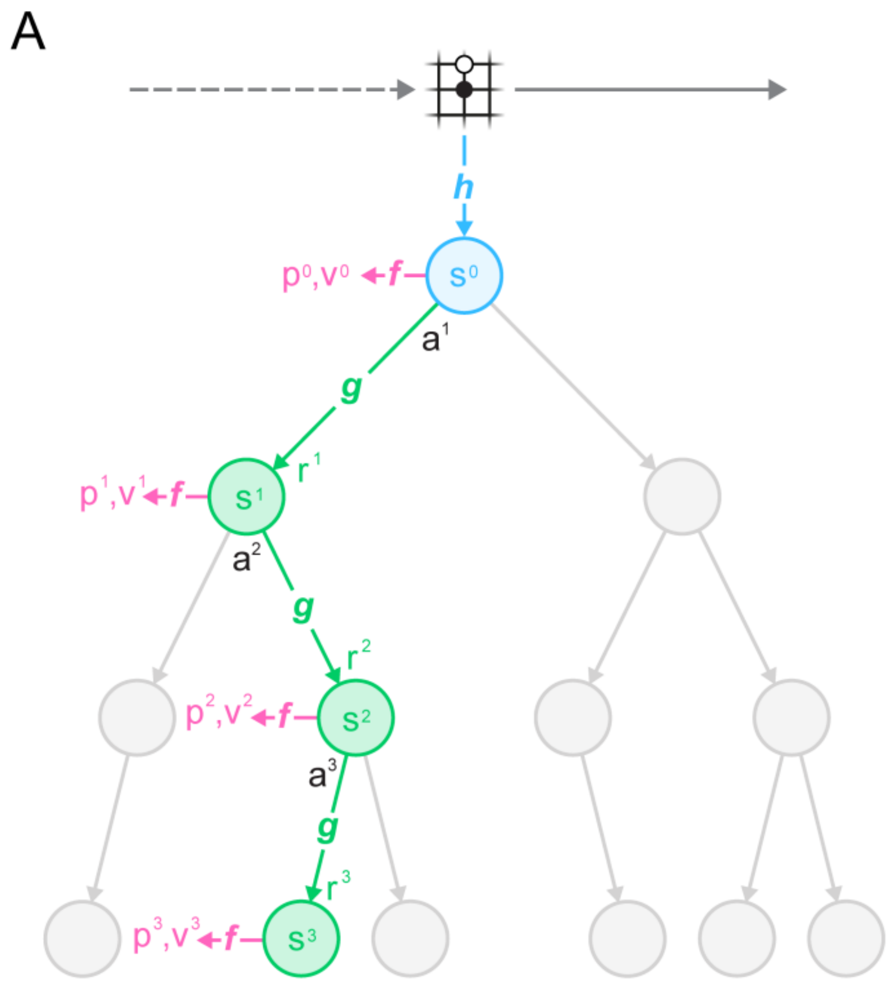
$$h(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t; \theta_h) \rightarrow \boldsymbol{s}_t$$

- ***prediction***

$$f(\boldsymbol{s}_t; \theta_f) \rightarrow \boldsymbol{p}(\boldsymbol{s}_t), v_t$$

- ***dynamics***

$$g(\boldsymbol{s}_t, \boldsymbol{a}_t; \theta_g) \rightarrow \boldsymbol{s}_{t+1}, r_{t+1}$$



[Image from https://arxiv.org/pdf/1911.08265v2]

The MuZero solution

- MuZero training
  - Sample a trajectory τ from the replay buffer
  - Select an initial step t

$$h(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t; \theta_h) \rightarrow \boldsymbol{s}_0$$

$$\boldsymbol{p}(\boldsymbol{s}_0), v_0 = f(\boldsymbol{s}_0; \theta_f)$$

  - Unroll for K steps, at each step k :

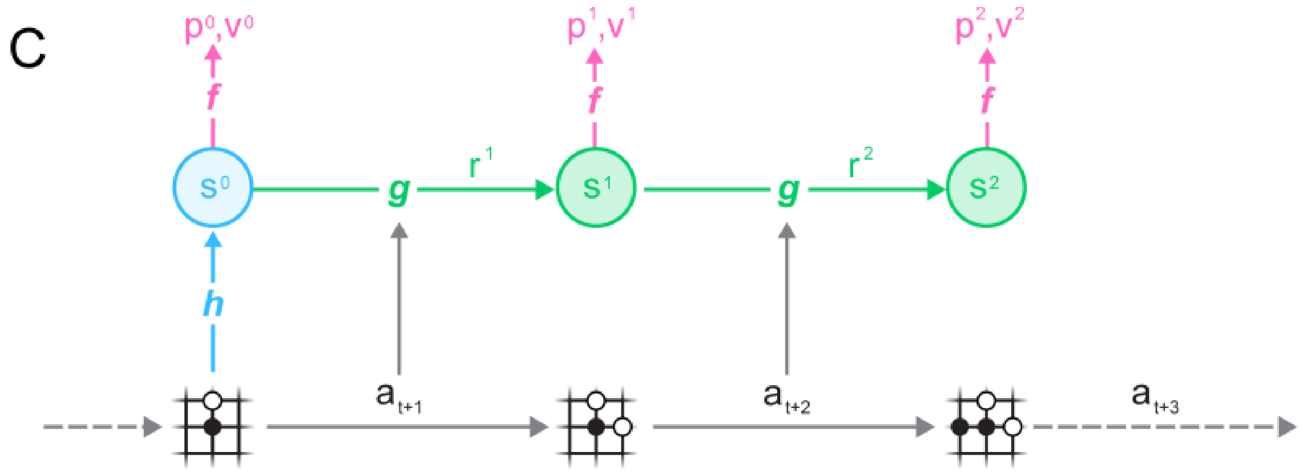
$$\boldsymbol{s}_k, r_k = g(\boldsymbol{s}_{k-1}, \boldsymbol{a}_{k-1}; \theta_g)$$

$$\boldsymbol{p}(\boldsymbol{s}_k), v_k = f(\boldsymbol{s}_k; \theta_f)$$

Update the three networks jointly by comparing

$$\boldsymbol{p}(\boldsymbol{s}_k), v_k, r_k$$

with the actual values stored in τ



[Image from https://arxiv.org/pdf/1911.08265v2]