A. Details on search for AlphaZero

Below we briefly present details of the search procedure for AlphaZero. Please refer to the original work (Silver et al., 2017a) for more comprehensive explanations.

As explained in the main text, the search procedure starts with a MDP state x_0, which is used as the root node of the tree. The rest of this tree is progressively built as more simulations are generated. In addition to Q-function $Q(x, a)$, prior $\pi_\theta(x, a)$ and visit counts $n(x, a)$, each node also maintains a reward $R(x, a) = r(x, a)$ and value $V(x)$ estimate.

In each simulation, the search consists of several parts: **Selection**, **Expansion** and **Backup**, as below.

Selection. From the root node x_0, the search traverses the tree using the action selection formula of Eq. 1 until a leaf node x_l is reached.

Expansion. After a leaf node x_l is reached, the search selects an action from the leaf node, generates the corresponding child node x_c and appends it to the tree $\mathcal{T}$. The statistics for the new node are then initialized to $Q(x_c, a) = \min_{x\in\mathcal{T},a'\in\mathcal{A}} Q(x, a')$ (pessimistic initialization), $n(x, a) = 0$ for $\forall a \in \mathcal{A}$.

Back-up. The back-up consists of updating statistics of nodes encountered during the forward traversal. Statistics that need updating include the Q-function $Q(x, a)$, count $n(x, a)$ and value $V(x)$. The newly expanded node n_c updates its value $V(x)$ to be either the Monte-Carlo estimation from random rollouts (e.g. board games) or a prediction of the value network (e.g. Atari games). For the other nodes encountered during the forward traversal, all other statistics are updated as follows:

$$\begin{aligned} V(x) &\leftarrow (V(x) \cdot \sum_b n(x, b) + (R(x, a) + \gamma V(\text{child}(x, a))/(1 + \sum_b n(x, b)) \\ Q(x, a) &\leftarrow R(x, a) + \gamma V(\text{child}(x, a)), \\ n(x, a) &\leftarrow n(x, a) + 1, \end{aligned}$$

where $\text{child}(x, a)$ refers to the child node obtained by taking action a from node x.

Note that, in order to make search parameters agnostic to the scale of the numerical rewards (and, therefore, values), Q-function statistics $Q(x, a)$ are always normalized by statistics in the search tree before applying the action selection formula; in practice, Eq. 1 uses the normalized $Q_z(x, a)$ defined as:

$$Q_z(x, a) = \frac{Q(x, a) - \min_{x\in\mathcal{T},a\in\mathcal{A}} Q(x, a)}{\max_{x\in\mathcal{T},a\in\mathcal{A}} Q(x, a) - \min_{x\in\mathcal{T},a\in\mathcal{A}} Q(x, a)}. \tag{18}$$

B. Implementation details

B.1. Agent

For ease of implementation and availability of computational resources, the experimental results from Section 5 were obtained with a scaled-down version of MuZero (Schrittwieser et al., 2019). In particular, our implementation uses smaller networks compared to the architecture described in Appendix F of (Schrittwieser et al., 2019): we use only 5 residual blocks with 128 hidden layers for the dynamics function, and the residual blocks in the representation functions have half the number of channels. Furthermore, we use a stack of only 4 past observations instead of 32. Additionally, some algorithmic refinements (such as those described in Appendix H of (Schrittwieser et al., 2019)) have not been implemented in the version that we use in this paper.

Our experimental results have been obtained using either 4 or 8 Tesla v100 GPUs for learning (compared to 8 third-generation Google Cloud TPUs (Google, 2020) in the original MuZero paper, which are approximately equivalent to 64 v100 GPUs). Each learner GPU receives data from a separated, prioritized experience replay buffer (Horgan et al., 2018) storing the last 500000 transitions. Each of these buffers is filled by 512 dedicated CPU actors[12], each running a different environment instance. Finally, each actor receives updated parameters from the learner every 500 learner steps (corresponding to

[12]For 50 simulations per step; this number is scaled linearly as $12 + 10 \cdot N_{\text{sim}}$ to maintain a constant total number of frames per second when varying N_{sim}.

approximately 4 minutes of wall-clock time); because episodes can potentially last several minutes of wall-clock time, weights updating will usually occur within the duration of an episode. The total score at the end of an episode is associated to the version of the weights that were used to select the final action in the episode.

Hyperparameters choice generally follows those of (Schrittwieser et al., 2019), with the exception that we use the Adam optimizer with a constant learning rate of 0.001.

B.2. Details on discretizing continuous action space

AlphaZero (Silver et al., 2017a) is designed for discrete action spaces. When applying this algorithm to continuous control, we use the method described in (Tang and Agrawal, 2019) to discretize the action space. Although the idea is simple, discretizing continuous action space has proved empirically efficient (Andrychowicz et al., 2020; Tang and Agrawal, 2019). We present the details below for completeness.

Discretizing the action space We consider a continuous action space $\mathcal{A} = [-1, 1]^m$ with m dimensions. Each dimension is discretized into $K = 5$ bins; specifically, the continuous action along each dimension is replaced by K atomic categorical actions, evenly spaced between $[-1, 1]$. This leads to a total of K^m actions, which grows exponentially fast (e.g. $m = 6$ leads to about 10^4 joint actions). To avoid the curse of dimensionality, we assume that the parameterized policy can be factorized as $\pi_\theta(\mathbf{a}|x) = \Pi_{i=1}^m \pi_\theta^{(i)}(a_i|x)$, where $\pi_\theta^{(i)}(a_i|x)$ is the marginal distribution for dimension i, $a_i \in \{1, 2...K\}$ is the discrete action along dimension i and $\mathbf{a} = [a_1, a_2...a_m]$ is the joint action.

Modification to the search procedure Though it is convenient to assume a factorized form of the parameterized policy (Andrychowicz et al., 2020; Tang and Agrawal, 2019), it is not as straightforward to apply the same factorization assumption to the Q-function $Q(x, \mathbf{a})$. A most naive way of applying the search procedure is to maintain a Q-table of size K^m with one entry for each joint action, which may not be tractable in practice. Instead, we maintain m separate Q-tables each with K entries $Q_i(x, a_i)$. We also maintain m count tables $n(x, a_i)$ with K entries for each dimension.

To make the presentation clear, we detail on how the search is applied. At each node of the search tree, we maintain m tables each with K entries as introduced above. The three core components of the tree search are modified as follows.

- **Selection.** During forward action selection, the algorithm needs to select an action $\mathbf{a}$ at node x. This joint action $\mathbf{a}$ has all its components a_i selected independently, using the action selection formula applied to each dimension. To select action at dimension i, we need the Q-table $Q_i(x, a_i)$, the prior $\pi_\theta^{(i)}(a_i|x)$ and count $n(x, a_i)$ for dimension i.

- **Expansion.** The expansion part does not change.

- **Back-up.** During the value back-up, we update Q-tables of each dimension independently. At a node x, given the downstream reward $R(x, a)$ and child value $V(\text{child}(x, \mathbf{a}))$, we generate the target update for each Q-table and count table as $Q(x, a_i) \leftarrow R(x, a) + \gamma V(\text{child}(x, \mathbf{a}))$ and $n(x, a_i) \leftarrow n(x, a_i) + 1$.

The m small Q-tables can be interpreted as maintaining the marginalized values of the joint Q-table. Indeed, let us denote by $Q(x, \mathbf{a})$ the joint Q-table with K^m entries. At dimension i, the Q-table $Q(x, a_i)$ increments its values purely based on the choice of a_i, regardless of actions in other dimension $a_j, j \neq i$. This implies that the Q-table $Q(x, a_i)$ marginalizes the joint Q-table $Q(x, \mathbf{a})$ via the visit count distribution.

Details on the learning At the end of the tree search, a distribution target $\hat{\pi}$ or $\bar{\pi}$ is computed from the root node. In the discretized case, each component of the target distribution is computed independently. For example, $\hat{\pi}_i$ is computed from $N(x_0, a_i)$. The target distribution derived from constrained optimization $\bar{\pi}_i$ is also computed independently across dimensions, from $Q(x_0, a_i)$ and $N(x_0, a_i)$. In general, let $\pi_{\text{target}}(\cdot|x)$ be the target distribution and $\pi_{\text{target}}^{(i)}(\cdot|x)$ its marginal for dimension i. Due to the factorized assumption on the policy distribution, the update can be carried out independently for each dimension. Indeed, $\text{KL}[\pi_{\text{target}}(\cdot|x), \pi_\theta(\cdot|x)] = \sum_{i=1}^m \text{KL}[\pi_{\text{target}}^{(i)}(\cdot|x), \pi_\theta^{(i)}(\cdot|x)]$, sums over dimensions.

B.3. Practical computation of $\bar{\pi}$

The vector $\bar{\pi}$ is defined as the solution to a multi-dimensional optimization problem; however, we show that it can be computed easily by dichotomic search. We first restate the definition of $\bar{\pi}$,

$$\bar{\pi} \triangleq \arg\max_{\mathbf{y}\in\mathcal{S}} \big[\mathbf{q}^T\mathbf{y} - \lambda_N \mathrm{KL}[\pi_\theta, \mathbf{y}]\big]. \tag{8}$$

Let us define

$$\forall a \in \mathcal{A} \quad \pi_\alpha[a] \triangleq \lambda_N \frac{\pi_\theta[a]}{\alpha - \mathbf{q}[a]} \quad \text{and} \quad \alpha^\star \triangleq \max\left\{\alpha \in \mathbb{R} \text{ s.t } \sum_b \pi_\alpha[b] = 1\right\}. \tag{19}$$

Proposition 4.

$$(i) \quad \pi_{\alpha^\star} = \bar{\pi} \tag{20}$$

$$(ii) \quad \alpha^\star \geq \alpha_{min} \triangleq \max_{b\in\mathcal{A}}(q[b] + \lambda_N \cdot \pi_\theta[b]) \tag{21}$$

$$(iii) \quad \alpha^\star \leq \alpha_{max} \triangleq \max_{b\in\mathcal{A}} q[b] + \lambda_N \tag{22}$$

$$\tag{23}$$

As $\sum_b \bar{\pi}_\alpha[b]$ is strictly decreasing on $\alpha \in (\alpha_{\min}, \alpha_{\max})$, Proposition 4 guarantees that $\bar{\pi}$ can be computed easily using dichotomic search over $(\alpha_{\min}, \alpha_{\max})$.

Proof of (i).

Proof. The proof start the same as the one of Lemma 3 of Appendix D.1 setting $f(x) = -\log(x)$ to get

$$\exists\alpha \quad q + \lambda_N \cdot \frac{\pi_\theta}{\bar{\pi}} = \alpha\mathbb{1}, \tag{24}$$

with $\mathbb{1}$ being the the vector such that $\forall a \ \mathbb{1}_a = 1$. Therefore there exists $\alpha \in \mathbb{R}$ such that

$$\bar{\pi} = \frac{\lambda_N \cdot \pi_\theta}{\alpha - q} \tag{25}$$

Then α is set such that $\sum_b \bar{\pi}_b = 1$ and $\forall b \ \bar{\pi}_b \geq 0$. □

Proof of (ii).

Proof.

$$\forall a \quad 1 \geq \bar{\pi}[a] = \frac{\lambda_N \cdot \pi_\theta[a]}{\alpha - q[a]} \implies \forall a \quad \alpha \geq q[a] + \lambda_N \cdot \pi_\theta[a] \tag{26}$$

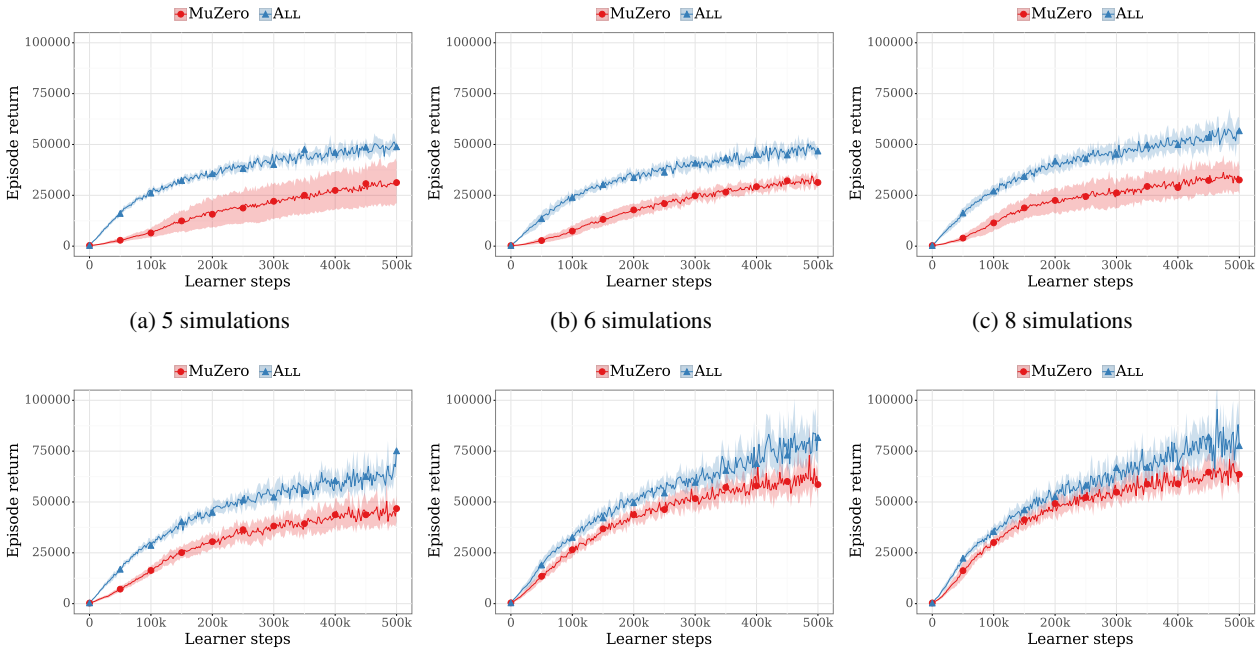
□

Proof of (iii).

Proof.

$$\sum_b \pi_{\alpha_{\max}}[b] = \sum_b \frac{\lambda_N \cdot \pi_\theta[b]}{\max_{c\in\mathcal{A}} q[c] + \lambda_N - q[b]} \leq \sum_b \frac{\lambda_N \cdot \pi_\theta[b]}{\lambda_N} = 1 \tag{27}$$

We combine this with the fact that $\sum_b \pi_\alpha[b]$ is a decreasing function of α for any $\alpha > \max_b q[b]$, and $\sum_b \pi_{\alpha^\star}[b] = 1$. □



(a) 5 simulations (b) 6 simulations (c) 8 simulations

(d) 12 simulations (e) 24 simulations (f) 50 simulations

Figure 6. Dispersion between seeds at different number of simulations per step on Ms Pacman.

C. Additional experimental results

C.1. Complements to Section 5.1

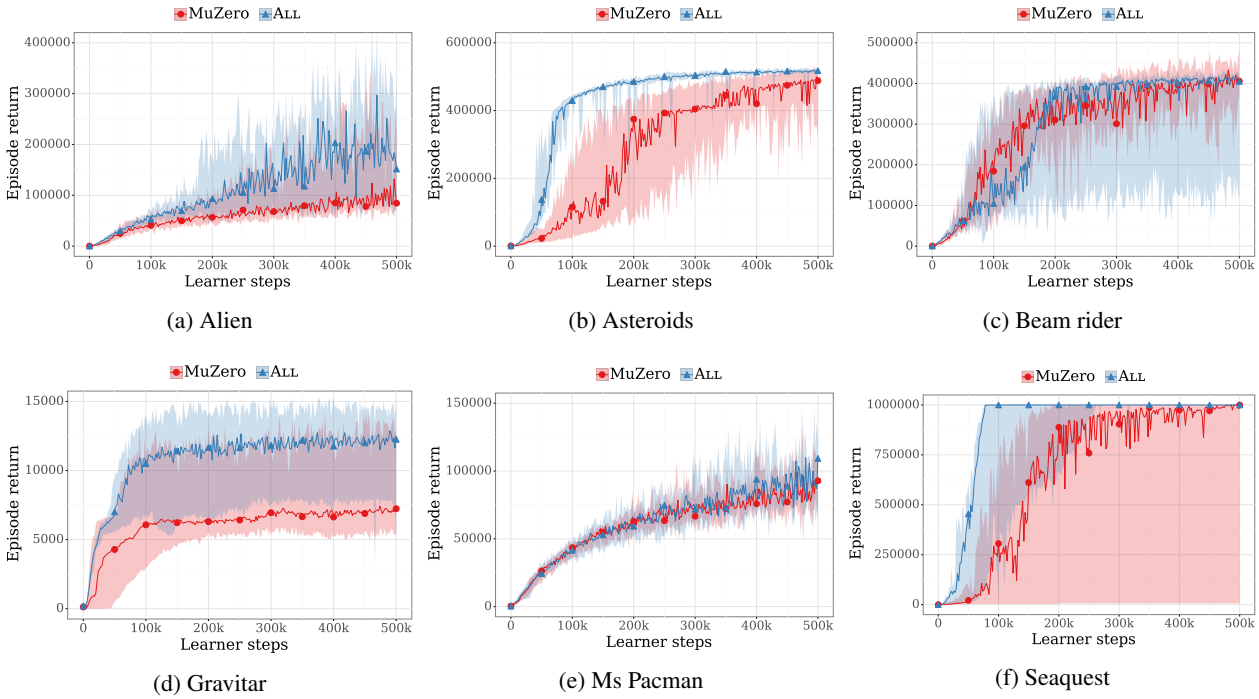
Figure 6 presents a comparison of the score obtained by our MuZero implementation and the proposed ALL variant at different simulation budgets on the Ms. Pacman level; the results from Figure 2 are also included fore completeness. In this experiment, we used 8 seeds with 8 GPUs per seed and a batch size of 256 per GPU. We use the same set of hyper-parameters for MuZero and ALL; these parameters were tuned on MuZero. The solid line corresponds to the average score (solid line) and the 95% confidence interval (shaded area) over the 8 seeds, averaged for each seed over buckets of 2000 learner steps without additional smoothing. Interestingly, we observe that ALL provides improved performance at low simulation budgets while also reducing the dispersion between seeds.

Figure 7 presents a comparison of the score obtained by our MuZero implementation and the proposed ALL variant on six Atari games, using 6 seeds per game and a batch size of 512 per GPU and 8 GPUs; we use the same set of hyper-parameters as in the other experiments. Because the distribution of scores across seeds is skewed towards higher values, we represent dispersion between seeds using the min-max interval over the 6 seeds (shaded area) instead of using the standard deviation; the solid line represents the median score over the seeds.

C.2. Complements to Section 5.3

Details on the environments The DeepMind Control Suite environments (Tassa et al., 2018) are control tasks with continuous action space $\mathcal{A} = [-1, 1]^m$. These tasks all involve simulated robotic systems and the reward functions are designed so as to guide the system for accomplish e.g. locomotion tasks. Typically, these robotic systems have relatively low-dimensional sensory recordings which summarize the environment states. To make the tasks more challenging, for observations, we take the third-person camera of the robotic system and use the image recordings as observations to the RL agent. These images are of dimension $64 \times 64 \times 3$.

Figures 9 to 12 present a comparison of MuZero and ALL on a subset of 4 of the medium-difficulty (Van de Wiele et al., 2020) DeepMind Control Suite (Tassa et al., 2018) tasks chosen for their relatively high-dimensional action space among these medium-difficulty problems ($n_{\text{dim}} = 6$). Figure 8 compare the score of MuZero and ALL after 100k learner steps on



(a) Alien (b) Asteroids (c) Beam rider

(d) Gravitar (e) Ms Pacman (f) Seaquest

Figure 7. Comparison of median score over 6 seeds of MuZero and ALL on six Atari games with 50 simulations per step. The shaded area correspond the the best and worst seeds.

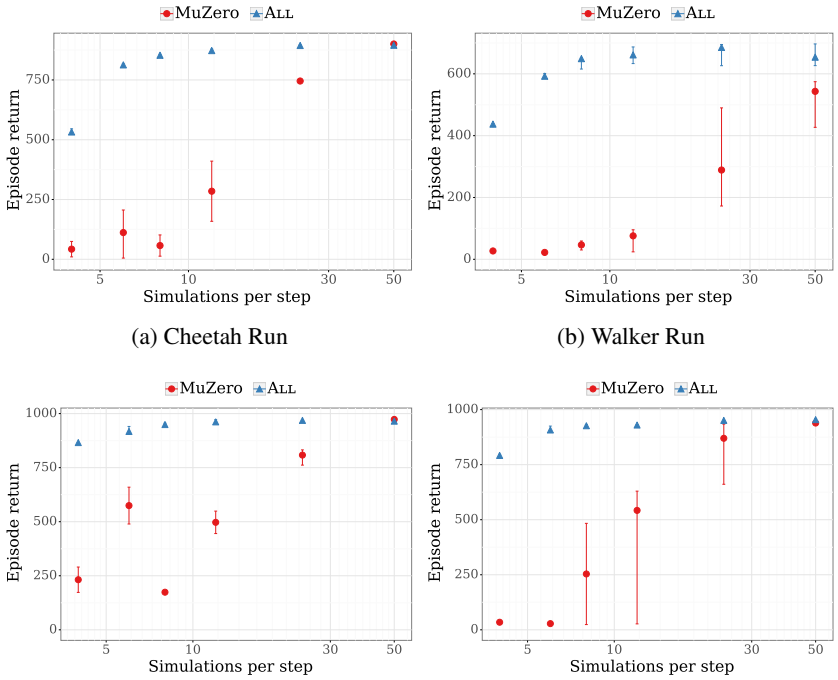
these four medium difficulty Control problems. These continuous control problems are cast to a discrete action formulation using the method presented in Appendix B.2; note that these experiments only use pixel renderings and not the underlying scalar states.

These curves present the median (solid line) and min-max interval (shaded area) computed over 3 seeds in the same settings as described in Appendix C.1. The hyper-parameters are the same as in the other experiments; no specific tuning was performed for the continuous control domain. The horizontal dashed line corresponds to the performance of the D4PG algorithm when trained on pixel observations only (Barth-Maron et al., 2018), as reported by (Tassa et al., 2018).

C.3. Complemantary experiments on comparison with PPO

Since we interpret the MCTS-based algorithms as regularized policy optimization algorithms, as a sanity check for the proposal's performance gains, we compare it with state-of-the-art proximal policy optimization (PPO) (Schulman et al., 2017). Since PPO is a near on-policy optimization algorithm, whose gradient updates are purely based on on-policy data, we adopt a lighter network architecture to ensure its stability. Please refer to the public code base (Dhariwal et al., 2017) for a review of the neural network architecture and algorithmic details.

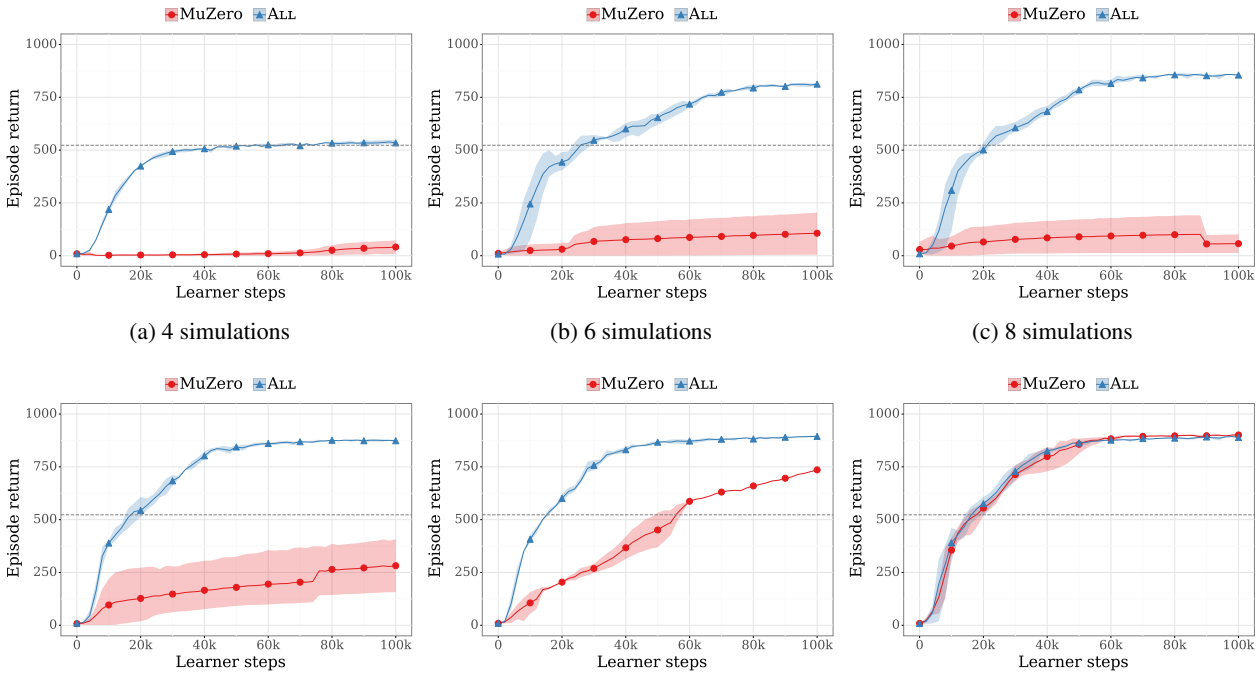
To assess the performance of PPO, we train with both state-based inputs and image-based inputs. State-based inputs are low-dimensional sensor data of the environment, which renders the input sequence strongly Markovian (Tassa et al., 2018). For image-based training, we adopt the same inputs as in the main paper. The performance is reported in Table 1 where each score is the evaluation performance of PPO after the convergence takes place. We observe that state-based PPO performs significantly better than image-based PPO, while in some cases it matches the performance of ALL. In general, image-based PPO significantly underperforms ALL.



(a) Cheetah Run (b) Walker Run

(c) Walker Stand (d) Walker Walk

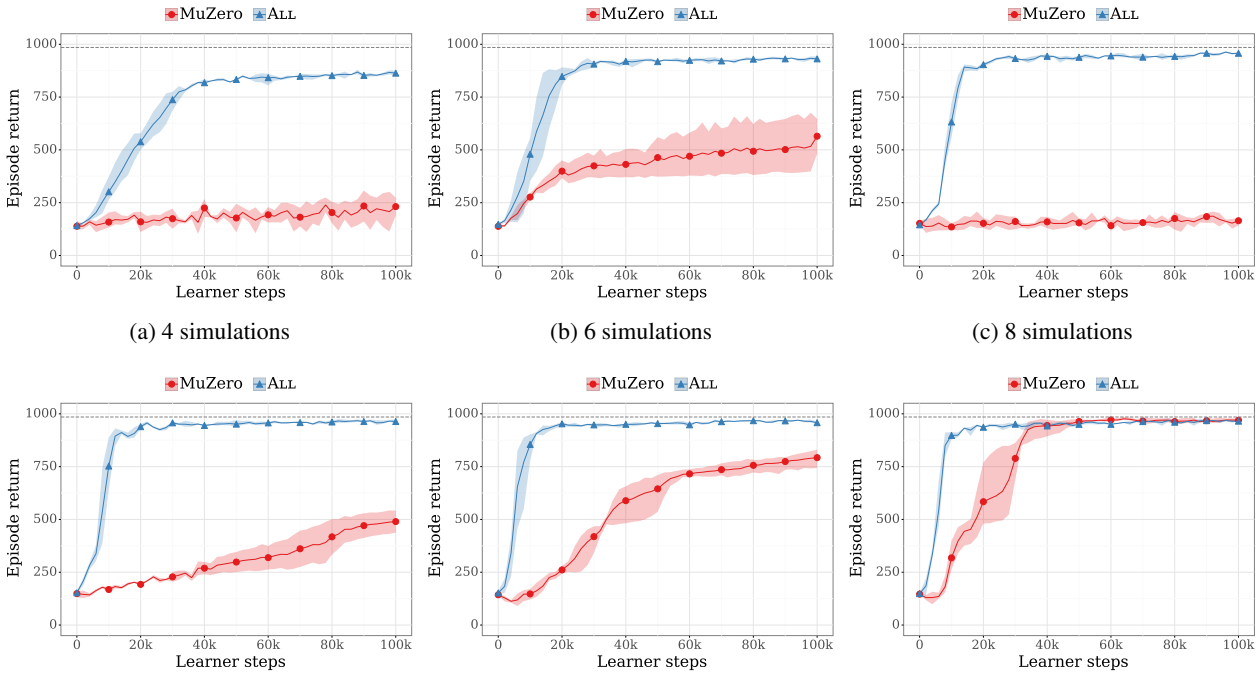
Figure 8. Score of MuZero and All on Continuous control tasks after 100k learner steps as a function of the number of simulations N_{sim}.



(a) 4 simulations (b) 6 simulations (c) 8 simulations

(d) 12 simulations (e) 24 simulations (f) 50 simulations

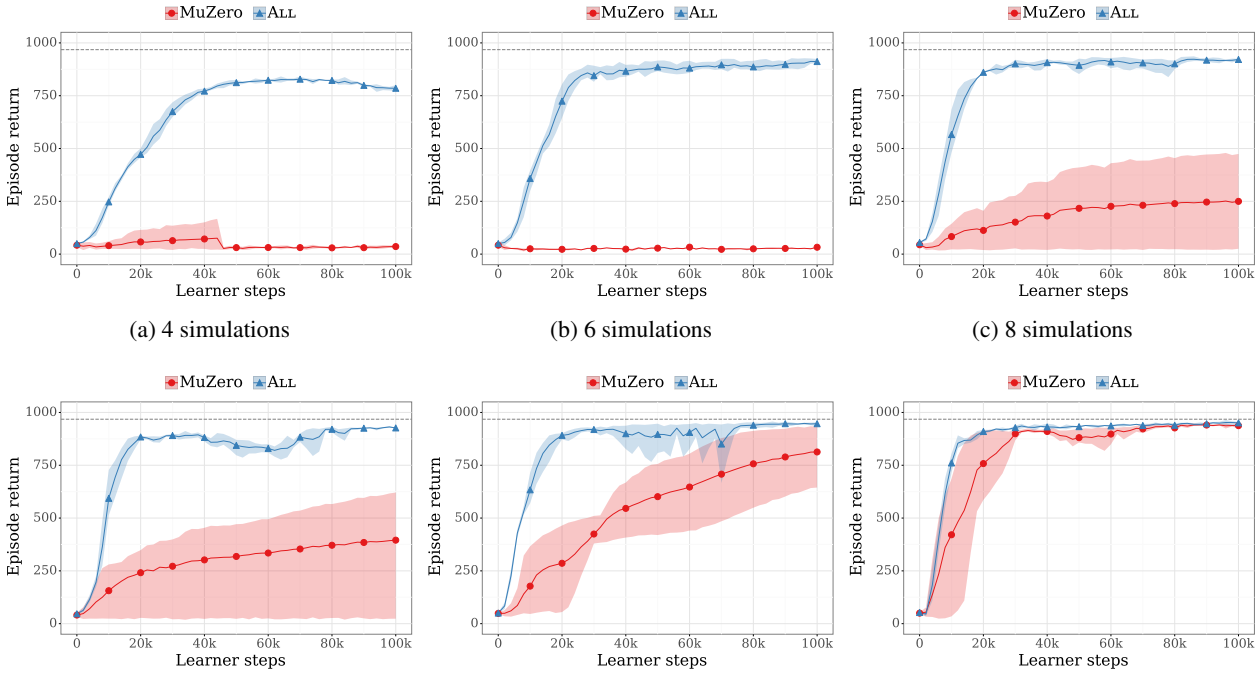
Figure 9. Comparison of MuZero and All on Cheetah Run.



(a) 4 simulations (b) 6 simulations (c) 8 simulations

(d) 12 simulations (e) 24 simulations (f) 50 simulations

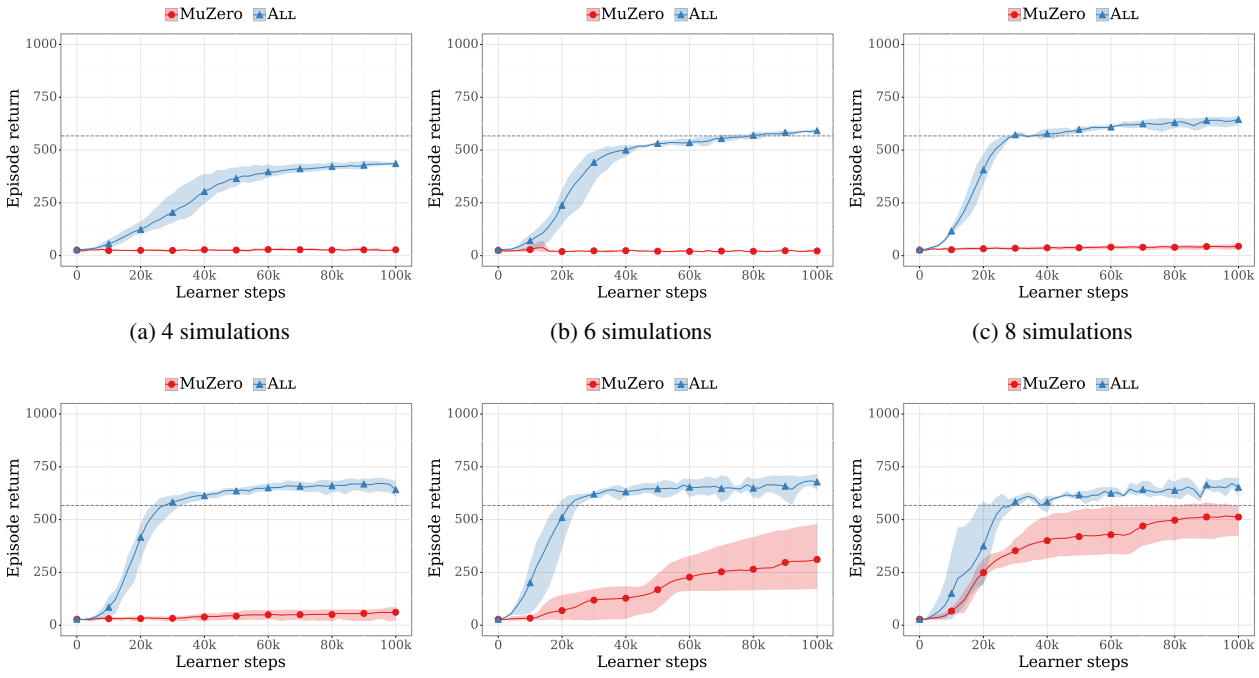
Figure 10. Comparison of MuZero and ALL on Walker Stand.



(a) 4 simulations (b) 6 simulations (c) 8 simulations

(d) 12 simulations (e) 24 simulations (f) 50 simulations

Figure 11. Comparison of MuZero and ALL on Walker Walk.



(a) 4 simulations (b) 6 simulations (c) 8 simulations

(d) 12 simulations (e) 24 simulations (f) 50 simulations

Figure 12. Comparison of MuZero and ALL on Walker Run.

Benchmarks	**PPO (state)**	**PPO (image)**	**MuZero (image)**	**ALL(image)**
WALKER-WALK	406	270	925	941
WALKER-STAND	937	357	959	951
WALKER-RUN	340	71	533	644
CHEETAH-RUN	538	285	887	882

Table 1. Comparison to the performance of PPO baselines on benchmark tasks. The inputs to PPO are either state-based or image-based. The performance is computed as the evaluated returns after the training is completed, averaged across 3 random seeds.

D. Derivations for Section 3

D.1. Proof of Proposition 1, Eq. 11 and Proposition 2.

We start with a definition of the f-divergence (Csiszár, 1964).

Definition 2 (f-*divergence*)**.** *For any probability distributions p and q on $\mathcal{A}$ and function $f : \mathbb{R} \to \mathbb{R}$ such that f is a convex function on $\mathbb{R}$ and $f(1) = 0$, the f-divergence D_f between p and q is defined as*

$$D_f(p, q) = \sum_{b \in \mathcal{A}} q(b) f\left(\frac{p(b)}{q(b)}\right) \tag{28}$$

Remark 1. *Let D_f be a f-divergence,*

(1) $\forall x, y\ D(x, y) \geq 0$ *(2)* $D(x, y) = 0 \iff x = y$ *(3) $D(x, y)$ is jointly convex in x and y.*

We states four lemmas that we formally prove in Appendix D.2.

Lemma 1.

$$\nabla_\pi \left(\mathbf{q}^T \pi - \lambda \cdot D_f[\pi, \pi_\theta]\right) = \mathbf{q} - \lambda \cdot f'\left(\frac{\pi}{\pi_\theta}\right) \tag{29}$$

Where π_θ is assumed to be non zero. We now restate the definition of $\hat{\pi}[a] \triangleq \frac{n_a+1}{N+|\mathcal{A}|}$.

Lemma 2.

$$\arg\max_a \left[\frac{\partial}{\partial n_a}\left(\mathbf{q}^T \pi - \lambda \cdot D_f[\hat{\pi}, \pi_\theta]\right)\right] = \arg\max_a \left[\mathbf{q_a} - \lambda \cdot f'\left(\frac{\hat{\pi}(a)}{\pi_\theta(a)}\right)\right] \tag{30}$$

Now we consider a more general definition of $\bar{\pi}$ using any f-divergence for some $\lambda_f > 0$ and assume $\pi_\theta > 0$,

$$\bar{\pi}_f \triangleq \arg\min_{\mathbf{y} \in \mathcal{S}} \mathbf{q}^T \mathbf{y} - \lambda_f D_f(\pi_\theta, \mathbf{y}). \tag{31}$$

We also consider the following action selection formula based on f-divergence D_f.

$$a_f^\star \triangleq \arg\max_a \left[q_a - \lambda_f \cdot f'\left(\frac{\hat{\pi}}{\pi_\theta}\right)\right], \quad \text{with.} \tag{32}$$

Lemma 3.

$$\arg\max_a \left[q_a - \lambda_f \cdot f'\left(\frac{\hat{\pi}(a)}{\pi_\theta(a)}\right)\right] = \arg\max_a \left[f'\left(\frac{\bar{\pi}_f(a)}{\pi_\theta(a)}\right) - f'\left(\frac{\hat{\pi}(a)}{\pi_\theta(a)}\right)\right]. \tag{33}$$

Lemma 4.

$$\hat{\pi}(a_f^\star) \leq \bar{\pi}(a_f^\star). \tag{34}$$

Applying Lemmas 2 and 4 with the appropriate function f directly leads to Proposition 1, Proposition 2, and Proposition 3. In particular, we use

$$\text{For AlphaZero:} \qquad f(x) = -\log(x) \tag{35}$$

$$\text{For UCT:} \qquad f(x) = 2 - 2\sqrt{x} \tag{36}$$

Algorithm	Function $f(x)$	Derivative $f'(x)$	Associated f-divergence	Associated action selection formula
—	$x \cdot \log(x)$	$\log(x) + 1$	$D_f(p, q) = \mathrm{KL}(p, q)$	$\arg\max_a q_a + \frac{c}{\sqrt{N}} \cdot \log\left(\frac{\pi_\theta(a)}{n_a+1}\right)$
UCT	$2 - 2\sqrt{x}$	$-\frac{1}{\sqrt{x}}$	$D_f(p, q) = 2 - 2\sum_{b \in \mathcal{A}} \sqrt{p_a \cdot q_a}$	$\arg\max_a q_a + c \cdot \sqrt{\frac{\pi_\theta}{n_a+1}}$
AlphaZero	$-\log(x)$	$-\frac{1}{x}$	$D_f(p, q) = \mathrm{KL}(q, p)$	$\arg\max_a q_a + c \cdot \pi_\theta \cdot \frac{\sqrt{N}}{n_a+1}$

D.2. Proofs of Lemmas 1 to 4

Proof of Lemma 1

Proof. For any action $a \in \mathcal{A}$ using basic differentiation rules we have

$$\big(\nabla_\pi\big(\mathbf{q}^T\pi - \lambda_f \cdot D_f[\pi,\pi_\theta]\big)\big)[a] = \frac{\partial}{\partial\pi_a}\left[\sum_{b\in\mathcal{A}}(q_b\cdot\pi_b) - \lambda_f\sum_{b\in\mathcal{A}}\pi_\theta(b)f\left(\frac{\pi_b}{\pi_\theta(b)}\right)\right] \tag{37}$$

$$= \frac{\partial}{\partial\pi_a}\left[\sum_{b\in\mathcal{A}}(q_b\cdot\pi_b)\right] - \lambda_f\cdot\frac{\partial}{\partial\pi_a}\sum_{b\in\mathcal{A}}\pi_\theta(b)f\left(\frac{\pi_b}{\pi_\theta(b)}\right) \tag{38}$$

$$= q_a - \lambda_f\cdot\pi_\theta(a)\cdot\frac{\partial}{\partial\pi_a}f\left(\frac{\pi_a}{\pi_\theta(a)}\right) \tag{39}$$

$$= q_a - \lambda_f\cdot f'\left(\frac{\pi_a}{\pi_\theta(a)}\right) = \left(\mathbf{q} - \lambda_f\cdot f'\left(\frac{\pi}{\pi_\theta}\right)\right)[a] \tag{40}$$

□

Proof of Lemma 2

Proof.

$$\frac{\partial}{\partial n_a}\big(\mathbf{q}^T\pi - \lambda_f\cdot D_f[\hat\pi,\pi_\theta]\big) = \frac{\partial}{\partial n_a}\left[\sum_{b\in\mathcal{A}}\left(q_b\cdot\frac{n_b+1}{|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c}\right) - \lambda_f\sum_{b\in\mathcal{A}}\pi_\theta(b)\cdot f\left(\frac{n_b+1}{(|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c)\cdot\pi_\theta(b)}\right)\right] \tag{41}$$

$$= \beta + \frac{q_a}{|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c} - \frac{\lambda_f}{|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c}f'\left(\frac{n_a+1}{(|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c)\cdot\pi_\theta(b)}\right) \tag{42}$$

$$= \beta + \frac{1}{|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c}\left(q_a - \lambda_f f'\left(\frac{\hat\pi(a)}{\pi_\theta(b)}\right)\right), \tag{43}$$

where $\beta = -\dfrac{\sum_{b\in\mathcal{A}}\left[q_b - \lambda_f f'\left(\frac{\hat\pi(b)}{\pi_\theta(b)}\right)\right]}{\big(|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c\big)^2}$ is independent of a. Also because $\frac{1}{|\mathcal{A}|+\sum_{c\in\mathcal{A}}n_c} > 0$ then

$$\arg\max_a \frac{\partial}{\partial n_a}\big(\mathbf{q}^T\pi - \lambda_f\cdot D_f[\hat\pi,\pi_\theta]\big) = \arg\max_a\left[\mathbf{q}_a - \lambda_f\cdot f'\left(\frac{\hat\pi(a)}{\pi_\theta(a)}\right)\right] \tag{44}$$

$$\tag{45}$$

□

Proof of Lemma 3

Proof. The Eq. 31 is a differentiable strictly convex optimization problem, its unique solution satisfies the KKT condition requires (see Section 5.5.3 of Boyd and Vandenberghe, 2004) therefore there exists $\alpha \in \mathbb{R}$ such that for all actions a,

$$\nabla_{\bar\pi}\big(\mathbf{q}^T\bar\pi - \lambda_f D_f(\pi_\theta,\bar\pi)\big) = \alpha\mathbb{1} \tag{46}$$

where $\mathbb{1}$ is the vector constant equal one: $\forall a\ \mathbb{1}_a = 1$. Using Lemma 1 setting π to $\bar\pi$ we get

$$\exists\alpha \quad q - \lambda_f\cdot f'\left(\frac{\bar\pi}{\pi_\theta}\right) = \alpha\mathbb{1}. \tag{47}$$

$$q - \lambda_f \cdot f'\left(\frac{\hat{\pi}}{\pi_\theta}\right) = q_a - \lambda_f \cdot \left(f'\left(\frac{\hat{\pi}}{\pi_\theta}\right) + f'\left(\frac{\bar{\pi}}{\pi_\theta}\right) - f'\left(\frac{\bar{\pi}}{\pi_\theta}\right)\right) \tag{48}$$

$$= \alpha \mathbb{1} + \lambda_f \cdot \left(f'\left(\frac{\bar{\pi}}{\pi_\theta}\right) - f'\left(\frac{\hat{\pi}}{\pi_\theta}\right)\right) \tag{49}$$

$$\arg\max\left[q - \lambda_f \cdot f'\left(\frac{\hat{\pi}}{\pi_\theta}\right)\right] = \arg\max\left[\alpha \mathbb{1} + \lambda_f \cdot \left(f'\left(\frac{\bar{\pi}}{\pi_\theta}\right) - f'\left(\frac{\hat{\pi}}{\pi_\theta}\right)\right)\right] \tag{50}$$

$$= \arg\max\left[f'\left(\frac{\bar{\pi}}{\pi_\theta}\right) - f'\left(\frac{\hat{\pi}}{\pi_\theta}\right)\right] \qquad \text{(because } \lambda_f > 0\text{)} \tag{51}$$

$$\tag{52}$$

□

Proof of Lemma 4

Proof. Since $\sum_a \hat{\pi}(a|x) = \sum_a \bar{\pi}(a|x) = 1$, there exists at least an action a_0 for which $0 \leq \hat{\pi}(a_0|x) \leq \bar{\pi}(a_0|x)$ then $0 \leq \frac{\hat{\pi}(a_0|x)}{\pi_\theta(a|x)} \leq \frac{\bar{\pi}(a_0|x)}{\pi_\theta(a|x)}$ as $\pi_\theta(a|x) > 0$. Because f is convex then f' is increasing and therefore

$$f'\left(\frac{\bar{\pi}(a_0|x)}{\pi_\theta(a_0|x)}\right) - f'\left(\frac{\hat{\pi}(a_0|x)}{\pi_\theta(a_0|x)}\right) \geq 0. \tag{53}$$

Now using Lemma 3

$$f'\left(\frac{\bar{\pi}(a_f^\star|x)}{\pi_\theta(a_f^\star|x)}\right) - f'\left(\frac{\hat{\pi}(a_f^\star|x)}{\pi_\theta(a_f^\star|x)}\right) \geq f'\left(\frac{\bar{\pi}(a_0|x)}{\pi_\theta(a_0|x)}\right) - f'\left(\frac{\hat{\pi}(a_0|x)}{\pi_\theta(a_0|x)}\right) \tag{54}$$

We put Equations (53) and (54) together

$$f'\left(\frac{\bar{\pi}(a_f^\star|x)}{\pi_\theta(a_f^\star|x)}\right) - f'\left(\frac{\hat{\pi}(a_f^\star|x)}{\pi_\theta(a_f^\star|x)}\right) \geq 0 \tag{55}$$

Finally we use again that f' is increasing and $\pi_\theta > 0$ to conclude the proof

$$\hat{\pi}(a_f^\star|x) \leq \bar{\pi}(a_f^\star|x) \tag{56}$$

□

D.3. Tracking property in the constant $\bar{\pi}$ case

Let π be some target distribution independent of the round $t \geq 0$. At each round t, starting from $t = 1$, an action $a_t \in \mathcal{A}$ is selected and for any $t \geq 0$, we define

$$p_t(a) \triangleq \frac{n_t(a_t) + 1}{|\mathcal{A}| + \sum_b n_t(b)},$$

where for any action $a \in \mathcal{A}$, $n_t(a)$ is the number of rounds the action a has been selected,

$$\forall a \in \mathcal{A} \quad n_t(a) \triangleq \sum_{i \leq t} \delta(a_t = a) \quad \text{ and } \delta(a_t = a) \triangleq 1 \text{ if and only if } a_t = a.$$

Proposition 5. *Assume that for all rounds $t \geq 1$, and for the chosen action $a_t \in \mathcal{A}$ we have*

$$p_t(a_t) \leq \pi(a_t). \tag{57}$$

Then, we have that

$$\forall a \in \mathcal{A}, t \geq 1 \quad |\pi(a) - p_t(a)| \leq \frac{|\mathcal{A}| - 1}{|\mathcal{A}| + t}.$$

Before proving the proposition above, note that $\mathcal{O}(1/t)$ is the best approximation w.r.t. t, since for any integer $k \geq 0$, taking $\pi(a) = (\frac{1}{2} + k)/(|\mathcal{A}| + t)$, we have that for all $n \geq 0$,

$$\left|\pi(a) - \frac{n+1}{|\mathcal{A}|+t}\right| \geq \frac{1}{2}\frac{1}{|\mathcal{A}|+t},$$

which follows from the fact that $\forall k, n \in \mathbb{N}$, $\left|\frac{1}{2} + k - (n+1)\right| = \left|k - n - \frac{1}{2}\right| \geq \frac{1}{2}$.

Proof. By induction on the round t, we prove that

$$\forall t \geq 1, a \in \mathcal{A} \quad p_t(a) \leq \pi(a) + \frac{1}{|\mathcal{A}|+t}. \tag{58}$$

At round $t = 1$, Eq. 58 holds as for any action a, $n_t(a) \geq 0$ therefore $p_t(a) \leq 1$. Now, let us assume that Eq. 58 holds for some $t \geq 1$. We have that for all a,

$$\frac{1 + n_t(a)}{|\mathcal{A}| + \sum_b n_t(b)} \leq \pi(a) + \frac{1}{|\mathcal{A}|+t}.$$

Note that at each round, there is exactly one action chosen and therefore, $\sum_b n_t(b) = t$. Furthermore, for $a' \neq a_{t+1}$, we have that $n_{t+1}(a') = n_t(a')$, since a' has not been chosen at round $t+1$. Therefore, for $a' \neq a_{t+1}$,

$$p_{t+1}(a') = \frac{n_{t+1}(a')+1}{|\mathcal{A}|+t+1} = \frac{n_t(a')+1}{|\mathcal{A}|+t+1} \leq \frac{|\mathcal{A}|+t}{|\mathcal{A}|+t+1}p_t(a') \leq \frac{|\mathcal{A}|+t}{|\mathcal{A}|+t+1}\left(\pi(a') + \frac{1}{|\mathcal{A}|+t}\right) \leq \pi(a') + \frac{1}{|\mathcal{A}|+t+1}.$$

Now, for the chosen action, $n_{t+1}(a_{t+1}) = n_t(a_{t+1}) + 1$. Using our assumption stated in Eq. 57, we have that

$$p_{t+1}(a_{t+1}) = \frac{n_{t+1}(a_{t+1})+1}{|\mathcal{A}|+t+1} = \frac{n_t(a_{t+1})+1+1}{|\mathcal{A}|+t+1} \leq \frac{n_t(a_{t+1})+1}{|\mathcal{A}|+t+1} + \frac{1}{|\mathcal{A}|+t+1} \leq \pi(a_{t+1}) + \frac{1}{|\mathcal{A}|+t+1},$$

which concludes the induction. Next, we compute a lower bound. For any action $a \in \mathcal{A}$ and round $t \geq 1$,

$$p_t(a) = 1 - \sum_{b \neq a} p_t(b) \geq 1 - \sum_{b \neq a}\left(\pi(b) + \frac{1}{|\mathcal{A}|+t}\right) = \left(1 - \sum_{b \neq a}\pi(b)\right) - \sum_{b \neq a}\frac{1}{|\mathcal{A}|+t} = \pi(a) - \frac{|\mathcal{A}|-1}{|\mathcal{A}|+t}.$$

We have for any action $a \in \mathcal{A}$,

$$\pi(a) - \frac{|\mathcal{A}|-1}{|\mathcal{A}|+t} \leq p_t(a) \leq \pi(a) + \frac{1}{|\mathcal{A}|+t}.$$

Since when $|\mathcal{A}| = 1$, then by definition, $p_t(a) = \pi(a) = 1$ and for all rounds $t \geq 1$, we get

$$||\pi - p_t||_\infty \leq \frac{|\mathcal{A}|-1}{|\mathcal{A}|+t}.$$

□