IEEE Robotics & Automation Magazine (RAM) paper, presented at ICRA 2026, Vienna, Austria. Cite as RAM paper.

Reinforcement Learning for Robust Athletic Intelligence: Lessons from the 2nd "AI Olympics with RealAIGym" Competition

Felix Wiebe[1], Niccolò Turcato[2], Alberto Dalla Libera[2], Jean Seong Bjorn Choe[3], Bumkyu Choi[3], Tim Lukas Faust[6], Habib Maraqten[6], Erfan Aghadavoodi[4], Marco Cali[2], Alberto Sinigaglia[2], Giulio Giacomuzzo[2], Ruggero Carli[2], Diego Romeres[5], Jong-kook Kim[3], Gian Antonio Susto[2], Shubham Vyas[1], Dennis Mronga[1], Boris Belousov[4], Jan Peters[4,6,9,10], Frank Kirchner[1,7] and Shivesh Kumar[1,8]

***Abstract*— In robotics many different approaches ranging from classical planning over optimal control to reinforcement learning (RL) are developed and borrowed from other fields to achieve reliable control in diverse tasks. In order to get a clear understanding of their individual strengths and weaknesses and their applicability in real-world robotic scenarios it is important to benchmark and compare their performances not only in a simulation but also on real hardware. The '2nd AI Olympics with RealAIGym' competition was held at the IROS 2024 conference to contribute to this cause and evaluate different controllers according to their ability to solve a dynamic control problem on an underactuated double pendulum system (Fig. 1) with chaotic dynamics. This paper describes the four different RL methods submitted by the participating teams, presents their performance in the swing-up task on a real double pendulum, measured against various criteria, and discusses their transferability from simulation to real hardware and their robustness to external disturbances.**

I. Introduction

There are many frameworks for comparing control methods in simulation environments (e.g., [1], [2]) but seldom provide a standardized real robot environment which is open and easily reproducible such as [3] with a three-end-effector robot for manipulation tasks. The RealAIGym project [4] focuses on simple, low degree of freedom systems capable of dynamic behaviors, such as a simple pendulum [5], AcroMonk [6] and the dual purpose double pendulum [7] used in this competition. With the software and hardware being open source and a user friendly Python API, RealAIGym intends to be an open benchmarking platform for comparing control methods on real hardware. This is the second iteration of this competition following the first held at the IJCAI 2023 conference [8].

The four best performing control algorithms submitted to this competition are all based on different variants of reinforcement learning (RL). This paper presents these methods, which have been chosen or developed by the teams for this task, and their results in the context of this competition. The competition is designed to evaluate the control methods' robustness on real hardware, requiring them to successfully bridge the simulation reality gap and to be robust to unknown external disturbances. RL for real-world robotic applications still faces open challenges such as sample efficiency and training stability, long-horizon tasks, principled approaches, leveraging foundation models, real-world learning and real-world benchmarking [9]. The latter two are addressed in this competition by the four finalist controllers with different algorithmic approaches (Table I), which were submitted and tuned by the participating teams and compared by means of the competitions' objective.

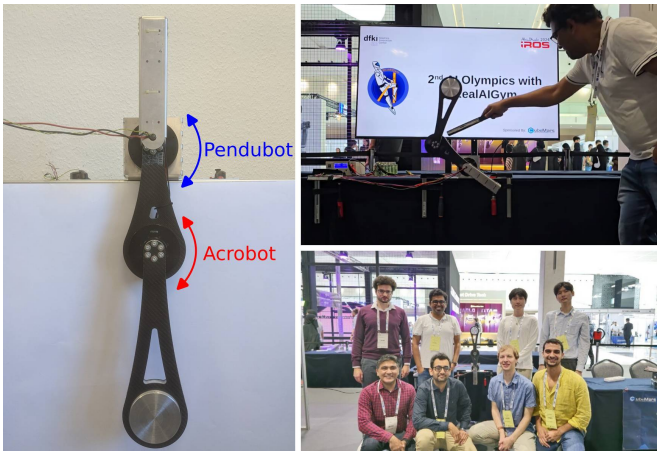


Fig. 1: Double pendulum hardware (left), the pendulum being disturbed with a stick at the conference site (top right) and participating teams (bottom right) at IROS 2024.

TABLE I: Algorithms submitted to the competition.

	Sim. usage	Model learning	Policy opt.	Model repr.
MC-PILCO	no sim.	model-based	offline	RBF network
AR-EAPO	zero-shot	model-free	on-policy	MLP
EvolSAC	zero-shot	model-free	off-policy	MLP
HistorySAC	few-shot	model-free	off-policy	CNN

II. Competition Rules

1) Test system: The competition was conducted with the dual purpose double pendulum system introduced in [7]

[1]Robotics Innovation Center, German Research Center for Artificial Intelligence (DFKI), Germany
[2]Department of Information Engineering, University of Padova, Italy
[3]Korea University, Seoul, South Korea
[4]Systems AI for Robot Learning, German Research Center for Artificial Intelligence (DFKI), Germany
[5]Mitsubishi Electric Research Lab (MERL), USA
[6]Technical University of Darmstadt, Germany
[7]University of Bremen, Germany
[8]Chalmers University of Technology, Sweden
[9]Center for Cognitive Science, Germany
[10]Hessian.AI, Germany

which can be operated as an acrobot or pendubot without changing the hardware. The pendulum is built with two quasi-direct drives (AK80-6 t-motors from Cubemars[1]) and ready-made carbon fiber sandwich panels as links. The used system had an attached mass of $0.5\,\text{kg}$ and link lengths of 0.2m and 0.3m. Model parameters identified with least squares optimization were provided to the participants, but the teams were free to do their own system identification.

2) Task: For the competition acrobot and pendubot were treated as separate challenges and evaluated in separate tracks. The task on both systems is to swing up the pendulum from the free hanging position to the upright position and stabilize it there. Each trial lasts for $10\,\text{s}$ and is considered successful if, in the end, the end-effector is at a height above the threshold of $0.45\,\text{cm}$ above the base. To test the controllers' robustness, external perturbations are applied on both motors during the execution.

3) Procedure: The competition was carried out in three phases: (i) A simulation phase, (ii) a remote hardware phase and (iii) an onsite phase. In the simulation phase the participants were asked to develop a controller for the swing-up and balance task on the acrobot and/or pendubot system and integrate it into the double pendulum toolkit on github[2]. After the simulation phase, the four best performing controllers were advanced to the next phase. In the remote hardware phase, the teams got the opportunity to login to a PC in our lab and remotely operate the pendulum hardware to test their controllers and collect position, velocity and torque data from the real system along with live video feed. The third phase took place on site at the IROS 2024 conference, where the teams were able to fine-tune their controllers before the final evaluation.

4) Evaluation: The evaluation of the controllers is based on different criteria from which two scores, the *performance score* and the *robustness score*, are calculated.

The performance score measures how fast, smooth and efficient the controller solves the swing-up and balance task and is calculated as

$$S_p = c_{succ}\left[1 - \frac{1}{5}\sum_{i\in\{t,e,\tau_c,\tau_s,v\}} \tanh\left(w_i c_i\right)\right] \tag{1}$$

based on the swing-up success $c_{succ} \in \{0, 1\}$, the swing-up time c_t, the used energy c_e, the torque cost c_{τ_c}, the torque smoothness c_{τ_s} and the velocity cost c_v. These quantities are normalized with the weights in Table II and scaled with the $\tanh$ function to the interval $[0, 1]$.

The robustness score is a simulation-only score and measures the controllers' robustness to unmodeled effects:

$$S_r = 1 - \frac{1}{6}\sum_{i\in\{m,v_n,\tau_n,\tau_r,d,p\}} c_i \tag{2}$$

with the criteria model inaccuracies c_m, measurement noise $c_{v,\text{n}}$, torque noise $c_{\tau,\text{n}}$, torque response $c_{\tau,\text{r}}$ and time delay c_d (all adopted from [7]) as well as random perturbations c_p which are randomly generated sequences of Gaussian perturbations added to the applied torque. For each criterion the severity of the external influence is varied in $N = 21$ steps (for the model inaccuracies for each independent model parameter) and the c_i are the percentage of failed swing-ups under these circumstances. For the perturbation criterion 50 random perturbations profiles are generated and evaluated.

The simulation phase controllers are evaluated with both, the performance score S_p and the robustness score S_r. For the hardware phases, the performance score, S_p, evaluates and ranks the controllers. During the experiments, the controllers had to showcase their robustness by reacting to unknown external perturbations consisting of Gaussian torque profiles on both motors at random times. The teams were allowed to use up to $0.5\,\text{Nm}$ torque on the passive joint for friction compensation. Each controller was tested in 10 trials and the final score is the average of the individual scores.

TABLE II: Weights for the performance score (1).

	w_t	w_E	w_{τ_c}	w_{τ_s}	w_v
Simulation	$\pi/20$	$\pi/60$	$\pi/20$	10π	$\pi/400$
Hardware	$\pi/20$	$\pi/60$	$\pi/100$	$\pi/4.0$	$\pi/400$

III. Algorithms

Four teams advanced to the final round of the competition and tested their control methods on the real hardware. This section briefly introduces the four methods and how they approached the task to improve robustness.

A. MC-PILCO

MC-PILCO (Monte Carlo - Probabilistic Inference for Learning COntrol) [10] is a model-based policy gradient RL algorithm. MC-PILCO relies on Gaussian Processes (GPs) to learn the system dynamics from data, and updates the policy by means of particle-based policy gradient optimization. Let $\boldsymbol{x}_t$ and $\boldsymbol{u}_t$ be, respectively, the state and input of the system at step t. A cost function $c(\boldsymbol{x}_t)$ encodes the task to solve. In the case of regulation tasks, like the ones of this competition, $c(\boldsymbol{x}_t)$ expresses the distance from the goal state $\boldsymbol{x}_g$. In this work, we adopted the exponentially saturated distance used also in [10] and [11] for the double pendulum swing-up task. A policy $\pi_{\boldsymbol{\theta}} : \boldsymbol{x} \to \boldsymbol{u}$ parametrized by $\boldsymbol{\theta}$ selects the inputs applied to the system. The objective is to find the policy parameters $\boldsymbol{\theta}^*$ which minimize the cumulative expected cost:

$$J(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathbb{E}[c(\boldsymbol{x}_t)], \quad x_0 \sim p(\boldsymbol{x}_0). \tag{3}$$

MC-PILCO iterates attempts to solve the objective task, also called trials. Each trial is composed of three main steps: (i) model learning, (ii) policy update, and (iii) policy execution. At the first trial, model learning is executed with data collected with an exploration policy, e.g., an exploration policy or a baseline controller. Fig. 2 illustrates the main details of each phase. In the following, we briefly describe

[1] https://www.cubemars.com/

[2] https://github.com/dfki-ric-underactuated-lab/double_pendulum

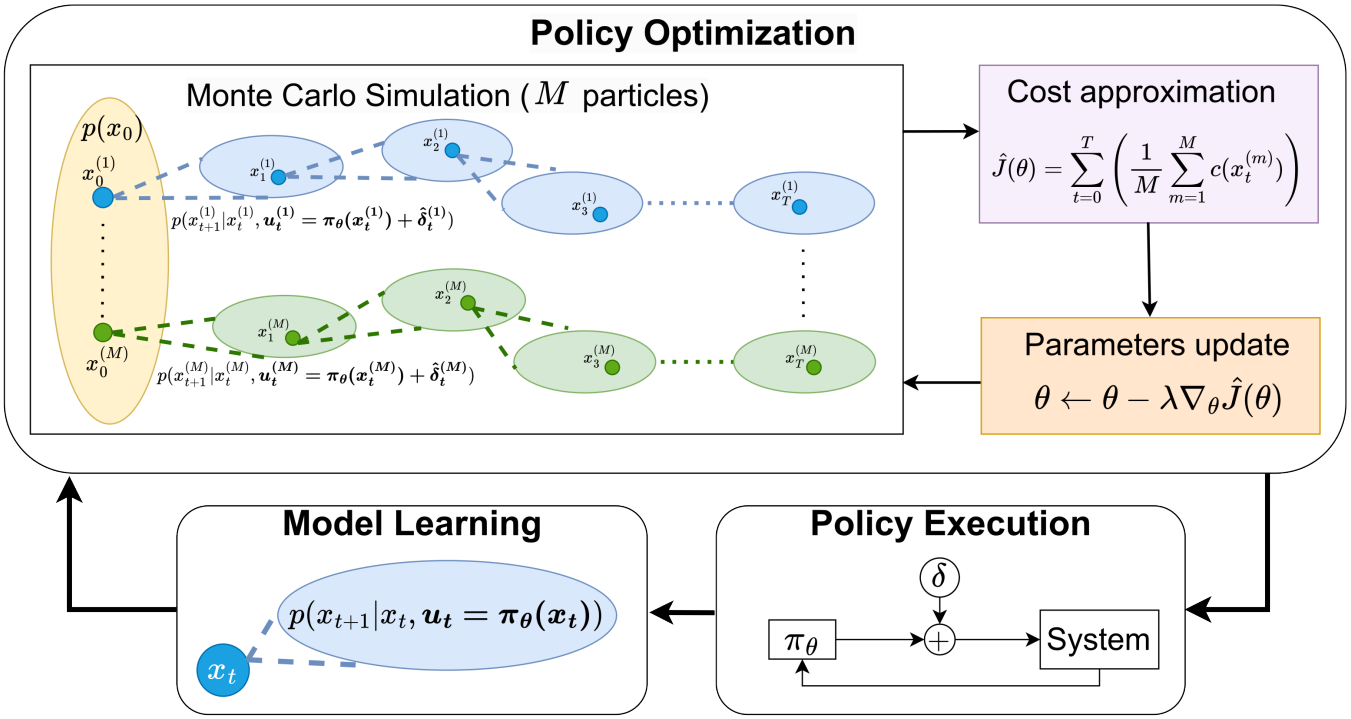


Fig. 2: Schematic representation of the robust MC-PILCO algorithm. Each episode is composed of (i) model learning, (ii) policy update and (iii) policy execution.

the model learning and policy optimization implemented for this competition.

Model learning: In this phase, previous experience is used to derive a one-step-ahead dynamics model, namely, a model that maps the tuple $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ in the next state $\boldsymbol{x}_{t+1}$. MC-PILCO relies on Gaussian process regression, thus obtaining a stochastic model. By the properties of GPs, $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$, the posterior distribution of $\boldsymbol{x}_{t+1}$ given data and $(\boldsymbol{x}_t, \boldsymbol{u}_t)$ is Gaussian, with mean and variance in closed-form expressions.

Policy optimization: MC-PILCO implements a gradient-based optimization of $\boldsymbol{\theta}$ to minimize the cost in (3). The expectation in (3) is computed w.r.t. the distribution induced by the initial state distribution $p(x_0)$, $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$ returned by the model learning phase, and the current policy parameters $\boldsymbol{\theta}$. Notice that this expectation involves long-term-state distributions that can not be computed in closed form. Then, MC-PILCO approximates (3) by Monte Carlo sampling. At each gradient descent step, the algorithm simulates the evolution of M particles, by sampling from $p(x_0)$ and $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$, with $\boldsymbol{u}_t = \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t)$. Finally, the algorithm approximates (3) with the particles' cumulative cost sample mean. The gradient is obtained by backpropagation and used to update $\boldsymbol{\theta}$.

For the sake of the competition, we modified the particles' sampling strategy to increase the policy's robustness to input disturbances. At each Monte Carlo simulation, for each particle $m = 1, \ldots, M$, we compute a disturbance profile $\hat{\boldsymbol{\delta}}_t^{(m)}$, , $t = 0, \ldots, T$, sampled from the known disturbance profile distribution. Subsequently, the next particle state at time $t+1$ is sampled from $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t) + \hat{\boldsymbol{delta}}_t^{(m)})$ instead from $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \pi_{\boldsymbol{\theta}}(\boldsymbol{x}_t))$. Fig. 2 visualizes the particles' simulation designed for this competition. By considering input disturbances in the particles' simulation, the algorithm promotes the derivation of policies robust to such effects.

B. AR-EAPO

Average-Reward Entropy Advantage Policy Optimization (AR-EAPO) [12] is a model-free RL algorithm that extends Entropy Advantage Policy Optimization (EAPO) [13], a maximum entropy (MaxEnt) on-policy actor-critic algorithm, into the average-reward setting.

AR-EAPO tackles the challenge of applying model-free RL to this complex problem by formulating the system as a recurrent Markov Decision Process (MDP), where every state is reachable from every other state in a finite number of steps. With the formulation, instead of using discounted cumulative rewards, AR-EAPO optimizes for long-term average rewards. This approach enables the use of straightforward reward functions aiming for long-term optimality rather than sophisticated reward functions for discounted settings. Here, the proposed solution utilizes a simple quadratic cost function.

Specifically, the objective of AR-EAPO is to find the optimal policy π^* that maximizes the entropy regularized (soft) *gain* $\tilde{\rho}^\pi$ – the sum of expected average reward ρ^π and average entropy $\rho^\pi_{\mathcal{H}}$ for a stationary policy π:

$$\tilde{\rho}^\pi(s) \coloneqq \lim_{T\to\infty} \frac{1}{T} \mathbb{E}_{\substack{s_0=s,\\ s_t\sim\pi}} \left[\sum_{t=0}^{T-1} r_t - \tau \log \pi(a_t|s_t) \right], \quad (4)$$

where s_t is the state of the MDP at time t, a_t is the action at time t, r_t is the reward received at time t, and τ is the

temperature. We omit the dependency on s in (4) since the gain is independent of the start state in unichain MDPs.

Building upon EAPO's separation of reward and entropy objectives, we define the reward vector $\tilde{\mathbf{r}}^\pi(s,a) := [r(s,a), -\tau \log \pi(a|s)]^\intercal$. The soft gain (4) can then be expressed as $\tilde{\rho}^\pi =: \mathbf{1}^\intercal \tilde{\boldsymbol{\rho}}^\pi$ where $\tilde{\boldsymbol{\rho}}^\pi = [\rho^\pi, \rho^\pi_{\mathcal{H}}]^\intercal$ consists of the long-term averages of each reward component. This formulation explicitly treats the reward gain ρ^π and the entropy gain $\rho^\pi_{\mathcal{H}}$ as separate quantities that are combined to form the overall objective (4).

The concept of bias value functions $\tilde{\mathbf{v}}^\pi(s)$ is then introduced as the solutions to the Bellman policy expectation equations [14]:

$$\tilde{\mathbf{v}}^\pi(s) + \tilde{\boldsymbol{\rho}}^\pi = \mathop{\mathbb{E}}_{a\sim\pi}\left[\tilde{\mathbf{r}}^\pi(s,a) + \mathbb{E}_{s'}[\tilde{\mathbf{v}}^\pi(s')]\right], \qquad (5)$$

which simultaneously captures both the reward and entropy bias functions. We finally define the advantage functions $\tilde{\mathbf{A}}^\pi(s,a) := \tilde{\mathbf{r}}^\pi(s,a) - \tilde{\boldsymbol{\rho}}^\pi + \mathbb{E}_{s'}[\tilde{\mathbf{v}}^\pi(s')] - \tilde{\mathbf{v}}^\pi(s)$.

The gain approximations at iteration k, denoted $\tilde{\boldsymbol{\rho}}^k$, are updated using the advantage estimates from samples at iteration k: $\tilde{\boldsymbol{\rho}}^{k+1} \leftarrow \tilde{\boldsymbol{\rho}}^k + \eta \mathbb{E}_{i\sim\mathcal{D}}[\tilde{\mathbf{A}}^\pi(s_i,a_i)]$, where $\mathcal{D}$ represents the collected trajectories at the iteration, and η is the step size hyperparameter.

In practice, AR-EAPO samples state-action pairs (s_t, a_t) from rollout trajectories $\mathcal{D}$ using a parameterized policy π_θ and estimates the advantage functions $\tilde{\mathbf{A}}^\pi(s_t,a_t)$ using generalized advantage estimation (GAE) [15] with hyperparameters $\boldsymbol{\lambda}$. The algorithm employs a critic parameterized by ϕ to approximate the bias functions $\tilde{\mathbf{v}}^\pi(s_t)$, while the gain estimates $\tilde{\boldsymbol{\rho}}^\pi$ are iteratively updated according to the advantage-based rule described above. This approach effectively extends the well-established proximal policy optimization (PPO) [16] to the average-reward MaxEnt setting, maintaining PPO's stability while optimizing for long-term average performance. Fig. 3 summarizes the complete algorithm.

In the double pendulum system, AR-EAPO demonstrates an interesting learning pattern. While the cost objective drives the controller to develop an efficient swing-up policy, the entropy component prevents the pendulum from remaining stationary at the uppermost position (Fig. 3). Instead, it encourages movement toward lower positions where average entropy is higher. Therefore, through these endless swing-ups and downs, the average-reward MaxEnt RL formulation naturally promotes diverse trajectories, thereby achieving a robust control policy.

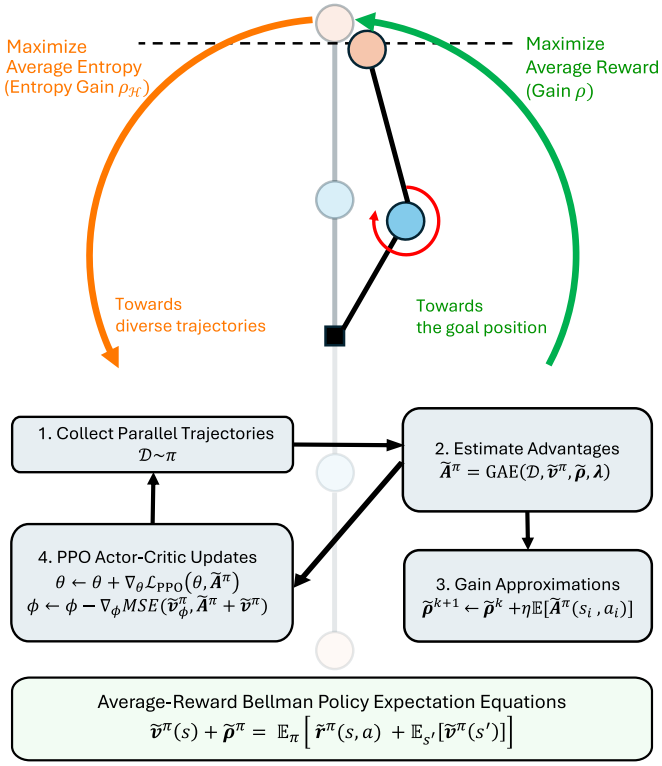


Fig. 3: Illustration of AR-EAPO in the double pendulum system: learning efficient and robust control in an infinite loop of swing-ups while balancing the cost and the entropy.

C. EvolSAC

The evolutionary soft actor critic (EvolSAC) algorithm integrates model-free deep RL with evolutionary strategies to optimize control performance in both simulation and real hardware stages. EvolSAC is presented in more detail in [17].

The core algorithm is Soft Actor-Critic (SAC) [18], a state-of-the-art model-free RL algorithm designed for continuous action spaces. SAC optimizes a stochastic policy by maximizing both the expected reward and the entropy of the policy, which promotes exploration and robustness. SAC's objective function is defined as:

$$J(\pi) = \mathbb{E}_{s_t,a_t\sim\pi}\left[\sum_t \gamma^t \left(r(s_t,a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t))\right)\right], \qquad (6)$$

where γ is the discount factor, $r(s_t,a_t)$ is the reward function, α is a temperature parameter that weighs the importance of the entropy term $\mathcal{H}$, encouraging the policy to explore a wide range of actions.

The initial stage of the methodology involves training the SAC agent with a physics-inspired surrogate reward function. This function is designed to approximate the complex objectives of the competition, which includes factors like swing-up success, energy consumption, and torque smoothness. The surrogate reward function aims to guide the agent towards achieving the swing-up task while managing the energy costs effectively:

$$R(s,a) = \begin{cases} V + \alpha[1+\cos(\theta_2)]^2 - \beta T \\ \qquad -\rho_1 a^2 - \phi_1 \Delta a & \text{, if } y > y_{th} \\ V - \rho_2 a^2 - \phi_2 \Delta a - \eta ||\dot{s}||^2 & \text{, otherwise} \end{cases}$$

where $\alpha, \beta, \phi, \eta, \rho$ are hyperparameters used to control the trade-offs.

To address the challenge of optimizing a policy for both performance and robustness, the methodology incorporates evolutionary strategies in the later stages of training. Evolutionary strategies are gradient-free optimization methods that are particularly effective in scenarios where the landscape

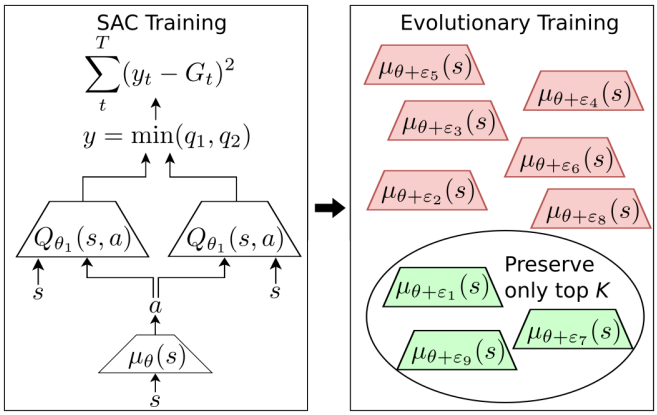


Fig. 4: Summary of training for EvolSAC. Left: SAC training for optimal policy, right: Evolutionary selection of high scoring and robust policies.

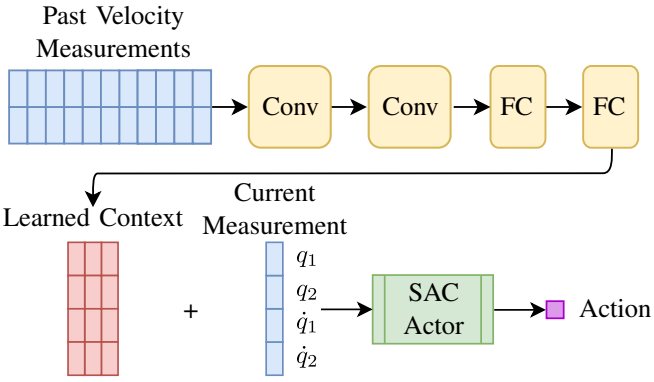


Fig. 5: The model architecture for encoding the history into a context representation in HistorySAC. A sequence of past velocity measurements is passed through convolutional (Conv) and fully-connected (FC) layers, and the output is attached to the current measurement before being passed to the actor and critic in SAC.

of the objective function is rugged or noisy. One specific strategy used is the Separable Natural Evolution Strategy (SNES) [19], which updates mutation strengths using a log-normal distribution, allowing efficient exploration of the parameter space:

$$\begin{bmatrix} \sigma_{\text{new},i} \\ \theta_{\text{new},i} \end{bmatrix} = \begin{bmatrix} \sigma_{\text{old},i} \exp\left(\tau \mathcal{N}(0,1) + \tau' \mathcal{N}(0,1)_i\right) \\ \theta_{\text{old},i} + \sigma_{\text{new},i} \mathcal{N}(0,1)_i \end{bmatrix} \tag{7}$$

where τ, τ' are learning rates controlling the mutation rate.

SNES is used to fine-tune the policy obtained from SAC, directly optimizing against the competition's score function defined in (1). This two-step process, initial training with SAC followed by fine-tuning with SNES, ensures that the developed controller is not only effective in achieving the desired task but also robust against various disturbances and model inaccuracies. The overall training strategy is illustrated in Fig. 4.

D. HistorySAC

This section proposes HistorySAC, a solution for the challenge that combines the Soft Actor-Critic algorithm [18] with a method for learning temporal features from past velocity observations.

Inferring system parameters of a chaotic system, such as the masses of both links, is challenging by only including the information of a transition between two states. In contrast to the original implementation of SAC, we aim to utilize measurements that reach further into the past, i.e. we incorporate a sequence of multiple past observations of velocities into the computation of our actor and critic networks. The velocities of eleven previous and the current time step are encoded in a learned temporal context (see 5) by leveraging convolutional and linear layers. The learned context is subsequently concatenated with the current state before feeding it into the actor and critic networks respectively (Fig. 5). For each encoding the context, we use two 1-dimensional convolutional layers with a kernel size of five and an output size of twelve, followed by two linear layers with a width of 256, where the first uses a ReLU and the second uses a tanh activation function. Kernel and layer sizes were found empirically. The underlying reinforcement learning algorithm is the Stable-Baselines3 implementation of SAC [1], but with a layer size of 1024 instead of the original 256 for the fully-connected layers. It is important to note that the networks of actor and critic do not share parameters.

The training uses the reward function

$$\begin{aligned} R_2(s,a) = &-0.05 \cdot \left((q_1 - \pi)^2 + q_2^2\right) \\ &- \Big[0.02 \cdot \left(\dot{q}_1^2 + \dot{q}_2^2\right) + 0.25 \cdot \left(a^2 + 2|a|\right) \\ &+ 0.02 \cdot \left|\frac{a - a_{\text{prev}}}{dt}\right| + 0.05 \cdot (\dot{q}_i \cdot a)\,\beta\Big], \end{aligned}$$

with the state s being the angles of both joints q_1 and q_2, and their derivatives. A regularization term (squared brackets) is subtracted which includes the angular velocities, the agent's action a and the previous action a_{prev}. For pendubot, β was set to $\beta = 0.1$ and $\dot{q}_i = \dot{q}_1$ and for acrobot $\beta = 0.025$ and $\dot{q}_i = \dot{q}_2$. In this application it turned out that providing the agent with negative values (punishments) instead of positive values (rewards) leads to better performance and higher robustness. Having the highest possible value of the reward function at 0 makes learning the Q-function much more stable than having an optimal value that is positive, since a successful swing-up policy does not suddenly change the Q-values of all previous states and thus prevents policy degradation after learning successful swing-ups.

Successful swing-up attempts on the real system using a policy that was purely trained in simulation require an accurate identification of the physical parameters of the double pendulum model. Differential Evolution [20], which is a gradient-free optimization algorithm, was used to optimize for the physical parameters of the simulated double pendulum θ_m to minimize the sim-to-real gap. The cost function for the optimization was chosen to be an importance-weighted squared error between simulated and

real trajectories:

$$J(\theta_m) = \sum_{i=1}^{N_{\text{traj}}} \sum_{t=1}^{T_i} \sum_{j=1}^{m} \Big(1 - \frac{0.5\,(t-1)}{T_i - 1}\Big) \times \Big(x^{(i)}_{\text{sim},j}[t;\theta_m] - x^{(i)}_{\text{real},j}[t]\Big)^2. \tag{8}$$

Here $m = 4$ the number of state indices, T_i is the time steps in a trajectory, and N_{traj} the number of trajectories.

Data from the real system was collected using swing-up policies that were trained in simulation for acrobot and pendubot. The torque time series from these real system trajectories is then used to create trajectories on the simulated system, resulting in pairs of trajectories for a single torque curve, $x^{\text{real}}_{1:T}$ and $x^{\text{sim}}_{1:T}$. The first round brackets in (8) denote a time-dependent weight. The weighting decreases linearly over the trimmed trajectory from 1 to 0.5, so that later points in the trajectory with a larger error accumulation have less impact. Since a solution for the optimization problem becomes increasingly difficult to find with a longer trajectory, only the first 1.5 seconds of each trajectory are considered in the optimization. For the training process, a multi-environment strategy, incorporating multiple solutions of the previous system identification process, prevented overfitting on a single environment.

IV. Results

The specifications for the training process of the four RL methods are listed in table III and the computed scoring criteria and final scores for all controllers are visualized in Fig. 6. Data, figures and videos of the individual attempts can be found on the online leaderboards[3]. The accompanying video of this paper shows the swing-ups in simulation and on the real hardware as well as the controllers robustness to disturbances induced by hitting the pendulum with a stick.

A. Acrobot

In the simulation phase HistorySAC and AR-EAPO showed the best performances (0.66 and 0.63) with a fast swing-up below 1 s, little and smooth torque usage and low velocities. They also showed good robustness results (0.75 and 0.73) especially concerning robustness to torque noise, torque responsiveness and delay. Like all other controllers, they were most sensitive to velocity noise. EvolSAC showed convincing results as well (perf. 0.52, robust. 0.69) only slightly behind HistorySAC and AR-EAPO. MC-PILCO was the least efficient controller of these four (perf. 0.31), with a higher energy usage an a less smooth torque signal. In the robustness score MC-PILCO was highly sensitive across most criteria except for torque responsiveness and perturbations (robust. 0.24).

In the hardware phase, the model-based algorithm MC-PILCO was trained based on recorded data from the real system which resulted in a policy which captured knowledge about effects which are not considered in the simple mathematical model. The policy proved to be very robust to the unknown external perturbations and MC-PILCO achieved 10/10 successful swing-ups (hardware avg. score 0.36). The swing-up trajectory of the highest scoring attempt is shown in Fig. 7b. AR-EAPO achieved even better scores in all five criteria but only managed 8/10 successful swing-ups resulting in a better best trial but a very close second place in the average score (avg. score 0.34). HistorySAC achieved only 1/10 swing-ups in the final evaluation (avg. score 0.03) and for EvolSAC the simulation-reality gap was too significant to swing-up the real acrobot system (avg. score 0.00).

[3]https://dfki-ric-underactuated-lab.github.io/real_ai_gym_leaderboard/

B. Pendubot

On the pendubot, all four controllers showed similar performances in swing-up time and energy usage. EvolSAC is the only one with a significantly higher torque cost and MC-PILCO the only one with a significantly higher torque smoothness value and velocity cost. This results in similar performance scores (HistorySAC: 0.68, AR-EAPO: 0.66, EvolSAC: 0.60, MC-PILCO: 0.48). In the robustness metrics, MC-PILCO again was sensitive to modeling errors and delay. AR-EAPO was quite robust to modeling errors and also very robust to random perturbations (100% success). This resulted in good robustness scores for all controllers (AR-EAPO: 0.91, EvolSAC: 0.79, HistorySAC: 0.77, MC-PILCO: 0.61) and AR-EAPO winning this category.

On the real hardware, MC-PILCO and AR-EAPO showed very convincing performances with 10/10 successful swing-ups and good scores in the individual criteria. Here, AR-EAPO took the win with a very slight margin (AR-EAPO: 0.65, MC-PILCO: 0.64). The best AR-EAPO trial is shown in Fig. 7c. HistorySAC also showed respectable results with 7/10 swing-ups (avg. score: 0.34). The EvolSAC team was not able to attend the conference and thus did not fine-tune the controller for the onsite setup, but EvolSAC was still successful 2/10 times (avg. score: 0.07).

TABLE III: Training specifications.

	Training	System Time	GPU	Control Freq.
MC-PILCO	$3-4$ h	$30-40$ s	RTX 4090	33 Hz
AR-EAPO	≈ 100 min	-	RTX 3080	500 Hz
EvolSAC	≈ 16 h*	-	Titan XP	500 Hz
HistorySAC	$3-4$ h	15 s	RTX 4070	500 Hz

*8h of training the on the GPU and 8h of evolutionary opt. on CPU

V. Discussion

In the simulation stage, the MC-PILCO controllers were trained with data only from the nominal plant used for performance scoring, without considering environmental variations of the robustness tests, thus reducing model learning reliability. However, MC-PILCO effectiveness highly depends on the model learning accuracy, hence, the poor robustness scores. In particular, the failure rate of the "model inaccuracies" benchmark is very high. On the real system, instead, the actual plant is fixed, thus model learning returns much accurate models, with benefits for policy effectiveness. The computational complexity of the stochastic Gaussian

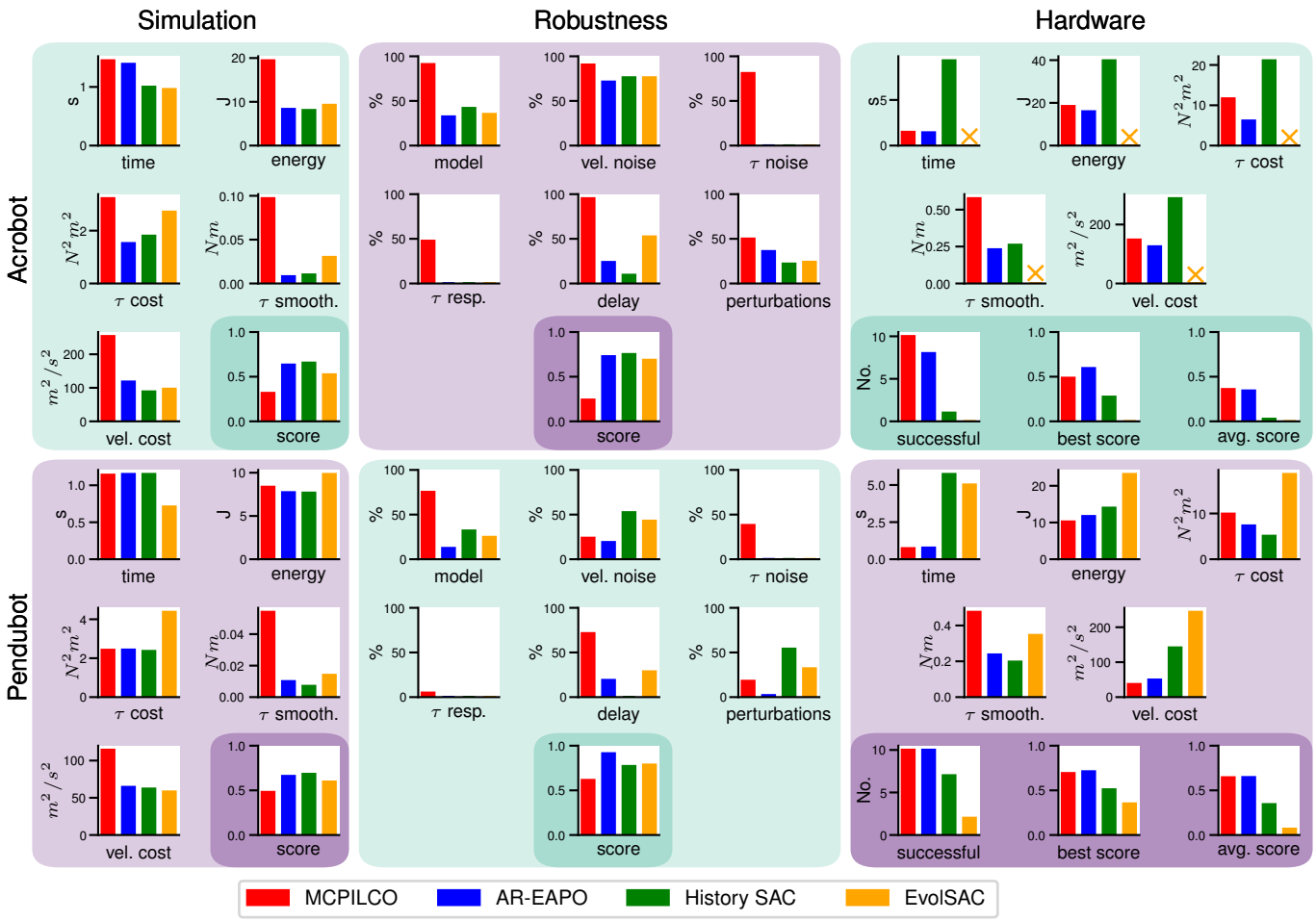
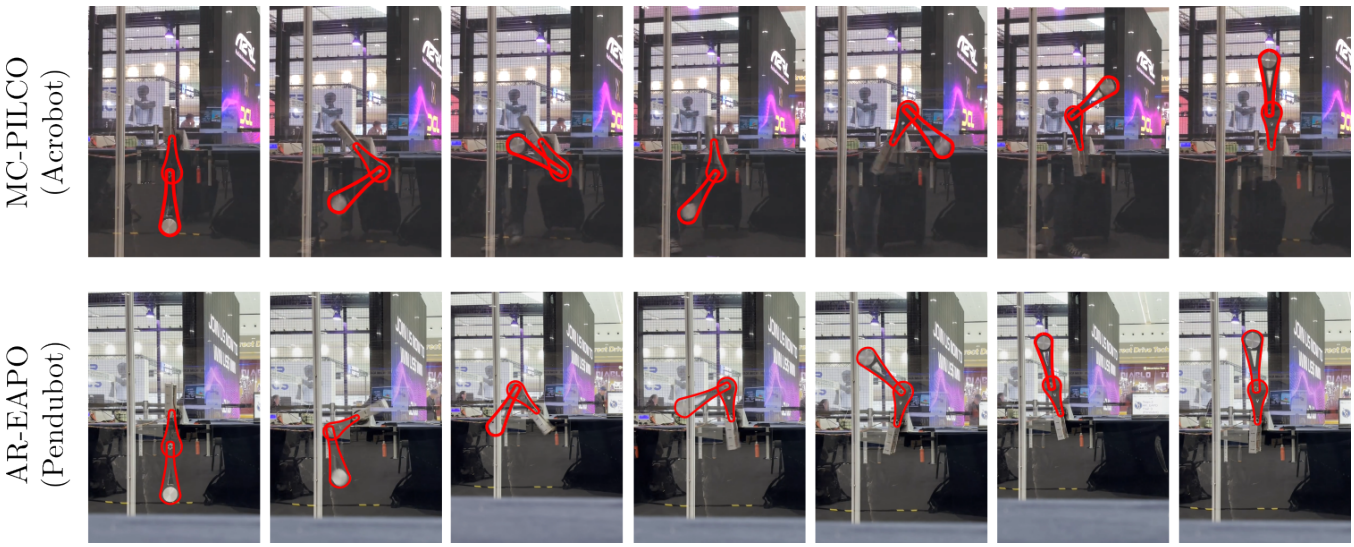


Fig. 6: All scores of the final round controllers. The simulation and robustness columns are computed from simulation data while the hardware score is evaluated from experiments on the real system. The robustness criteria are the percentages of failures in that category. For all criteria lower values are better and result in higher scores with 1.0 being the best achievable rating.

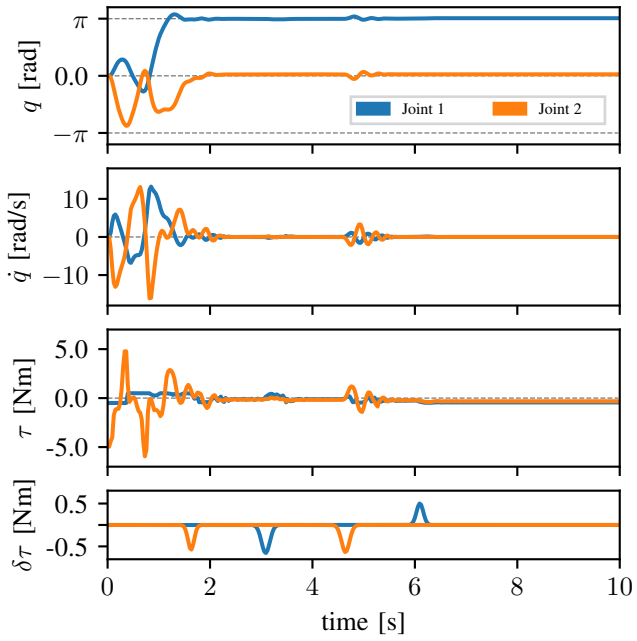
Process models is proportional to the cube of the control frequency, but 33 Hz is sufficient for the task in this competition. Compared to the model-free RL methods, MC-PILCO does not suffer from the sim-to-real problem since the policy is optimized w.r.t. a model learned from data collected on the actual system. The crucial hyperparameters concern the optimization problem in the policy update phase, in particular, performance is very sensitive to changes in the control frequency, the maximum control torque, and the horizon length. Most tuning was done by hand, while referring to similar applications, as the training process takes considerable time and requires system use.

AR-EAPO demonstrates outstanding robustness across both simulation and real systems, achieving effective zero-shot sim-to-real transfer without domain randomization or model learning. Tuning AR-EAPO requires empirically determining the temperature hyperparameter τ, which controls the trade-off between task performance and policy robustness. We attribute the policies' robustness to the algorithm's maximum entropy objective in the average-reward setting, which encourages exploration of diverse trajectories during training. This diversity appears to make the learned policy less sensitive to unmodeled dynamics and environmental variations. However, the relatively weaker performance on the Acrobot task suggests that the simple quadratic reward function might induce suboptimal trajectories during swing-ups. As an on-policy method, AR-EAPO exhibits the expected sample inefficiency typical of this class of algorithms, requiring approximately 20M simulator interactions for convergence. However, the algorithm maintains computational efficiency through vectorized simulations and its relatively simple algorithmic structure, making it practical for applications where simulation cost is manageable but real-world robustness is critical.

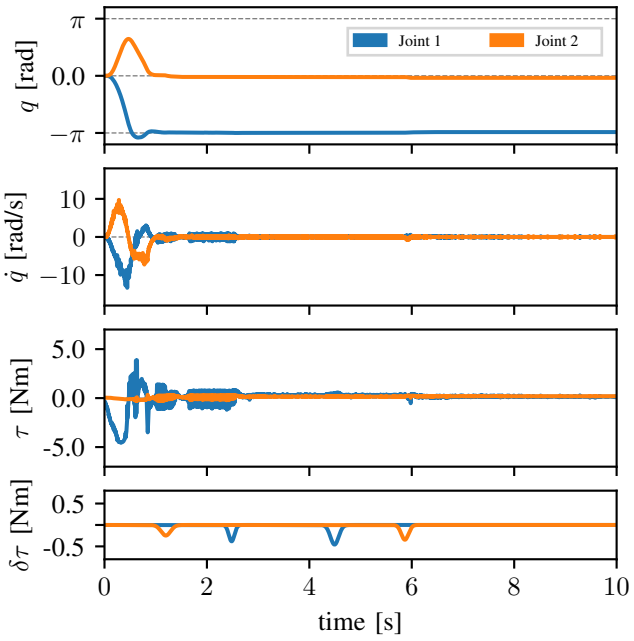
While model-free HistorySAC shows competitive performance in simulation, it falls behind model-based approaches on the real system. HistorySAC suffers vastly from the sim-to-real gap, which makes system identification a necessity for applying the learned policy on the real system. Temporal contextualization over multiple past velocity measurements improved the convergence speed and value when compared to learning from a single transition between two states.



(a) Snapshots of the experiments.



(b) MC-PILCO on the real acrobot system.



(c) AR-EAPO on the real pendubot system.

Fig. 7: Best swing-up trajectories on the real hardware for acrobot and pendubot. On top there are two rows with snapshots from the swing-up trajectories. The plots below show, from top to bottom, the evolution of the joint angles, velocities, motor torques and disturbances.

Network structure as well as past horizon length were determined empirically with focus on performance, efficiency and learning stability. Domain randomization during training and system identification afterwards improved the results.

The hybrid RL-Evolutionary approach EvolSAC showed competitive results in the simulation stage, with both performance and robustness scores aligned with the other model-free approaches. The choice of maximum allowed torque proved to be crucial, as high torque limits made the system unstable. Instead, lower torque limits provided more robust policies. The employed values were tuned with a small grid search. As shown in [17], evolutionary fine-tuning improved performance in both systems, while only minimally reducing robustness scores. Despite the simulated training incorporated a degree of domain randomization, the sim-to-real gap heavily impacted the results in the hardware phase, in particular the acrobot policy completely failed on the real system.

VI. Conclusion

The '2nd AI Olympics with RealAIGym' competition at the IROS 2024 conference gives interesting insights into the usability of state of the art RL algorithms on a real robotic system. The fact that all four teams which advanced to the

final round submitted RL controllers reflects that RL is a very active research field in robotics. The model-based RL method MC-PILCO achieved very good results with high robustness by learning very sample-efficiently from data recorded from the real system and won the acrobot competition track. The other method which stood out was AR-EAPO, with good scores on both systems and winning the pendubot category. Both methods exhibited a high robustness to unknown perturbations and even recovered most of the time after being pushed far away from their nominal path with a stick, proving to be promising candidates for finding a global swing-up and balance policy for the underactuated double pendulum systems. We made sure to make all results available online and hope that this competition inspires further research and thorough comparisons of control methods for dynamic behaviors on underactuated robots with high robustness.

ACKNOWLEDGMENT

The competition organizers at DFKI-RIC (FW, SV, DM, FK, and SK) acknowledge the support of the CoEx (Grant 01IW24008) and ActGPT (Grant 01IW25002) projects, funded by the German Federal Ministry of Research, Technology and Space (BMFTR) and are additionally sup- ported with funds from the federal state of Bremen (Grant 265/004-08-02-02-30365/2024-102966/2024-740847/2024). TV, BB, and JP acknowledge the grant "Einrichtung eines Labors des Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) an der Technischen Universität Darmstadt" of the Hessisches Ministerium für Wissenschaft und Kunst. This research was supported by Research Clusters "The Adaptive Mind" and "Third Wave of AI", funded by the Excellence Program of the Hessian Ministry of Higher Education, Science, Research and the Arts. Parts of the calculations for this research were conducted on the Lichtenberg high-performance computer of the TU Darmstadt. Alberto Dalla Libera and Giulio Giacomuzzo were supported by PNRR research activities of the consortium iNEST (Interconnected North-Est Innovation Ecosystem) funded by the European Union Next GenerationEU (Piano Nazionale di Ripresa e Resilienza (PNRR) – Missione 4 Componente 2, Investimento 1.5 – D.D. 1058 23/06/2022, ECS_00000043). This manuscript reflects only the Authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them. TV and BB also thank Daniel Palenicek and Tim Schneider for their support while performing the experiments on the IAS group's computing cluster at TU Darmstadt.

REFERENCES

[1] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html

[2] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[3] N. Funk, C. Schaff, R. Madan, T. Yoneda, J. U. De Jesus, J. Watson, E. K. Gordon, F. Widmaier, S. Bauer, S. S. Srinivasa, T. Bhattacharjee, M. R. Walter, and J. Peters, "Benchmarking structured policies and policy optimization for real-world dexterous object manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 478–485, 2022.

[4] F. Wiebe, S. Vyas, L. J. Maywald, S. Kumar, and F. Kirchner, "Realaigym: Education and research platform for studying athletic intelligence," in *Proceedings of Robotics Science and Systems Workshop Mind the Gap: Opportunities and Challenges in the Transition Between Research and Industry, New York*, 2022.

[5] F. Wiebe, J. Babel, S. Kumar, S. Vyas, D. Harnack, M. Boukheddimi, M. Popescu, and F. Kirchner, "Torque-limited simple pendulum: A toolkit for getting familiar with control algorithms in underactuated robotics," *Journal of Open Source Software*, vol. 7, no. 74, p. 3884, 2022. [Online]. Available: https://doi.org/10.21105/joss.03884

[6] M. Javadi, D. Harnack, P. Stocco, S. Kumar, S. Vyas, D. Pizzutilo, and F. Kirchner, "Acromonk: A minimalist underactuated brachiating robot," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3637–3644, 2023.

[7] F. Wiebe, S. Kumar, L. J. Shala, S. Vyas, M. Javadi, and F. Kirchner, "Open source dual-purpose acrobot and pendubot platform: Benchmarking control algorithms for underactuated robotics," *IEEE Robotics & Automation Magazine*, vol. 31, no. 2, pp. 113–124, 2024.

[8] F. Wiebe, N. Turcato, A. Dalla Libera, C. Zhang, T. Vincent, S. Vyas, G. Giacomuzzo, R. Carli, D. Romeres, A. Sathuluri, M. Zimmermann, B. Belousov, J. Peters, F. Kirchner, and S. Kumar, "Reinforcement learning for athletic intelligence: Lessons from the 1st "ai olympics with realaigym" competition," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, K. Larson, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2024, pp. 8833–8837, demo Track. [Online]. Available: https://doi.org/10.24963/ijcai.2024/1043

[9] C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín, and P. Stone, "Deep reinforcement learning for robotics: A survey of real-world successes," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 27, Apr. 2025, pp. 28 694–28 698. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/35095

[10] F. Amadio, A. Dalla Libera, R. Antonello, D. Nikovski, R. Carli, and D. Romeres, "Model-based policy search using monte carlo gradient estimation with real systems application," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3879–3898, 2022.

[11] N. Turcato, A. Dalla Libera, G. Giacomuzzo, R. Carli, *et al.*, "Teaching a robot to toss arbitrary objects with model-based reinforcement learning," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*. IEEE, 2023, pp. 1126–1131.

[12] J. S. C. Choe, B. Choi, and J.-k. Kim, "Simplifying reward design in complex robotics: Average-reward maximum entropy reinforcement learning," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*, 2025.

[13] J. S. B. Choe and J.-K. Kim, "Maximum entropy on-policy actor-critic via entropy advantage estimation," 2024. [Online]. Available: https://arxiv.org/abs/2407.18143

[14] V. Dewanto, G. Dunn, A. Eshragh, M. Gallagher, and F. Roosta, "Average-reward model-free reinforcement learning: a systematic review and literature mapping," *arXiv preprint arXiv:2010.08920*, 2020.

[15] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[17] M. Calì, A. Sinigaglia, N. Turcato, R. Carli, and G. A. Susto, "Ai olympics challenge with evolutionary soft actor critic," *arXiv preprint arXiv:2409.01104*, 2024.

[18] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[19] T. Schaul, T. Glasmachers, and J. Schmidhuber, "High dimensions and heavy tails for natural evolution strategies," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011, pp. 845–852.

[20] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of global optimization*, vol. 11, pp. 341–359, 1997.