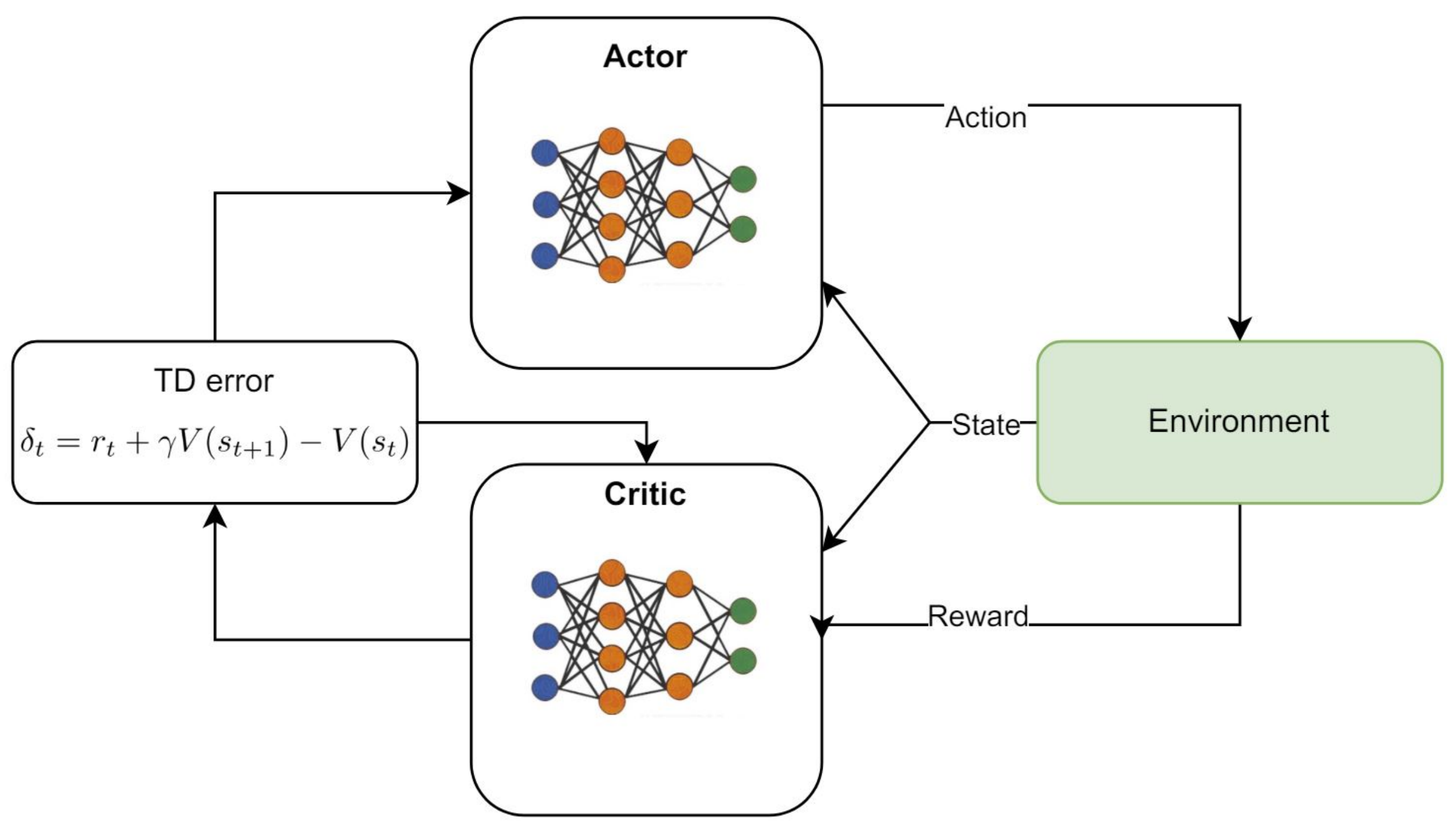
PPO vs. SAC: A Comparative Study on Learning Stability and Performance

Mischa Mez & Guillaume Spahr

Introduction

Deep Reinforcement Learning (Deep RL) represents a significant advancement in the field of artificial intelligence, integrating the principles of reinforcement learning (RL) with the powerful function approximation capabilities of deep neural networks.

Reinforcement Learning methods can be broadly categorized into three main types: value-based, policy-based, and actor-critic methods. Our goal is to detail two actor-critic algorithm, namely Proximal Policy optimization (PPO) and Soft Actor Critic (SAC).



Outline of PPO

While existing reinforcement learning methods like deep Q-learning, vanilla policy gradient methods, and TRPO have limitations in scalability, data efficiency, and robustness, PPO aims to combine the benefits of TRPO with simpler implementation and better sample complexity. PPO uses a novel objective function with clipped probability ratios to limit policy updates, ensuring they are not too drastic. It aims to balance sample efficiency, simplicity, and runtime performance.

Clipped Surrogate Objective

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, and ϵ is a hyperparameter (typically $\epsilon = 0.2$).

$\hat{A}_t$ is the generalized advantage estimator used by [2]

PPO-Clip objective

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t\left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t)\right]$$

L_t^{CLIP} is the previously defined surrogate loss

L_t^{VF} is the value function loss. It measures the difference between the predicted value function and the actual returns. It can be defined by a squared-error loss: $L_t^{VF}(\theta) = \left(V_\theta(s_t) - V_t^{\text{targ}}\right)^2$, where $V_\theta(s_t)$ is the predicted value of state s_t under the policy parameterized by θ, and V_t^{targ} is an estimate of the return using the Generalized Advantage Estimation (GAE) [2].

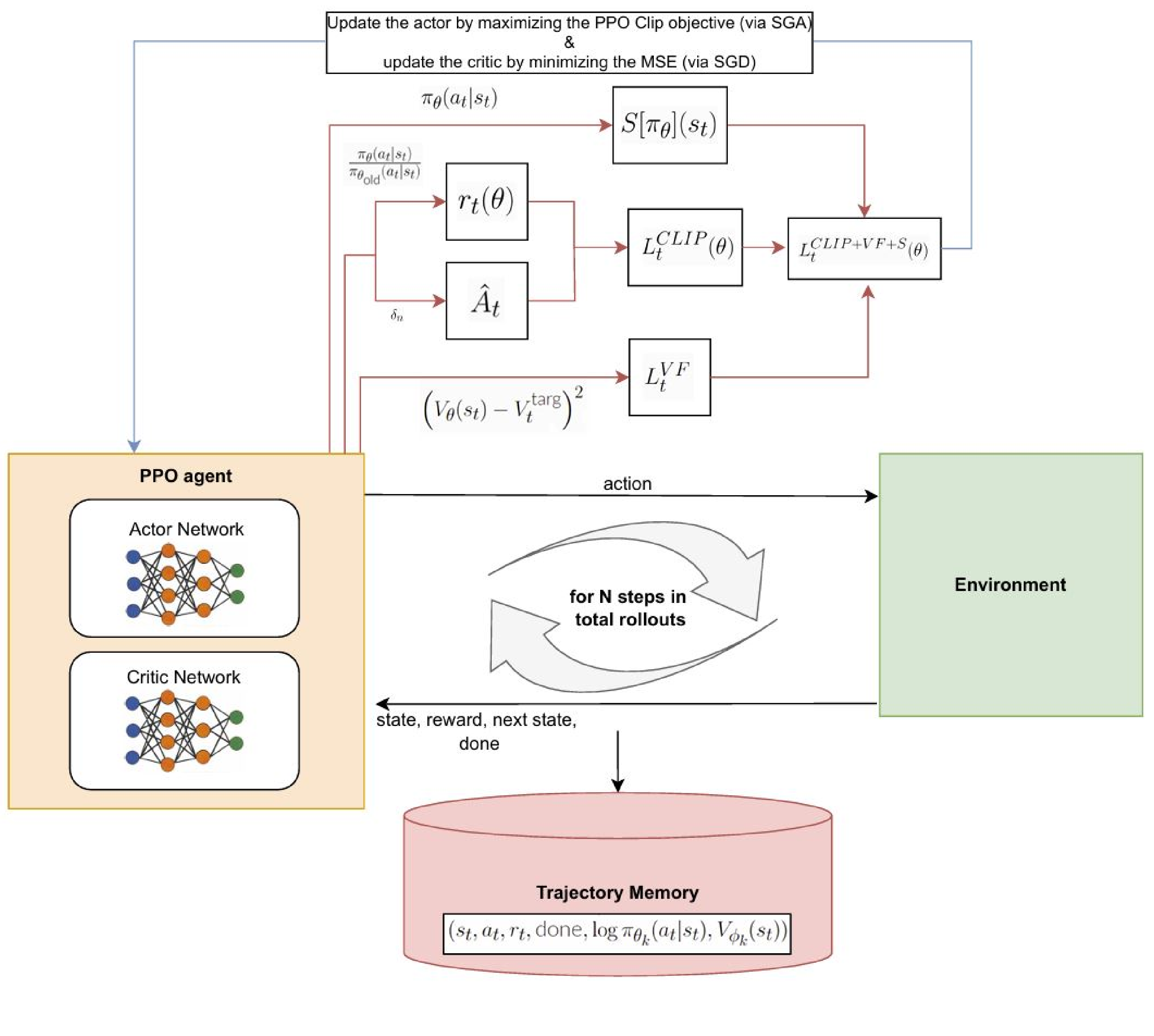

Update the actor by maximizing the PPO Clip objective (via SGA) & update the critic by minimizing the MSE (via SGD)
$\pi_\theta(a_t|s_t)$
$S[\pi_\theta](s_t)$
$\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
$r_t(\theta)$
$L_t^{CLIP}(\theta)$
$L_t^{CLIP+VF+S}(\theta)$
δ_t
$\hat{A}_t$
L_t^{VF}
$(V_\theta(s_t) - V_t^{\text{targ}})^2$
PPO agent
Actor Network
Critic Network
action
Environment
for N steps in total rollouts
state, reward, next state, done
Trajectory Memory
$(s_t, a_t, r_t, \text{done}, \log\pi_{\theta_k}(a_t|s_t), V_{\phi_k}(s_t))$

Figure 1. High-level diagram of the proximal policy optimization algorithm

Outline of SAC

The Soft Actor-Critic (SAC) algorithm is a model-free, off-policy actor-critic algorithm designed for continuous action spaces. In our case, we also design a SAC algorithm for discrete action spaces. We also decided to simplify the original algorithm by omitting the value-functions, including the target one, by replacing them by the use of a second pair of Q-functions expressing the new targets. We took this concept from [3].

The SAC (Soft Actor-Critic) algorithm is distinguished by its objective to maximize entropy, which maintains a degree of randomness in the policy while optimizing the reward. Another notable aspect of SAC is the soft update of the target networks, which helps prevent the actor from getting stuck in local minima.

SAC objective

The objective function to maximize is:

$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi}[r(s_t,a_t) + \alpha\mathcal{H}(\pi(a_t|s_t))]$

with the entropy term $\mathcal{H}(\pi(\cdot|s))] = \mathbb{E}_{a\sim\pi}[-log(\pi(a|s))]$

To achieve this optimization process, we define the critic loss to minimize:

$J_Q(\theta) = \mathbb{E}_{(s,a,r,s',a')\sim\mathcal{D}}[\frac{1}{2}(Q_\theta(s,a) - \hat{Q}(s,a))^2]$

with $\hat{Q}(s,a)) = r(s,a) + \gamma(\min_{j=1,2} Q_{\theta_{target,j}}(s',\tilde{a}') - \alpha log(\pi_\phi(\tilde{a}'|s'))$

We can then define the actor loss that must be maximized:

$J_\pi(\phi) = \mathbb{E}_{s\sim D,\epsilon_t\sim\mathcal{N}}[\min_{j=1,2} Q_{\phi_j}(s,\tilde{a}) - \alpha log(\pi_\phi(\tilde{a}|s)]$

Note: s, a are the current state, action and s', a' are the next state and action. The is the action sampled from the policy. In the continuous case, is obtained through the reparameterization trick.

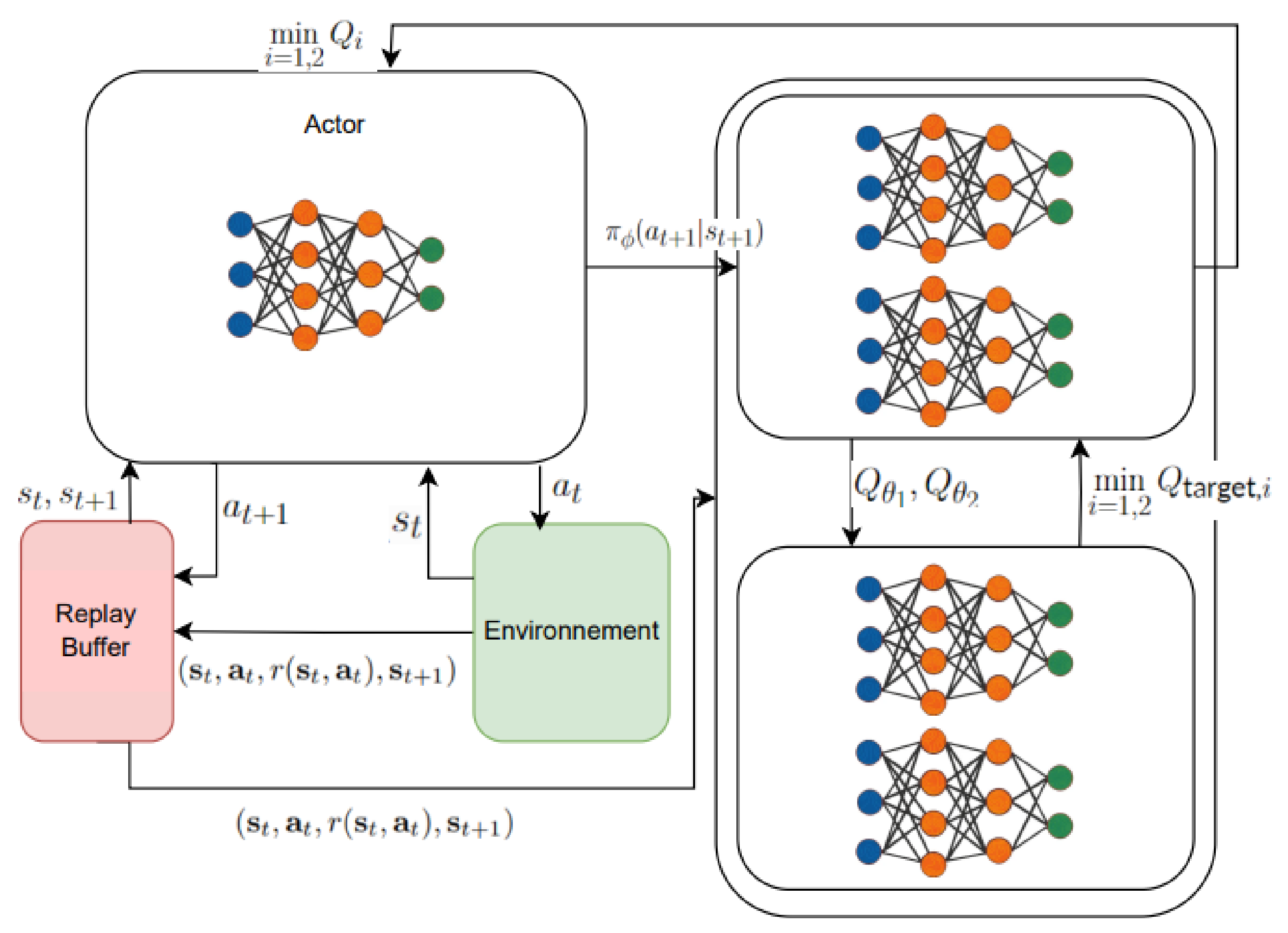

$\min_{i=1,2} Q_i$
Actor
$\pi_\phi(a_{t+1}|s_{t+1})$
s_t, s_{t+1}
a_{t+1}
s_t
a_t
$Q_{\theta_1}, Q_{\theta_2}$
$\min_{i=1,2} Q_{\text{target},i}$
Replay Buffer
Environnement
$(s_t, a_t, r(s_t,a_t), s_{t+1})$
$(s_t, a_t, r(s_t,a_t), s_{t+1})$

Figure 2. High-level diagram of the soft actor critic algorithm

Comparing Optimal Performances between SAC and PPO

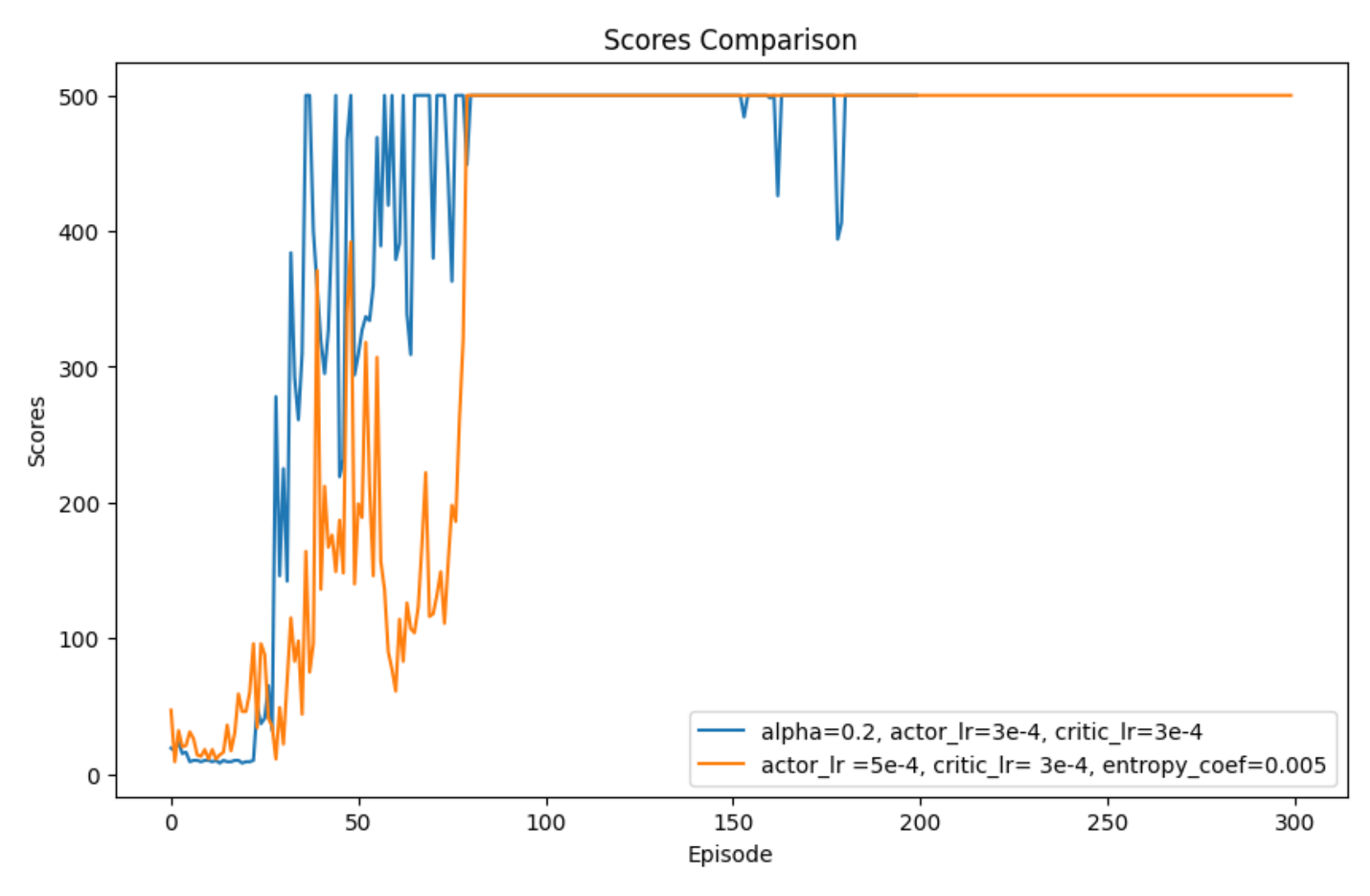


Figure 3. hyperparameters for Cartpole

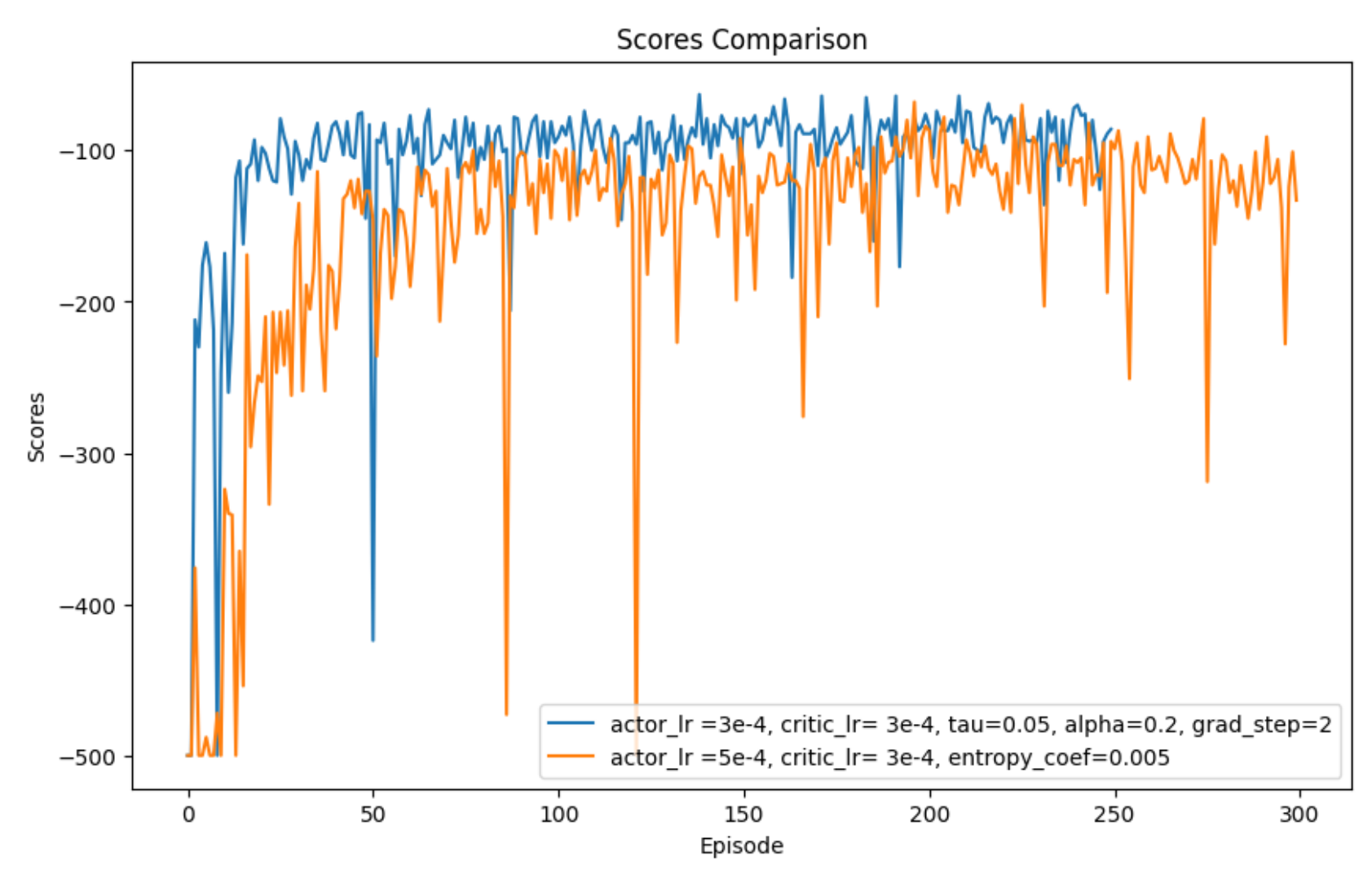


Figure 4. hyperparameters for Acrobot

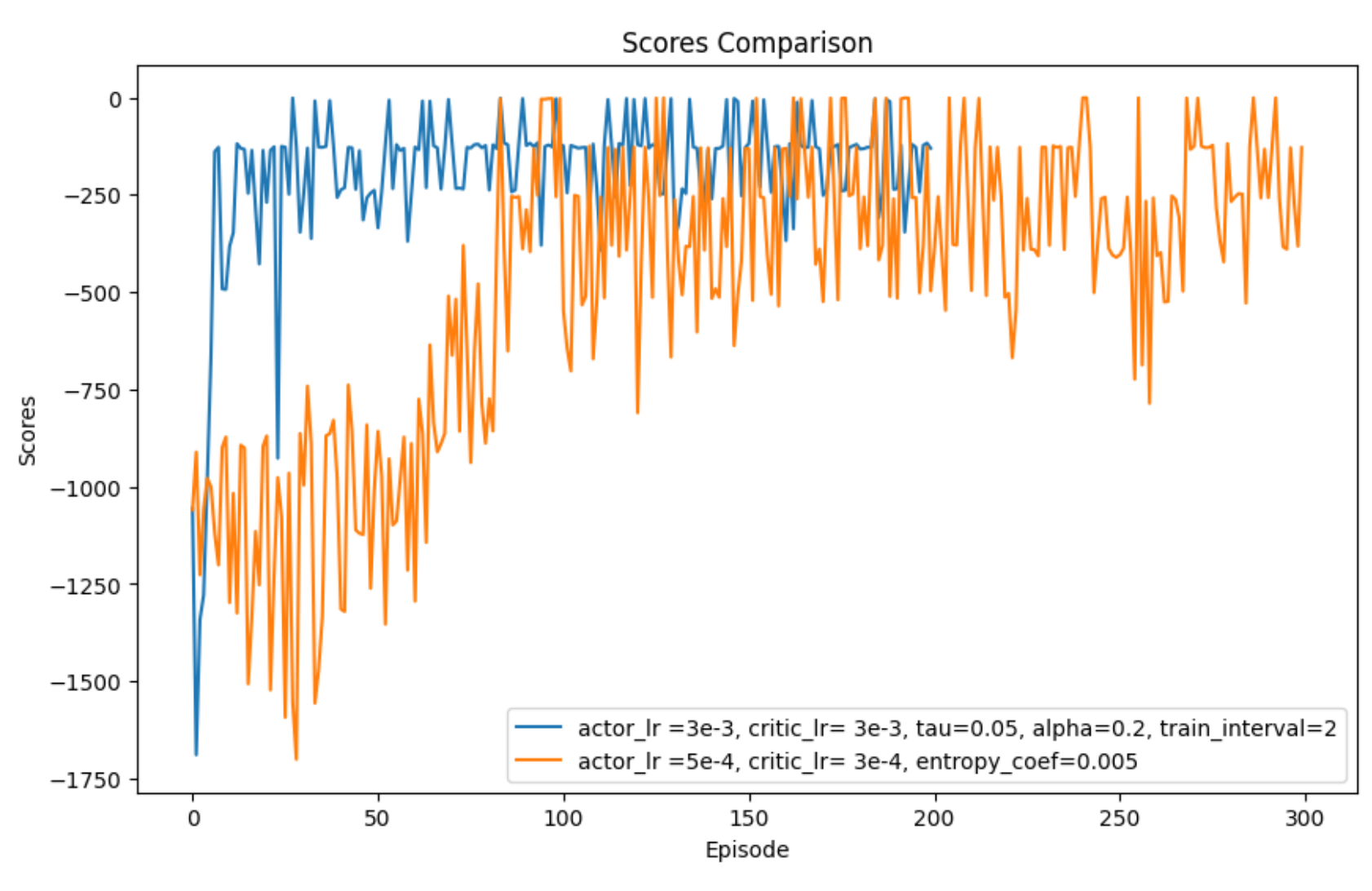


Figure 5. hyperparameters for Pendulum

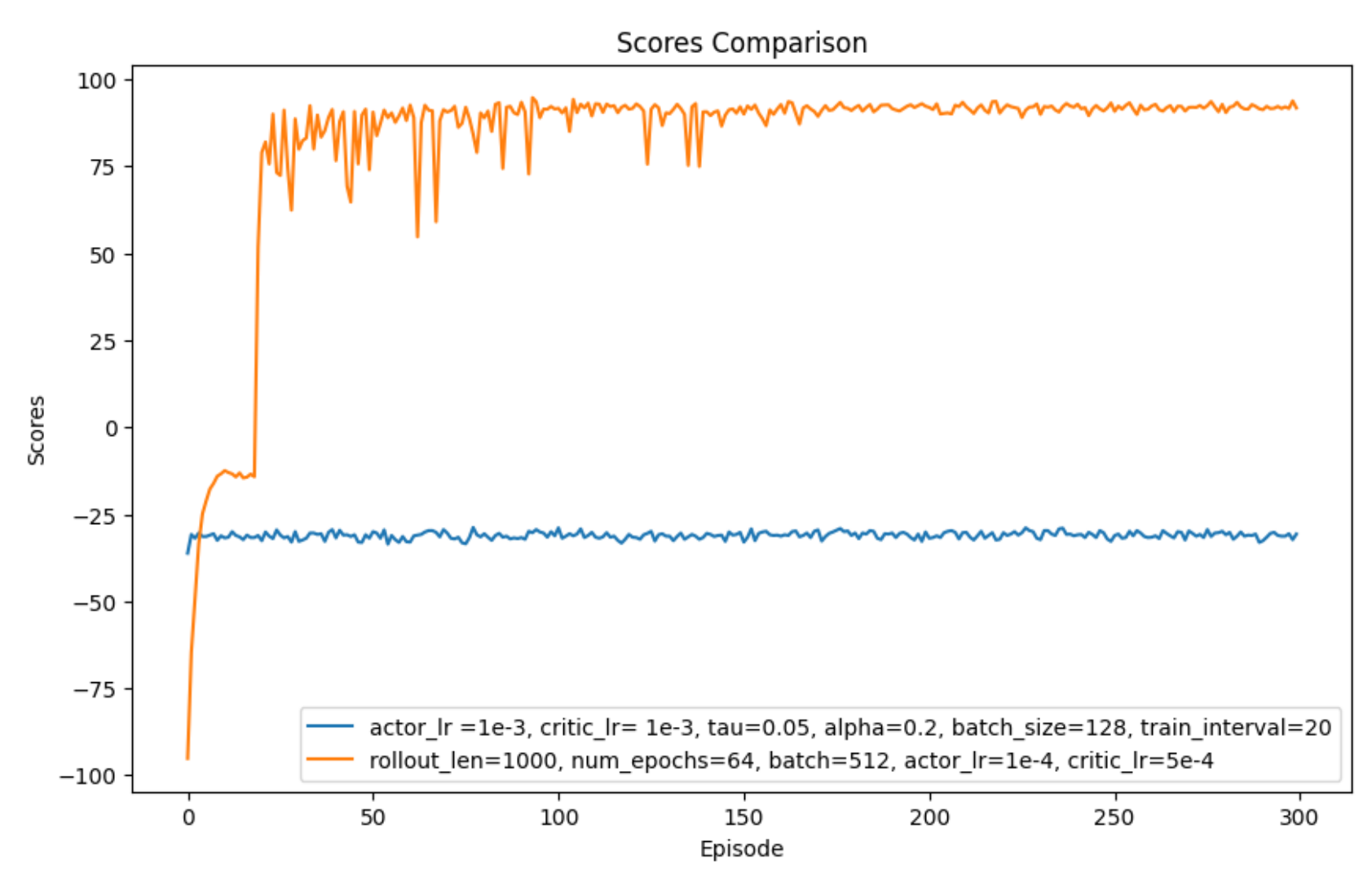


Figure 6. hyperparameters for MountainCarContinious

Feature	**SAC (Soft Actor-Critic)**	**PPO (Proximal Policy Optimization)**
Algorithm Type	Off-policy	On-policy
Stability	Highly stable due to entropy regularization, soft updates, twin Q	Generally stable, but can be sensitive to hyperparameters (entropy coeff and learning rates)
Sample Efficiency	High (uses replay buffer to reuse samples)	Moderate (does not reuse samples)
Exploration	Encourages exploration through entropy maximization	Balanced exploration-exploitation via clipping objective
Convergence Speed	higher convergence (frequent updates)	Lower (due to exploration/exploitation tradeoff)
Computation Cost	Higher computational cost due to off-policy learning and replay buffer management	Lower computational cost, simpler updates
Hyperparameter Sensitivity	Less sensitive due to entropy regularization	More sensitive, requires careful selection of buffer size and learning rates

Table 1: Comparison of SAC and PPO algorithms

References

[1] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[2] Volodymyr Mnih et al. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937.

[3] OpenAI. *Soft Actor-Critic*. `https://spinningup.openai.com/en/latest/algorithms/sac.html`. Accessed: 2024-05-20. 2018.

[4] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).