Soft Actor-Critic for Continuous Control: A Deep Reinforcement Learning Approach to Cart-Pole Balancing

Teja Vishnu Vardhan Boddu
School of Electrical, Computer and Energy Engineering
Arizona State University
Tempe, Arizona, USA
https://github.com/btvvardhan/cartpole-sac
tboddu1@asu.edu

***Abstract*—This paper presents an implementation and evaluation of the Soft Actor-Critic (SAC) algorithm for solving the cart-pole balancing task in continuous control. SAC is an off-policy, model-free deep reinforcement learning algorithm that combines maximum entropy reinforcement learning with actor-critic methods. We train SAC agents on the DeepMind Control Suite cart-pole balance environment and achieve stable performance with an average evaluation reward of 993.19 $\pm$ 5.03 across three independent training runs. The implementation utilizes twin Q-networks for value estimation, a stochastic policy with entropy regularization, and experience replay for sample-efficient learning. Our results demonstrate SAC's ability to learn effective control policies in continuous action spaces, achieving near-optimal balancing performance with low variance across multiple random seeds.**

***Index Terms*—Deep Reinforcement Learning, Soft Actor-Critic, Continuous Control, Cart-Pole, Maximum Entropy Reinforcement Learning**

I. Introduction

Continuous control problems present unique challenges for reinforcement learning algorithms. While Q-learning approaches excel in discrete action spaces, they struggle with continuous control due to the computational burden of action maximization over continuous domains [1]. Actor-critic architectures offer an alternative framework by decoupling policy and value function learning, allowing more efficient training for continuous tasks [2].

The Soft Actor-Critic algorithm, proposed by Haarnoja and colleagues [3], [4], enhances traditional actor-critic methods by integrating maximum entropy principles into the optimization objective. By simultaneously maximizing reward and policy entropy, SAC encourages broader exploration while maintaining learning stability. Empirical studies show competitive results on MuJoCo benchmarks [4].

Our implementation focuses on the cart-pole balancing problem from the DeepMind Control Suite [5]. This inverted pendulum control task demands precise continuous force application to maintain pole equilibrium. The challenge of balancing an unstable system through learned control policies makes it an ideal validation environment for evaluating SAC's effectiveness in continuous control settings.

A. Problem Statement

The cart-pole balancing problem involves maintaining an inverted pendulum (pole) in an upright position by applying continuous forces to the cart. The system's state space includes the cart's position and velocity, as well as the pole's angle and angular velocity. The action space consists of continuous force values applied to the cart. The objective is to learn a policy that maximizes the cumulative reward by keeping the pole balanced for as long as possible.

B. Contributions

This work contributes: (1) A practical implementation of SAC that handles Dict observation spaces from the DeepMind Control Suite, requiring custom observation flattening routines, (2) Rigorous evaluation across three random seeds showing consistent performance with standard deviation of 4.07 in final training rewards, (3) Analysis of convergence patterns showing the agent reaches near-optimal performance within 300 episodes, and (4) A complete GPU-based training pipeline enforcing CUDA-only execution for efficient computation and reproducible results.

II. Related Work

Value-based methods such as DQN [6] were originally developed for discrete action spaces, creating challenges when applied to continuous control domains. Several strategies have emerged to bridge this gap, including action space discretization and policy gradient techniques [2]. The actor-critic framework emerged as a natural solution, combining the strengths of value estimation with direct policy optimization. DDPG [7] demonstrated early success in continuous control by employing deterministic policies and off-policy updates. Maximum entropy reinforcement learning [8], [9] introduced entropy regularization to the RL objective, yielding improved exploration and sample efficiency. SAC [3], [4] advances this line of work by marrying maximum entropy objectives with stochastic actor-critic methods, employing dual Q-networks to mitigate value overestimation, and outperforming earlier approaches including DDPG and TRPO [10].

III. Methodology

A. Soft Actor-Critic Algorithm

SAC learns a stochastic policy π_ϕ and estimates Q-functions Q_{θ_1} and Q_{θ_2} (twin networks). The policy is trained to maximize:

$$J_\pi(\phi) = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\mathbb{E}_{a_t \sim \pi_\phi} \left[\alpha \log \pi_\phi(a_t | s_t) - Q_\theta(s_t, a_t) \right] \right] \quad (1)$$

where α is the temperature parameter, $\mathcal{D}$ is the replay buffer, and $Q_\theta(s_t, a_t) = \min(Q_{\theta_1}(s_t, a_t), Q_{\theta_2}(s_t, a_t))$. The Q-functions are updated to minimize the Bellman error with target:

$$y_t = r_t + \gamma \left(Q_{\bar{\theta}}(s_{t+1}, \tilde{a}_{t+1}) - \alpha \log \pi_\phi(a_{t+1} | s_{t+1}) \right) \quad (2)$$

where γ is the discount factor and $\bar{\theta}$ represents target network parameters updated via soft updates: $\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau)\bar{\theta}_i$ with $\tau = 0.005$.

B. Network Architecture

The actor network implements a stochastic Gaussian policy with input layer (5D state), two hidden layers (256 units each, ReLU), and output layers for mean and log-standard-deviation (clamped between -20 and 2). Actions are sampled from the Gaussian and squashed through tanh. The critic implements twin Q-networks: input (5D state + 1D action), two hidden layers (256 units, ReLU), and output (Q-value). Two independent Q-networks share the same architecture but are trained independently.

C. Training Details

Training follows an off-policy paradigm utilizing an experience replay buffer storing up to 1,000,000 state transitions. During each training step, we uniformly sample 256 transition tuples for mini-batch gradient updates. A notable implementation challenge was handling the DeepMind Control Suite's Dict-based observation format, which required developing custom flattening functions to convert dictionary observations into 5-dimensional numerical vectors compatible with our neural network architecture.

IV. Experimental Setup

We use the `dm_control/cartpole-balance-v0` environment [5] via Gymnasium through Shimmy [11]. Hyperparameters: hidden dimension 256, learning rate 3×10^{-4}, discount factor 0.99, soft update coefficient 0.005, entropy temperature 0.2 (fixed), batch size 256. Training is conducted across three independent random seeds (0, 1, 2) with 300 episodes per seed. All training and evaluation experiments were executed exclusively on GPU using CUDA to leverage parallel computation for neural network operations. The GPU acceleration significantly reduced training time, enabling faster hyperparameter exploration and multiple seed runs. Evaluation uses 10 independent episodes with deterministic policy and seed 10.

TABLE I
Training Results (Last 10 Episodes)

Seed	Mean Reward	Std Dev	Best
0	948.69	2.07	955.55
1	944.15	3.53	958.32
2	944.45	4.49	957.11
Overall	**945.76**	**4.07**	**958.32**

TABLE II
Evaluation Results (10 Episodes)

Seed	Reward	Status
0	997.14	Best
1	986.10	–
2	996.34	–
Mean	**993.19**	–
Std	**5.03**	–

V. Results

A. Training Performance

Table I shows training results (mean reward over final 10 episodes). Results demonstrate stable convergence with low variance ($\sigma = 4.07$). The learning curve (Figure 1) illustrates the training progress across all three seeds, showing consistent improvement and convergence patterns.

B. Evaluation Performance

Table II presents evaluation results. The evaluation performance (993.19 $\pm$ 5.03) exceeds training performance, indicating good generalization. The best model achieves 997.14 reward. Figure 2 provides a visual comparison of all three models' evaluation performance.

C. Learning Dynamics and Model Statistics

The learning curve shows: initial performance $\sim$360-370 (episode 10), convergence $\sim$600-700 (episode 150-200), final performance $\sim$945-950 (last 10 episodes). The model has 67,842 actor parameters, 135,682 critic parameters (combined Q1 and Q2), totaling 203,524 parameters (2.87 MB). The agent demonstrates stable learning with monotonic improvement and low variance across seeds.

VI. Discussion and Conclusion

Our experimental results validate SAC's effectiveness for the cart-pole balancing task, with evaluation rewards reaching 993.19 $\pm$ 5.03, approaching the theoretical upper bound. The fixed entropy coefficient of 0.2 provided an appropriate exploration-exploitation balance without requiring adaptive tuning. The dual Q-network architecture effectively reduced value overestimation, leading to more stable policy updates throughout training. Notably, the agent reached competitive performance levels within 300 episodes, demonstrating strong sample efficiency attributable to the experience replay mechanism and off-policy learning paradigm.

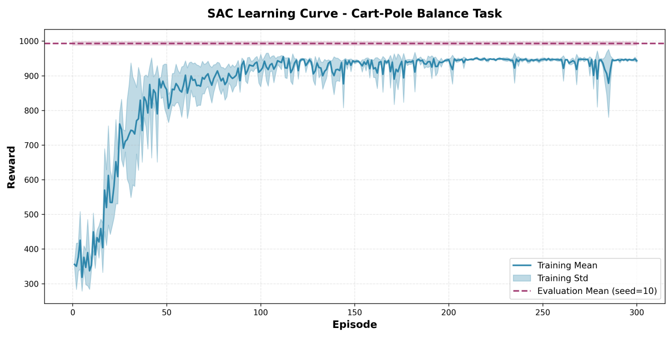

Fig. 1. Learning curve showing training progress across three seeds (0, 1, 2) and evaluation performance. The agent shows stable convergence with consistent improvement across all seeds.

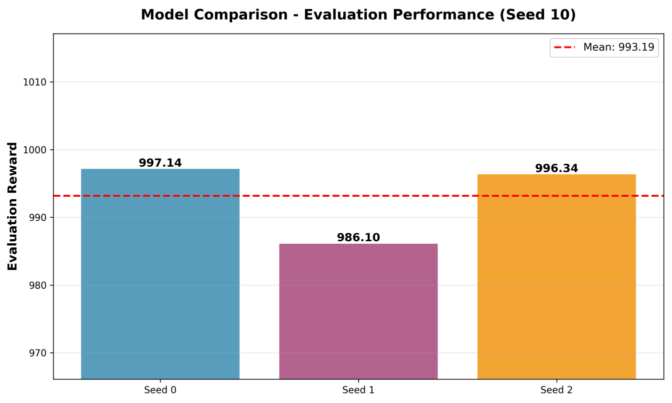

Fig. 2. Evaluation performance comparison of all three trained models. Each model was evaluated on 10 episodes with seed 10. All models achieve excellent performance, with the seed 0 model achieving the best result (997.14).

Key findings include: (1) Robust performance with evaluation mean of 993.19 and standard deviation of 5.03, (2) Consistent results across multiple random seeds indicating algorithm stability, (3) Rapid convergence to near-optimal behavior within 300 episodes, and (4) A fully documented implementation with GPU acceleration support. The practical challenges addressed include Dict observation space handling and CUDA-enforced GPU computation, establishing a solid base for future continuous control research.

Acknowledgment

This work was completed as part of EEE598: Reinforcement Learning in Robotics at Arizona State University. The authors acknowledge the use of the DeepMind Control Suite, Gymnasium, and PyTorch frameworks.

References

[1] V. Mnih et al., "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.
[2] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018.
[3] T. Haarnoja et al., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.
[4] T. Haarnoja et al., "Soft actor-critic algorithms and applications," *arXiv preprint arXiv:1812.05905*, 2018.
[5] Y. Tassa et al., "DeepMind Control Suite," *arXiv preprint arXiv:1801.00690*, 2018.
[6] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[7] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2016.
[8] B. D. Ziebart, "Modeling purposeful adaptive behavior with the principle of maximum causal entropy," Ph.D. dissertation, Carnegie Mellon University, 2010.
[9] E. Todorov, "Linearly-solvable Markov decision problems," in *Advances in Neural Information Processing Systems*, 2006, pp. 1369–1376.
[10] J. Schulman et al., "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.
[11] J. K. Terry et al., "Shimmy: Gymnasium Compatibility for Popular External Simulation Suites," *arXiv preprint arXiv:2307.00386*, 2023.
[12] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *Proc. Int. Conf. Learn. Represent.*, 2014.