A comparison of PPO and SAC agents in the Cartpole Environment

Pranav Srivastava [1 2]
Yousef Swed [1 2]
Kevin Bretz [1 2]

Abstract

This paper presents an implementation and analysis of two state-of-the-art reinforcement learning algorithms: Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) in the CartPole environment. Building upon previous work with REINFORCE and Actor-Critic methods, this report examines how these advanced algorithms improve learning stability and performance through trust region optimization and entropy maximization respectively. The results demonstrate that PPO achieves superior performance in environments with simple, discrete action spaces, such as CartPole, with more stable learning and consistent convergence compared to both SAC and previous methods.

1. Introduction

Recent advances in reinforcement learning have led to increasingly sophisticated actor-critic architectures that address the limitations of basic policy gradient methods. While our previous work examined REINFORCE and basic Actor-Critic approaches, this paper investigates two state-of-the-art algorithms: PPO and SAC. These methods represent different approaches to improving actor-critic learning - PPO through constrained policy updates and SAC through entropy maximization and off-policy learning. This study implements both algorithms in the CartPole environment, providing insights into their relative strengths in discrete action spaces.

2. Theory

2.1. Basic Policy Loss Review

Before examining PPO and SAC, we recall the basic policy gradient loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{s\sim D}\left[\mathbb{E}_{a\sim\pi_\theta(a|s)}\left[-Q^{\pi_{\theta_k}}(s,a)\right]\right]$$

This forms the foundation that both PPO and SAC build upon, each addressing its limitations in different ways.

2.2. PPO Algorithm

PPO improves upon basic policy gradient methods by introducing a trust region constraint through clipping. The core objective is:

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

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$ is the probability ratio between new and old policies, and ϵ is the clipping parameter. This clipping mechanism prevents large policy changes, promoting stable learning.

2.3. SAC Algorithm

SAC takes a different approach by incorporating entropy maximization into the objective:

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

As implemented in our code, SAC uses several key components:

- Twin Q-networks to prevent overestimation bias
- Automatic entropy tuning through temperature parameter α
- Experience replay for off-policy learning
- Soft target updates using Polyak averaging

2.4. Relation to Basic Actor-Critic

Both algorithms build upon the basic actor-critic framework while addressing its limitations:

PPO vs A2C: While A2C directly updates using policy gradients, PPO constrains updates through clipping:

$$\begin{aligned} L^{VF}(\theta) = \\ \mathbb{E}_t[\min((V_\theta(s_t) - R_t)^2, \\ (clip(V_\theta(s_t) - V_{\theta_{old}}(s_t), -\epsilon, \epsilon) \\ + V_{\theta_{old}}(s_t) - R_t)^2)] \end{aligned}$$

SAC vs A2C: SAC modifies the basic actor-critic framework by:

- Adding entropy maximization for exploration
- Using twin Q-networks instead of a single value network
- Implementing off-policy learning through experience replay

3. Implementation Details

3.1. Network Architectures

Both algorithms were implemented using neural networks with the same architecture so that the results are directly comparable. The only difference in design is that for our PPO agent we shared a single network between both actor and critic heads. For SAC, we assigned completely independent networks for each actor / critic.

3.1.1. PPO Network Architecture

The PPO implementation uses a single network with shared feature extraction: **Actor-Critic Network:**

- Shared Features: Linear(*state_dim*, 128)
- Policy Head: Linear(128, *action_dim*)
- Value Head: Linear(128,1)

3.1.2. SAC Network Architecture

Our SAC implementation uses separate networks as shown in our code:

- Policy Network: Linear(*state_dim*, 128) + ReLU + Linear(128, *action_size*)
- Q-Networks: Linear(*state_dim*, 128) + ReLU + Linear(128, *action_size*)

3.2. Engineering Improvements

3.2.1. PPO Enhancements

Our PPO implementation includes several crucial stability improvements:

1. **Generalized Advantage Estimation (GAE):**

$$A_t^{GAE(\gamma,\lambda)} = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ following (8). The generalized advantage estimation improves on the basic advantage by introducing a hyperparameter λ/τ that dictates how far into the future we should look, effectively balancing bias and variance. A λ/τ of 0 results in a 1-step TD(0) - high bias, low variance, whereas a λ/τ of 1 leads to a Monte Carlo style evaluation - low bias, high variance.

2. **Value Function Clipping:**

$$\begin{aligned} L^{VF}(\theta) = \min((V_\theta(s_t) - R_t)^2, \\ (clip(V_\theta(s_t) - V_{\theta_{old}}(s_t), -\epsilon, \epsilon) \\ + V_{\theta_{old}}(s_t) - R_t)^2) \end{aligned}$$

following (9). This is the core concept of the PPO algorithm. Policy updates are clipped by the parameter ϵ to prevent making too large changes per update.

3. **Mini-batch Processing:** Implementation of mini-batch updates (size 64) from collected trajectories following (1). By collecting and storing trajectory information in mini batches, then training on these batches for multiple epochs, we drastically improve our sample efficiency. This lessens the overall noise and increases stability, smoothing the training by reducing variance in the policy updates. Through mini batch processing, we can achieve faster convergence with fewer total steps.

4. **Decaying Entropy Coefficient:** To balance exploration and exploitation, it is common to add a fixed entropy coefficient to the objective function, such as done by (6). We take this one step further by implementing a decaying entropy coefficient, which attempts to mimic the effect of epsilon decay:

$$\beta_{t+1} = \max(\beta_t \cdot \text{decay}, \beta_{min})$$

where β_t is the entropy coefficient at time t, initialized at 0.01, with a decay rate of 0.995 and minimum value of 0.001. This gradually reduces exploration as the policy improves, similar to epsilon decay in DQN. The entropy term is added to the PPO objective:

$$L^{total}(\theta) = L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + \beta_t \mathcal{H}[\pi_\theta]$$

This implementation allows for strong initial exploration while ensuring convergence to a more deterministic policy in later training stages.

3.2.2. SAC Enhancements

Our SAC implementation incorporates several key components for stability:

1. **Twin Q-Networks:** To prevent overestimation bias, we maintain two Q-networks and use their minimum for value estimation:

$$Q_{target}(s,a) = \min(Q_1(s,a), Q_2(s,a))$$

following (3).

2. **Experience Replay:** Implementation of a replay buffer with capacity 1,000,000 transitions for off-policy learning following (5). This is the same technique we previously implemented in our DQN agent from assignment 1.

3. **Automatic Entropy Tuning:** The temperature parameter α is automatically adjusted using:

$$J(\alpha) = \mathbb{E}_{a_t \sim \pi_t}[-\alpha \log \pi_t(a_t|s_t) - \alpha \mathcal{H}_{target}]$$

following (4). In previous iterations, α was given a fixed value, but it is beneficial to have it tuned as training progresses to allow for more dynamic exploration.

4. **Soft Target Updates:** Q-network targets are updated using Polyak averaging:

$$\theta_{target} \leftarrow \tau\theta + (1-\tau)\theta_{target}$$

following (4) and (7). This introduces the hyperparameter τ, which controls how gradual the target network updates are. For a value of 0, no updating happens, and for a value of 1 we get hard updates, as found in the DQN algorithm. We use a value of 0.005 to allow for slow, smooth updates.

4. Experiments

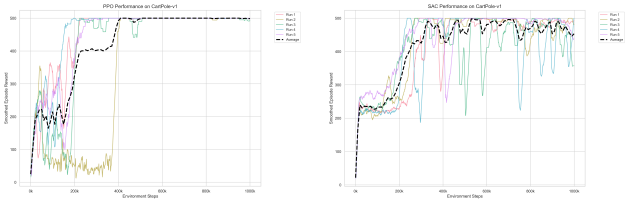
4.1. Experimental Setup

Both algorithms (PPO and SAC) were implemented using similar neural network architectures where possible. The experiments were conducted in the CartPole-v1 environment from Gymnasium. Each algorithm was run 5 times with 1 million environment steps per run. Further key details include:

- **Environment:** Gymnasium CartPole-v1
- **Common Hyperparameters:**
  - Hidden Size: 128
  - Learning Rate: 0.001
  - Discount Factor (gamma): 0.99
  - Total Steps: 1,000,000
  - Number of runs: 5
  - Seed: 42
- **PPO-specific parameters:**
  - Clipping parameter (ϵ): 0.2
  - GAE parameter (λ): 0.95
  - Value function coefficient: 0.5
  - Initial entropy coefficient: 0.01
  - Entropy decay: 0.995
  - Minimum entropy coefficient: 0.001
  - Batch size (on-policy): 64
  - Number of epochs: 10
- **SAC-specific parameters:**
  - Buffer size: 1,000,000
  - Batch size (off-policy): 128
  - Updates per step: 1
  - Target entropy: -log(action_size)
  - Temperature parameter (τ): 0.005

5. Results

We obtain the results by comparing PPO and SAC on the CartPole-v1 environment, with reference to our previous implementations of DQN and Actor-Critic methods. Performance graphs are provided in terms of smoothed average episode reward per environment step, with both algorithms run for 1 million steps a total of 5 times each.



((a)) Average performance for PPO

((b)) Average performance for SAC

Figure 1. Average performance comparison

As shown in Figure 1, PPO demonstrates superior stability and learning speed.

PPO Performance: Proximal Policy Optimization (PPO) demonstrated strong and stable performance throughout training. The agent learned rapidly, achieving near-optimal performance in under 400,000 steps. Across multiple runs, the learning curve remained highly stable with minimal variance, and the agent consistently maintained the optimal reward of 500. This smooth convergence pattern reflects the effectiveness of PPO's trust region-based optimization, which appears to stabilize learning.

SAC Performance: Soft Actor-Critic (SAC) showed even faster initial learning than PPO, which suggests effective early sampling. However, unlike PPO, SAC failed to converge to the optimal reward, and performance varied more significantly across runs. The entropy maximization component, which encourages continued exploration, likely increased this high variance. While it helps prevent early convergence, in this case, it may have hindered the agent from converging to an optimal policy, preventing it from consistently reaching the 500 reward threshold.

When compared to earlier implementations, both PPO and SAC exhibited greater stability than basic Actor-Critic methods.

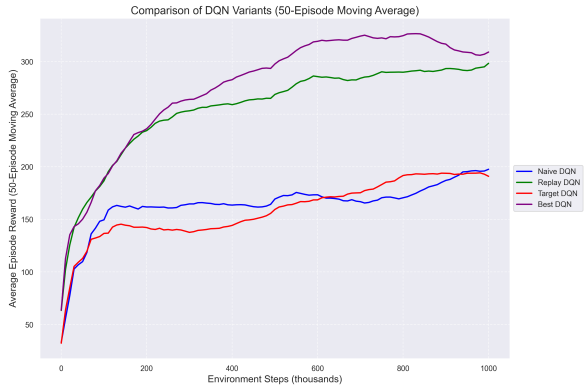


Figure 2. DQN Variants

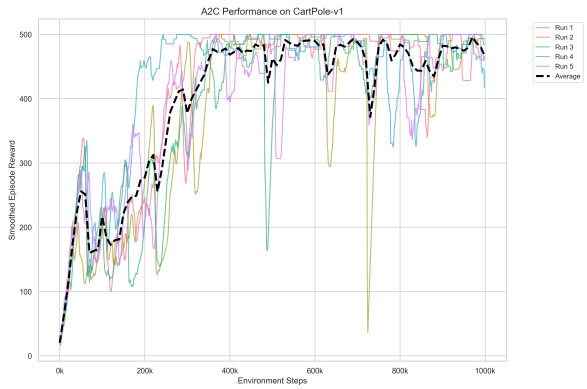


Figure 3. A2C Performance

Both PPO and SAC demonstrated greater stability compared to our earlier implementations of basic Actor-Critic methods as seen in Figure 3. In addition to being more stable, they also achieved faster convergence than our previous Deep Q-Network (DQN) implementation, as seen in Figure 2. Particularly, PPO benefited from its objective clipping heuristic, which helped maintain stable learning by preventing issues such as exploding gradients. On the other hand, SAC's off-policy learning approach, combined with experience replay, provided sample efficiency that was comparable to what we observed with A2C.

These results suggest that PPO's trust region optimization is particularly well-suited for discrete action spaces like CartPole, while SAC's entropy maximization and off-policy learning may be more beneficial in continuous action domains.

6. Discussion

The variations in performance observed between PPO and SAC can be attributed to their distinct approaches to policy optimization and their suitability for discrete action spaces.

PPO's Advantages: The strong performance of PPO in our CartPole implementation can be attributed to several key factors. The clipping mechanism plays a critical role in preventing large and potentially harmful policy updates, thereby promoting stable learning throughout training. Additionally, the use of Generalized Advantage Estimation (GAE) creates a good balance between bias and variance in the advantage calculation, enhancing learning stability. The simplicity of the CartPole environment, especially its discrete action space, aligns well with PPO's straightforward policy update mechanism, allowing it to perform efficiently.

SAC's Characteristics: Although SAC ultimately achieves good performance, its more sophisticated architecture presents certain limitations when applied to the CartPole environment. The entropy maximization component, which generally encourages beneficial exploration, may actually lead to excessive exploration in such a simple task, potentially hindering convergence. Furthermore, SAC's use of twin Q-networks adds architectural complexity that is typically more advantageous in continuous action spaces. In this discrete setting, the benefits of off-policy learning and experience replay—central to SAC—appear less pronounced compared to the on-policy nature of PPO.

Engineering Impact: Several engineering decisions played an important role in shaping the learning dynamics of both algorithms. For PPO, the use of mini-batch updates combined with GAE was essential in ensuring stable and efficient learning. In the case of SAC, automatic entropy tuning helped with balancing exploration and exploitation, although it required proper parameter initialization to work effectively. Additionally, adapting SAC to handle discrete actions, as proposed by (2), introduced further implementation complexity without delivering clear advantages in this particular environment.

7. Conclusion

Our implementation and analysis of PPO and SAC in the CartPole environment reveals that while both algorithms achieve strong performance, their relative strengths differ significantly. PPO demonstrates superior stability and learning efficiency, particularly well-suited for discrete action spaces. This is evidenced by its rapid convergence and consistent maintenance of optimal performance. SAC, while also effective, shows more variable learning characteristics that may be better suited to continuous action domains.

The comparison with our previous implementations of DQN and Actor-Critic methods shows clear improvements in both stability and final performance, particularly for PPO. However, SAC's more complex architecture, while theoretically more powerful, may be over-sophisticated for simple discrete environments like CartPole.

These findings suggest several directions for future work. One improvement would be to explore the performance of both PPO and SAC in environments that feature larger discrete action spaces, where the advantages and limitations observed here may differ. Another improvement could be refining SAC's entropy maximization strategy to better suit discrete environments, potentially improving its exploration efficiency without hindering convergence. Lastly, developing hybrid approaches that integrate PPO's stability with the sample efficiency of SAC's off-policy learning could offer the best of both worlds, leading to more robust and scalable reinforcement learning algorithms.

The results underscore the importance of matching algorithm complexity to environment requirements, with simpler, more focused approaches like PPO potentially outperforming more complex methods in straightforward tasks.

References

[1] Marcin Andrychowicz, Anton Raichuk, Piotr Stańczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, Léonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, et al. What matters in on-policy reinforcement learning? A large-scale empirical study. *arXiv preprint arXiv:2006.05990*, 2020.

[2] Petros Christodoulou. Soft actor-critic for discrete action settings. *arXiv preprint arXiv:1910.07207*, 2019.

[3] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International conference on machine learning*, pages 1861–1870, 2018.

[4] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[5] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

[6] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *International conference on machine learning*, pages 1928–1937, 2016.

[7] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[8] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.