Learning to Walk in BipedalWalker

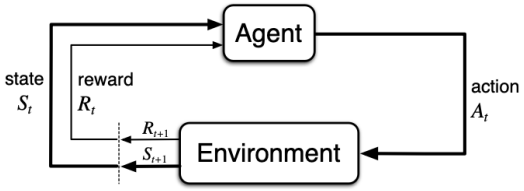
Anonymous author

Abstract

Reinforcement learning is a computational approach to understanding and automating goal-directed learning and decision-making [13] to maximise a numerical reward signal[12].Solutions to this class of problems have evolved from solving finite Markov decision processes [3] to developing model-free off-policy methods such as Soft Actor-Critic (SAC)[5]. This paper will discuss the latter while also evaluating the model's performance and experimenting with different approaches in the continuous Bipedal-Walker and BipedalWalkerHardcore environments.

1 Methodology

In this study, we implemented the Soft Actor-Critic (SAC), a sophisticated off-policy actor-critic method centred around Markov Decision Processes (MDPs)[6], essential for modelling environments with stochastic dynamics in reinforcement learning (RL). MDPs are defined by a set of states S, actions A, state-to-state transitions P, rewards R, and a discount factor γ, which influences the valuation of long-term rewards. Within this framework, the agent operates as a learner and a decision-maker. At each timestep, the agent observes a state from S, chooses an action from A, receives a reward from R, and transitions to a new state S_{t+1}. The policy π, guided by γ, dictates the agent's actions, allowing for strategic decision-making that balances immediate rewards with future gains. This off-policy approach enables the agent to learn from a behaviour policy different from the one being optimized, enhancing the flexibility and effectiveness of learning.[12]



[12]

We followed the standard procedures established by Tuomas Haarnoja in [5] to implement SAC. Our methodology involved setting up two separate neural networks: one for the policy π_ϕ and one for the action-value function Q_ϕ. These networks are updated asynchronously. The policy network is optimised using a combination of the expected return and the entropy of the policy. On the other hand, the action-value function is estimated using dual Q-networks to mitigate positive bias in the policy improvement step. It is updated through a Bellman equation modified to include an entropy term.

$$J(\pi_\theta) = \mathbb{E}_{s_t \sim D}\left[\mathbb{E}_{a_t \sim \pi_\theta}\left[\alpha \log \pi_\theta(a_t|s_t) - \min_{i=1,2} Q_{\phi_i}(s_t, a_t)\right]\right] \tag{1}$$

where α is the temperature that controls the importance of entropy vs reward, s_t are the states sampled from the replay buffer D, and α_t are the actions sampled according to the policy π_ϕ.

$$Q^\pi(s,a) = \mathbb{E}_{s' \sim P, a' \sim \pi}\left[r(s,a,s') + \gamma(Q^\pi(s',a') - \alpha \log \pi(a'|s'))\right] \tag{2}$$

where r is the reward recived after tacking action α in state s, leading to the next state s'. The term α' represents the next action sampled from the policy π_ϕ and γ weighs the importance of the future action.

The Kullback-Leibler Divergence [4] and reparameterisation trick[7] were pivotal in our SAC implementation. They enhanced stochastic policy optimisation by converting it into a deterministic form through Gaussian noise addition. This transformation permits direct gradient backpropagation through stochastic nodes, boosting learning efficiency and stability. Moreover, the training utilised a replay buffer to leverage diverse, uncorrelated transitions, allowing the agent to learn optimal behaviours across a broad spectrum of states not limited to immediate experiences. This off-policy method significantly improved the generalisation and robustness of the agent's performance.

In paper [4], they introduced a dynamically tuned temperature parameter α using a dual gradient descent method adapted from [2]. This approach involves truncated optimisations, involving a single gradient step, to adjust α efficiently. This automatic adjustment of α is essential for managing the trade-off between exploration and exploitation, especially in environments. This will be an extension of our model and will be referred to as sac v2.

$$J(\alpha) = \mathbb{E}_{a\sim\pi_\theta}\left[-\alpha \log \pi_\theta(a|s) - \alpha\bar{H}\right] \text{ where } \bar{H} = -\dim(A)$$

This approach will be applied to the BipedalWalker-v3 and BipedalWalkerHardcore-v3, both continuous action spaces with 4 actions and 24 observations [8]. An agent reaching the end will receive a score of 300, and if the agent falls, incur a penalty of 100.

2 Convergence Results

All experiments were conducted on Google Colab, and the ncc environments both using CPU high RAM. Each run of the algorithm was seeded with the value of 42.

2.1 experiment 1: Sac v1 vs sac v2

Comparing the performance of the two aforementioned implementations [11][10].

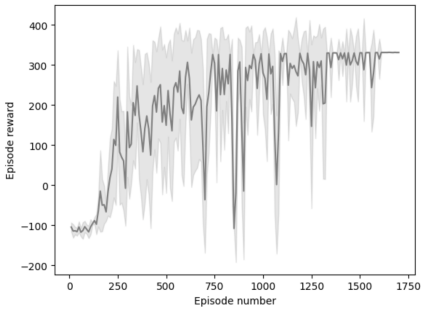


Figure 1: sac v1

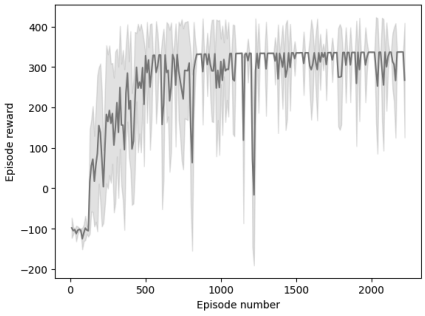


Figure 2: sac v2

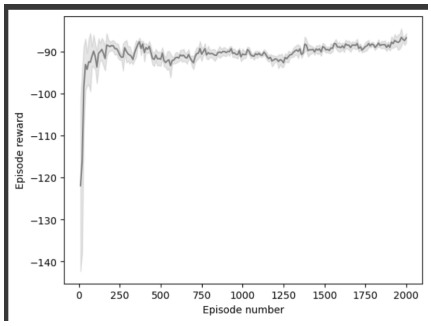
SAC v2 outperforms SAC v1 by dynamically adjusting the temperature parameter α, which finely tunes the balance between exploration and exploitation according to environmental changes. This adaptability allows SAC v2 to more effectively navigate the complex BipedalWalker environment, characterized by high variability in optimal strategies. In contrast, SAC v1, likely using a fixed α, struggles to adapt its strategy based on environmental feedback, resulting in lower performance and more significant reward fluctuations. Consequently, SAC v2 achieves higher reward peaks and a more stable learning trajectory, making it more suitable for dynamic and challenging environments.

2.2 experiment 2: sac v2 + LSTM

In the Soft Actor-Critic (SAC), integrating Long Short-Term Memory (LSTM) networks enhances the agent's ability to learn from sequential data and make informed decisions

in partially observable environments. The LSTM network, parameterised by θ, is embedded within the policy function $\pi_\theta(a_t|s_t)$, where it dynamically updates its hidden and cell states based on past observations and actions. This enables the policy network to capture correlations between temporal dependencies and long-term environmental dynamics.

using the implementation [9] inspired from paper [14] with the hyper-parameter with the soft Q, policy network and alpha learning rates = 0.0003, reward scale = 10, target entropy = -2, gamma = 0.99, batch size = 2.



The graph above shows that the reward over time using sac v2 + LSTM on the BipedalWalker environment seems constant. After observing the video recordings for each implementation, it was clear that the BipedalWalker was stuck in a local minima - with one knee joint on the ground, it would very incrementally inch forward. Since falling down incurs a penalty of negative -100. In normal circumstances, it may take hours to observe positive returns; the value of α may cause an improvement with this combination of strategies.

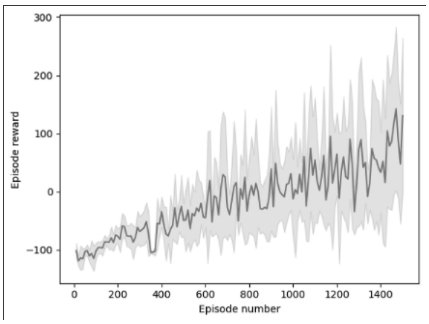
2.3 EXPERIMENT 3: HYPER-PARAMETER TUNING

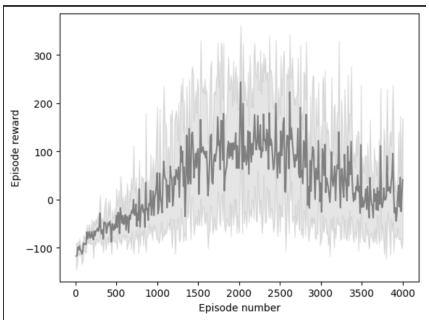
Comparative analysis identified SAC v2 as the superior model due to its dynamic balance of exploration and exploitation. Subsequently, the focus shifted to hyperparameter tuning, adjusting learning rates, reward scaling, and soft update coefficients to enhance performance. The objective was to fine-tune these parameters for sustained high performance across diverse scenarios.

	parameter	**walker**	**Hardcore**
0	Learning rate l_r	$3e-4$	$3e-4$
1	Discount factor	0.99	0.99
2	Polyak coefficient	0.002	0.002
3	Target entropy H	-4	-2
4	reply buffer size	$1,000,000$	$1,000,000$
5	Batch size	512	300
6	Hidden Layers	512	512
7	Activation Function	*ReLU*	*ReLU*
8	Optimizer	*Adam*	*Adam*

Table 1: The best parameters for sac v2

The results indicate optimal hyperparameters for standard and hardcore environments, with notable differences in target entropy and batch size based on the environment's complexity. For BipedalWalker v3, a target entropy of $H = -4$ and a batch size of 512 provide sufficient exploration and stability for less complex settings. In contrast, the more demanding BipedalWalkerHardcore-v3 requires a higher target entropy of $H = -2$ and a smaller batch size of 300 to enhance the agent's responsiveness and adaptability in navigating its challenging terrain. These hyperparameter adjustments ensure that SAC v2 effectively balances exploration and learning dynamics tailored to the specific needs of each environment.





2.4 EXPERIMENT 4: BIPEDALWALKERHARDCORE

In the BipedalWalkerHardcore-v3 environment, the SAC v2 algorithm faces challenges due to the randomness of terrain and obstacles. The algorithm exhibits high variance in rewards, which results in slow reward convergence. So, the training lengths were changed at random intervals starting with 1800, eventually building to training lengths of 4000 episodes. This experimentation solved the BipedalWalkerHardcore, with an agent reaching the end in a few iterations across the 4000 episodes.

2.5 BEST RESULTS

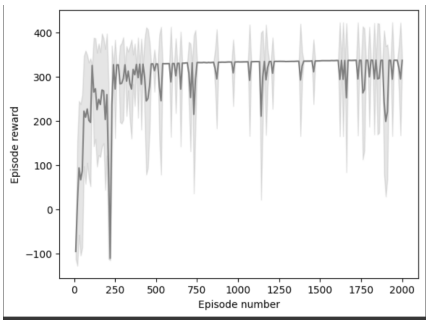


Figure 3: BipedalWalker-v3

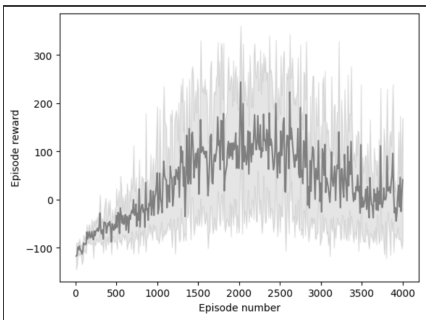


Figure 4: BipedalWalkerHardcore

The model in Figure 3 does converge and gets positive scores with a consistent uptrend with the odd anomaly; it consistently reaches a score of 300 with minimal deviation, as shown by the minimal shadow, making it sample efficient. Figure 4 demonstrates a clear trend of improvement in episode rewards over 4000 episodes, suggesting effective convergence despite high-performance variability. This pattern indicates a non-zero probability that, despite fluctuations, the agent is gradually learning to optimise its policy, likely refining its strategies over time in response to complex and varying environmental challenges.

3 LIMITATIONS

Despite extensive training, this study found that the agent struggled to produce a optimal rewards trajectory in the BipedalWalkerHardcore environment.

FUTURE WORK

To inspire future breakthroughs, the study recommends extending the training duration, exploring more robust reinforcement learning frameworks, and incorporating sophisticated techniques such as Hindsight Experience Replay (HER) [1].

References

[1] Marcin Andrychowicz et al. *Hindsight Experience Replay*. 2018. arXiv: 1707.01495 [cs.LG].

[2] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[3] Wikipedia contributors. *Bellman equation – Wikipedia, The Free Encyclopedia*. [Online; accessed 10-February-2022]. 2021. URL: https://en.wikipedia.org/w/index.php?title=Bellman_equation&oldid=1061087881.

[4] Tuomas Haarnoja et al. *Soft Actor-Critic Algorithms and Applications*. 2019. arXiv: 1812.05905 [cs.LG].

[5] Tuomas Haarnoja et al. "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.

[6] Beakcheol Jang et al. "Q-Learning Algorithms: A Comprehensive Classification and Applications". In: *IEEE Access* 7 (2019), pp. 133653–133667. DOI: 10.1109/ACCESS.2019.2941229.

[7] OpenAI. *Soft Actor-Critic*. Accessed: 2024-05-07. 2019. URL: https://spinningup.openai.com/en/latest/algorithms/sac.html.

[8] OpenAI Gym. *BipedalWalker-v3*. GitHub repository. Accessed: 2024. URL: https://github.com/openai/gym/wiki/BipedalWalker-v3.

[9] quantumiracle. *SAC v2 LSTM Implementation*. GitHub repository. Accessed: 2024-05-07. 2019. URL: https://github.com/quantumiracle/Popular-RL-Algorithms/blob/master/sac_v2_lstm.py.

[10] quantumiracle. *Soft Actor-Critic (SAC) Implementation*. GitHub repository. Accessed: 2024-05-07. 2023. URL: https://github.com/quantumiracle/Popular-RL-Algorithms/blob/master/sac.py.

[11] quantumiracle. *Soft Actor-Critic Version 2 Implementation*. GitHub repository. Accessed: 2024-05-07. 2023. URL: https://github.com/quantumiracle/Popular-RL-Algorithms/blob/master/sac_v2.py.

[12] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: http://incompleteideas.net/book/the-book-2nd.html.

[13] Chris Watkins. *Lecture 1: Introduction to Reinforcement Learning*. 2018. URL: https://cwkx.github.io/data/teaching/reinforcement-learning/rl-lecture1.pdf.

[14] Zhihan Yang and Hai Nguyen. *Recurrent Off-policy Baselines for Memory-based Continuous Control*. 2021. arXiv: 2110.12628 [cs.LG].