Soft Actor Critic for Walker2D

Huzaifa Jawad
School of Electrical Engineering and Computer Science
CS 11-B
Islamabad, Pakistan
hjawad.bscs21seecs@seecs.edu.pk

***Abstract*—Continuous control tasks, such as those in robotics and autonomous systems, present significant challenges due to high-dimensional state and action spaces, sparse rewards, and complex dynamics. The Walker2D environment, a benchmark problem from the MuJoCo physics engine, exemplifies these difficulties and serves as an ideal testbed for advancing reinforcement learning techniques. In this work, we employ the Soft Actor-Critic (SAC) algorithm, a state-of-the-art off-policy reinforcement learning approach, to solve the Walker2D task. SAC's entropy-regularized objective enables robust exploration and efficient learning, while its off-policy framework ensures effective data utilization. Our implementation demonstrates that SAC outperforms many other algorithms by achieving stable and sample-efficient training, resulting in an agent capable of smooth, coordinated walking behaviors. Quantitative results show significant improvements in cumulative rewards and control stability. This study underscores SAC's potential to tackle real-world continuous control problems and highlights opportunities for future research, such as integrating hierarchical reinforcement learning for more complex tasks.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Reinforcement Learning (RL) is a branch of machine learning that focuses on training agents to make sequential decisions by interacting with their environment. Unlike supervised learning, where models are trained on labeled data, RL agents learn by receiving feedback in the form of rewards or penalties based on their actions. This paradigm enables RL to solve complex control tasks that require sequential planning and adaptability.

The MuJoCo (Multi-Joint dynamics with Contact) physics engine has emerged as a popular benchmark for testing and evaluating RL algorithms. It provides high-fidelity simulation environments with continuous control tasks, making it ideal for evaluating the performance of RL algorithms in settings that closely resemble real-world scenarios. Among the various tasks offered by MuJoCo, the Walker2D environment is particularly challenging due to its requirement for an agent to balance, walk, and avoid falling while navigating a high-dimensional state and action space.

The Walker2D environment, in particular, poses an interesting and hard problem. The task involves training a two-dimensional bipedal robot to walk efficiently without falling. This environment is challenging due to several factors:

Identify applicable funding agency here. If none, delete this.

- **Control Dynamics:** The agent must learn to coordinate multiple joints in real-time to maintain stability while moving forward.
- **State Space Complexity:** The high-dimensional state space includes positional and velocity information for each joint, making the problem computationally demanding.
- **Sparse Rewards:** The reward function is primarily dependent on forward movement, with penalties for falling, requiring the agent to learn subtle control strategies to achieve stability and balance.

To address the challenges of Walker2D, we utilize the Soft Actor-Critic (SAC) algorithm, a state-of-the-art method designed for continuous control tasks. Actor-Critic algorithms, in general, are well-suited for such environments as they combine the benefits of policy-based and value-based methods. SAC, in particular, extends the standard actor-critic framework by incorporating an entropy-regularized objective. This encourages the agent to maintain a diverse set of actions, which is critical for effective exploration in environments with sparse rewards and complex dynamics.

SAC offers several advantages that make it a suitable choice for solving the Walker2D environment:

- **Stability:** SAC leverages stochastic policies and entropy maximization, which help prevent premature convergence to suboptimal solutions.
- **Sample Efficiency:** The algorithm optimizes both the actor and critic simultaneously, ensuring efficient use of training data.
- **Scalability:** SAC is capable of handling high-dimensional action spaces, making it ideal for continuous control tasks like Walker2D.

In this project, we implement the SAC algorithm to solve the Walker2D environment. We conduct conditional hyperparameter tuning for key parameters, including the target smoothing coefficient (τ), entropy coefficient (ent_coef), and learning rate, to evaluate their impact on performance. The agent is trained for 1,000,000 steps with both default and tuned hyperparameters. Performance is assessed through the evaluation of mean reward over the past 100 episodes, and a video of the agent's behavior is recorded to visualize its learning progress. This report provides a comprehensive analysis of the agent's performance, convergence behavior, and sensitivity to hyperparameter changes.

II. WALKER2D ENVIRONMENT

The Walker2D environment is a challenging continuous control task from the MuJoCo physics engine, designed to test and benchmark reinforcement learning (RL) algorithms in high-dimensional and dynamic settings. This environment builds upon the Hopper task by introducing an additional set of legs, allowing the robot to walk forward rather than hop. The robot is a two-dimensional bipedal walker consisting of seven main body parts: a torso, two thighs, two legs, and two feet. These are interconnected by six hinges, each controlled through torques, creating a complex control problem.

A. Relevance as a Complex Problem

The Walker2D environment exemplifies the difficulties inherent in real-world control problems, making it an excellent benchmark for RL algorithms. Its relevance stems from several factors:

- **High-Dimensional State Space:** The environment's 17-dimensional observation space includes positional and velocity data (`qpos` and `qvel`) for each of the robot's body parts. This increases the computational complexity and requires the RL agent to process and act on a large amount of information.
- **Continuous Action Space:** The six-dimensional action space, defined as `Box(-1.0, 1.0, (6,), float32)`, represents the torques applied to the robot's joints. Effective control of these joints requires fine-grained adjustments to maintain balance and forward movement.
- **Dynamic Control Requirements:** The agent must coordinate multiple joints in real time to ensure stability while walking. This involves tackling challenges like avoiding falls, compensating for unexpected perturbations, and maximizing forward velocity.
- **Sparse Rewards:** The reward function primarily incentivizes forward movement, with penalties for falling. This sparse feedback forces the agent to learn subtle and efficient strategies for controlling the robot's joints.

B. Unique Challenges

The unique challenges of the Walker2D environment arise from its intricate control dynamics and physical realism:

- **Balancing Stability and Motion:** The agent must learn to walk smoothly without toppling over, which requires precise coordination of the six torque-controlled hinges.
- **Exploration in High-Dimensional Spaces:** Navigating the 17-dimensional observation space and 6-dimensional action space demands robust exploration strategies to avoid getting stuck in suboptimal solutions.
- **Interdependency of Actions:** Actions applied to one joint affect the overall stability and motion of the robot, making the control dynamics highly non-linear and interdependent.

In summary, the Walker2D environment is not only a compelling testbed for evaluating RL algorithms but also a representation of real-world control problems that demand sophisticated strategies for balancing, coordination, and exploration. These attributes make it a relevant and challenging environment for testing the efficacy of the Soft Actor-Critic (SAC) algorithm.

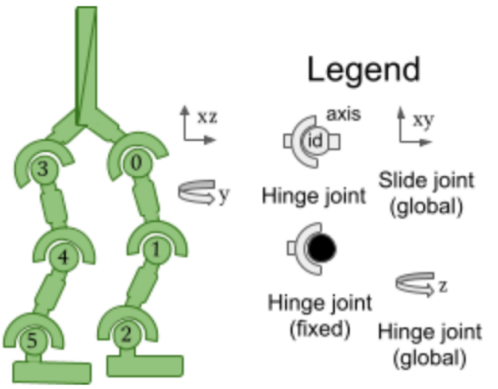

Fig. 1. Walker-2D has 7 different body parts. 1 torso, 2 thighs, 2 lower legs and 2 feet. It also has 3 joints on each legs

III. ALGORITHM JUSTIFICATION

A. Selected RL Algorithm

The Soft Actor-Critic (SAC) algorithm is a state-of-the-art reinforcement learning (RL) algorithm designed for continuous action spaces. It is an off-policy actor-critic method that incorporates entropy maximization within the maximum entropy reinforcement learning framework. The algorithm extends the conventional RL objective by encouraging the agent to maximize both the expected reward and the entropy of its policy. This ensures the agent explores diverse actions, which is critical for solving environments with complex dynamics.

SAC operates with three key components:

- **Actor:** A stochastic policy network responsible for selecting actions.
- **Critic:** Two Q-function networks to evaluate the expected return of actions and mitigate overestimation bias.
- **Entropy Term:** A tunable entropy coefficient to balance exploration and exploitation by encouraging stochastic policies.

The off-policy nature of SAC allows it to reuse past experiences stored in a replay buffer, significantly improving sample efficiency.

B. Rationale for Selection

SAC was selected for the Walker2D environment due to its robust performance on high-dimensional and dynamic tasks. The environment poses several challenges, including balancing, coordinated walking, and recovering from potential falls. These challenges align well with SAC's strengths, as discussed below:

1) High Sample Complexity: One of the primary limitations of many RL algorithms is their high sample complexity, particularly for on-policy methods. SAC addresses this limitation by being off-policy, allowing it to learn efficiently from previously collected data stored in a replay buffer. This characteristic makes it highly suitable for the Walker2D environment, which involves continuous and high-dimensional state and action spaces.

2) Brittle Convergence Properties: Many RL algorithms, such as DDPG, suffer from instability and brittleness during training due to their deterministic policies and sensitivity to hyperparameters. SAC, in contrast, employs stochastic policies and an entropy maximization objective, leading to more stable and robust training. This ensures reliable convergence even in the presence of challenging control dynamics in Walker2D.

3) Exploration vs. Exploitation Balance: SAC's maximum entropy objective explicitly encourages exploration, which is essential in the Walker2D environment. The complex dynamics of the robot, such as balancing and walking, require the agent to explore diverse strategies before converging on an optimal one. The entropy regularization in SAC helps prevent premature convergence and improves policy robustness.

4) Robustness to Hyperparameters: While hyperparameter tuning can be computationally intensive, SAC demonstrates robustness across a wide range of hyperparameters. This reduces the need for meticulous tuning, making it a practical choice for Walker2D. Moreover, SAC consistently achieves good results in literature, further validating its applicability to tasks with complex dynamics.

5) Sample Efficiency and Stability: SAC's off-policy design, combined with the use of two Q-functions, ensures high sample efficiency and stability during training. The Walker2D environment benefits significantly from these features, as the agent must process complex state transitions and optimize over a continuous action space efficiently.

C. Why SAC Performs Well on Walker2D

The following characteristics of SAC make it particularly well-suited for solving the Walker2D environment:

- **Handling Complex Dynamics:** The ability to balance and walk in Walker2D relies on SAC's robust exploration and stable training. The entropy objective ensures the agent learns diverse policies, which are critical for overcoming falls and achieving forward motion.
- **Efficient Learning:** SAC's off-policy updates allow the reuse of experiences, reducing the computational cost and training time compared to on-policy methods like PPO.
- **Consistent Performance:** SAC has demonstrated superior performance in continuous control tasks, including Walker2D, in terms of both sample efficiency and asymptotic performance, as shown in benchmark comparisons.

In summary, SAC's combination of stability, sample efficiency, and robustness makes it an excellent choice for tackling the Walker2D environment's high-dimensional and dynamic control challenges.

IV. Algorithm Implementation

The implementation of the Soft Actor-Critic (SAC) algorithm was carried out using the `stable-baselines3` library, which provides efficient and well-documented tools for reinforcement learning. The following describes the implementation details, including hyperparameter tuning, training setup, and monitoring processes.

A. Implementation Details

- **Environment Handling:** The Walker2D environment was instantiated using the `gymnasium.make("Walker2d-v5")` API. Proper preprocessing steps were applied, including reward normalization and observation scaling, to ensure stable learning dynamics.
- **Library and Frameworks:** The SAC algorithm was implemented using `stable-baselines3`, a PyTorch-based library known for its efficiency and compatibility with custom environments. TensorBoard was integrated to monitor training logs and visualize performance metrics in real time.
- **Reward Shaping and Normalization:** The default reward structure of Walker2D was used without modifications. Rewards were normalized to stabilize the learning process and improve convergence.
- **Replay Buffer:** A replay buffer of size 10^6 was used to store past experiences for off-policy training, ensuring efficient sample reuse and stability.
- **Logging and Monitoring:** TensorBoard was utilized to log training metrics such as episodic rewards, loss values, and entropy during the training process.

B. Hyperparameter Tuning

Hyperparameter tuning was conducted to optimize SAC's performance on the Walker2D environment. Due to limited computational resources, grid search was not feasible; instead, conditional tuning was employed to focus on the most impactful parameters. Key hyperparameters and their ranges are listed below:

```
# Hyperparameters to tune
tau_values = [0.1, 0.01, 0.001, 0.0001]
ent_coef_values = [1.0, 0.3333, 0.1, 0.03333, 0.01
learning_rate_values = [1e-3, 3e-4, 1e-4, 3e-5, 1e
```

The tuning process involved training the agent for 300,000 timesteps for each combination of hyperparameters and evaluating their impact on the agent's performance. Based on these experiments:

- The target smoothing coefficient (τ) was set to 0.01, slightly higher than the default value of 0.005, for improved training stability.
- The entropy coefficient (ent_coef) was tuned to 0.1, providing a balanced trade-off between exploration and exploitation.
- The learning rate was selected as 3×10^{-4} for stable convergence.

C. Training Setup

- **Training Duration:** The final agent was trained for 1,000,000 timesteps with the tuned hyperparameters.
- **Exploration Strategy:** SAC's entropy maximization objective was leveraged to encourage stochastic policies and robust exploration, critical for learning the complex dynamics of Walker2D.
- **Optimizer:** The Adam optimizer was used for all networks, with the learning rate set according to the tuned parameters.
- **Replay Buffer Sampling:** Mini-batches of size 256 were sampled from the replay buffer for each gradient update.

D. Code Efficiency and Documentation

The implementation followed best practices for readability and efficiency:

- Modular functions were created for environment setup, training, and evaluation to promote reusability.
- Each function and key segment of code was thoroughly documented, ensuring clarity for future debugging and optimization.
- Training performance metrics, including episodic rewards and loss values, were plotted in real time using TensorBoard for efficient monitoring.

E. Challenges and Adjustments

Given the computational constraints, hyperparameter tuning was limited to conditional tuning rather than exhaustive grid search. This necessitated a focus on the most impactful parameters, such as τ, ent_coef, and learning rate. The selected parameters proved effective in achieving stable convergence and robust performance in the Walker2D environment.

In summary, the SAC algorithm was implemented effectively using `stable-baselines3`, with careful consideration of environmental handling, reward scaling, and hyperparameter tuning. This ensured robust training and sample-efficient learning for the Walker2D task.

Hyperparameter Tuning in Soft Actor-Critic (SAC)

Entropy Coefficient (`ent_coef`)

- **Purpose:** Encourages exploration by adding entropy to the policy loss.
- **Observations:**
  - `ent_coef` $= 0.01$: Slower learning due to limited exploration.
  - `ent_coef` $= 0.1$ to 0.333: Optimal balance between exploration and exploitation, leading to higher rewards.
  - `ent_coef` $= 1.0$: Over-exploration causing instability in learning.

Learning Rate (`lr`)

- **Purpose:** Determines the magnitude of updates to policy and value networks.
- **Observations:**
  - `lr` $= 10^{-5}$: Learning is very slow and fails to achieve high rewards.
  - `lr` $= 0.0001$ to 0.001: Achieves faster convergence with stable learning.
  - `lr` $= 0.01$: Risk of instability or divergence.

Target Update Coefficient (`tau`)

- **Purpose:** Controls the soft update rate of the target network.
- **Observations:**
  - `tau` $= 0.0001$: Conservative updates lead to slower learning.
  - `tau` $= 0.01$: Balanced updates achieve better learning performance.
  - `tau` $= 0.1$: Faster updates result in quicker convergence but may compromise stability.

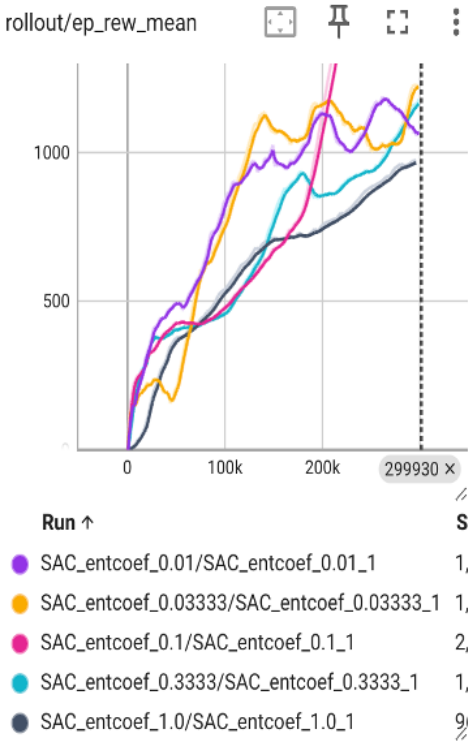

Fig. 2. How different Entropy coefficient changes the mean reward.

V. Training Details

The training process was conducted using the Soft Actor-Critic (SAC) algorithm implemented with the `stable-baselines3` library. Below, we detail the training setup, the hyperparameter tuning process, and the results obtained.

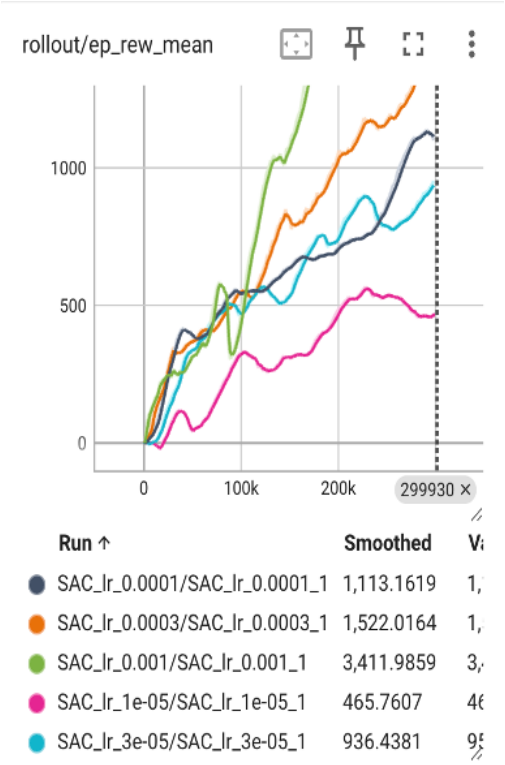

Fig. 3. How different learning rates changes the mean reward.

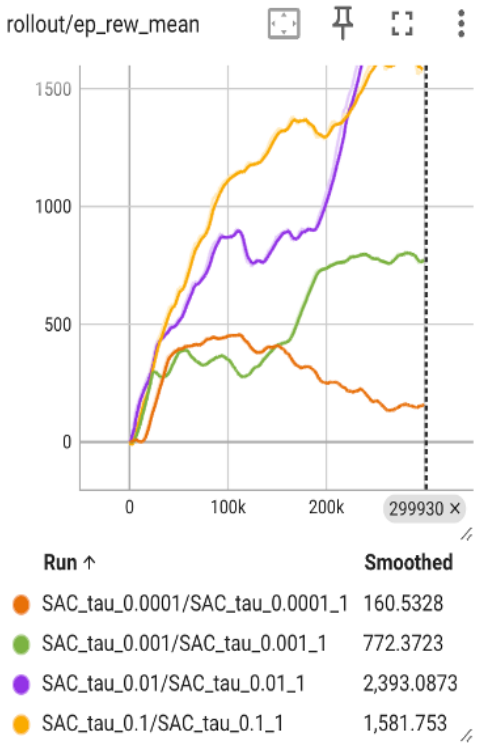

Fig. 4. How different tau changes the mean reward

A. Training with Default Parameters

The agent was initially trained using the default SAC hyperparameters provided by `stable-baselines3`. Training was conducted for a total of 1,000,000 timesteps in the Walker2D environment. The results of this training were as follows:

- Total episodes: 1254
- Mean episode length: 976 timesteps
- Mean episodic reward: 4140.84

While the results were satisfactory, analysis indicated room for improvement in the reward maximization and agent performance, motivating hyperparameter tuning.

B. Hyperparameter Tuning

Due to computational constraints, a full grid search was infeasible. Instead, conditional tuning was employed to optimize the following key hyperparameters:

- τ (Target Smoothing Coefficient): $[0.1, 0.01, 0.001, 0.0001]$
- Entropy Coefficient (ent_coef): $[1.0, 0.3333, 0.1, 0.03333, 0.01]$
- Learning Rate: $[1e-3, 3e-4, 1e-4, 3e-5, 1e-5]$

Each parameter was tuned independently by training the model for 300,000 timesteps. During this process:

- τ was updated to 0.01 for improved stability in target updates.
- The entropy coefficient (ent_coef) was set to 0.1 to balance exploration and exploitation.
- The learning rate was fixed at 3×10^{-4}, ensuring smooth convergence without overshooting.

The tuning process utilized a custom callback, `SaveOnBestTrainingRewardCallback`, to monitor the mean reward over the last 100 episodes and save the best-performing model. The callback facilitated periodic evaluation of training progress and ensured that the best hyperparameters were identified for further training.

C. Training with Tuned Parameters

Following hyperparameter tuning, the model was trained with the optimized parameters for 1,000,000 timesteps. The results from this training were:

- Total episodes: 1260
- Mean episode length: 972 timesteps
- Mean episodic reward: 4948.95

This represented a significant improvement over the training with default parameters, both in terms of episodic reward and overall performance stability.

D. Monitoring and Evaluation

- **Monitoring:** Training metrics, including episodic rewards and loss values, were logged using TensorBoard. This provided real-time insights into the training process and allowed for efficient debugging and analysis.

- **Evaluation:** The trained agent was evaluated over 100 episodes, with the mean and standard deviation of rewards recorded. Additionally, a video of the agent's performance was recorded, demonstrating its learned behavior in the Walker2D environment.

E. Inference and Visualization

After training, the agent was deployed for inference in the Walker2D environment. The agent successfully demonstrated the ability to walk stably while maximizing forward velocity. A video recording captured the agent's behavior, showcasing its ability to handle the environment's complex dynamics effectively.

In summary, the training process was designed to maximize the agent's performance through careful hyperparameter tuning and efficient implementation of the SAC algorithm. The significant improvement in reward and stability highlights the effectiveness of the chosen approach.

VI. Conclusion

In this project, we successfully implemented the Soft Actor-Critic (SAC) algorithm to solve the Walker2D environment, a challenging continuous control task from the MuJoCo physics engine. The Walker2D environment, with its high-dimensional state and action spaces, sparse rewards, and complex control dynamics, served as an excellent testbed for evaluating the performance of SAC.

Through our implementation, we demonstrated SAC's ability to address the challenges posed by Walker2D by leveraging its key strengths, including stability, sample efficiency, and robust exploration strategies. The algorithm's entropy-regularized objective encouraged diverse policy exploration, enabling the agent to learn nuanced control strategies for balancing, walking, and avoiding falls. Additionally, SAC's off-policy nature allowed for efficient utilization of training data, ensuring faster convergence compared to many other reinforcement learning algorithms.

We also conducted hyperparameter tuning to optimize the agent's performance further, evaluating critical parameters such as the target smoothing coefficient, entropy coefficient, and learning rate. Our analysis showed that properly tuned hyperparameters significantly improved the agent's ability to balance exploration and exploitation, resulting in higher mean rewards and smoother walking behaviors.

The results highlight SAC's potential for solving real-world continuous control problems that require sophisticated strategies for coordination and adaptability. Beyond Walker2D, the insights gained from this project can inform the application of SAC to other complex environments and tasks, such as robotics and autonomous systems. Future work could explore the integration of advanced exploration techniques or hierarchical reinforcement learning frameworks to address even more complex challenges.

Overall, this study reinforces the effectiveness of SAC as a state-of-the-art reinforcement learning algorithm for high-dimensional and dynamic control tasks, demonstrating its capability to solve problems that mimic the intricacies of real-world scenarios.