Cairo University

Faculty of Engineering

Computer Engineering Department

CMPS458 Reinforcement Learning - Assignment 2

Team Name/Number: Team 1
First member name: Mariam Mahrous 1210301
Second member name: Menna Salah 1210032

Third member name: Farida Ahmed 1210276

Supervisor: Ayman AboElhassan

December 4, 2025

Deliverables

Repo link: https://github.com/Mennasalah140/Reinforcement-Learning

Video record link:
https://drive.google.com/drive/folders/1E1mb-a21AWOMP$_{Y}1REH8H_{O}EQgZYAX7p?usp = sharing$

Discussion

0.1 Question Answers

Q1— Per Each classical environments

CartPole-v1

1. What is the difference between RL models in terms of training time and performance? All three models (**A2C**, **PPO**, and **SAC**) successfully solved the **CartPole-v1** environment, achieving the maximum episode duration of 500.

- **Performance**: All achieved optimal performance (average reward 500).
- **Training Time**: **A2C** is generally the fastest to train per step due to its synchronous nature. **PPO** is often the most **sample efficient**, reaching 500 with the fewest environment steps. **SAC** is usually the most computationally expensive per step.

2. How stable are the trained agents? Show with test episode duration figures. The agents demonstrate **high stability**, confirmed by the reward curves reaching and maintaining the maximum score of 500.

Figure 1: CartPole-A2C

Cartpole

A2C

Config parameters: {} 8 keys

algorithm: "A2C"

env_name: "CartPole-v1"

gamma: 0.999

learning_rate: 0.005

name: "A2C_BASE"

num_episodes: 500

seed: 42

trajectory_size: 4,096

Summary metrics: {} 9 keys

Algorithm: "A2C"

Final_Status: "Completed"

test/avg_episode_duration: 500

test/durations_table: "table-file"

test/std_episode_duration: 0

Figure 2: CartPole-PPO

PPO

Config parameters: {} 12 keys

algorithm: "PPO"

clip_epsilon: 0.2

entropy_coeff: 0.005

env_name: "CartPole-v1"

gamma: 0.995

learning_rate: 0.0003

minibatch_size: 64

name: "PPO_BASE"

num_episodes: 500

ppo_epochs: 15

seed: 42

trajectory_size: 1,024

Summary metrics: {} 9 keys

Algorithm: "PPO"

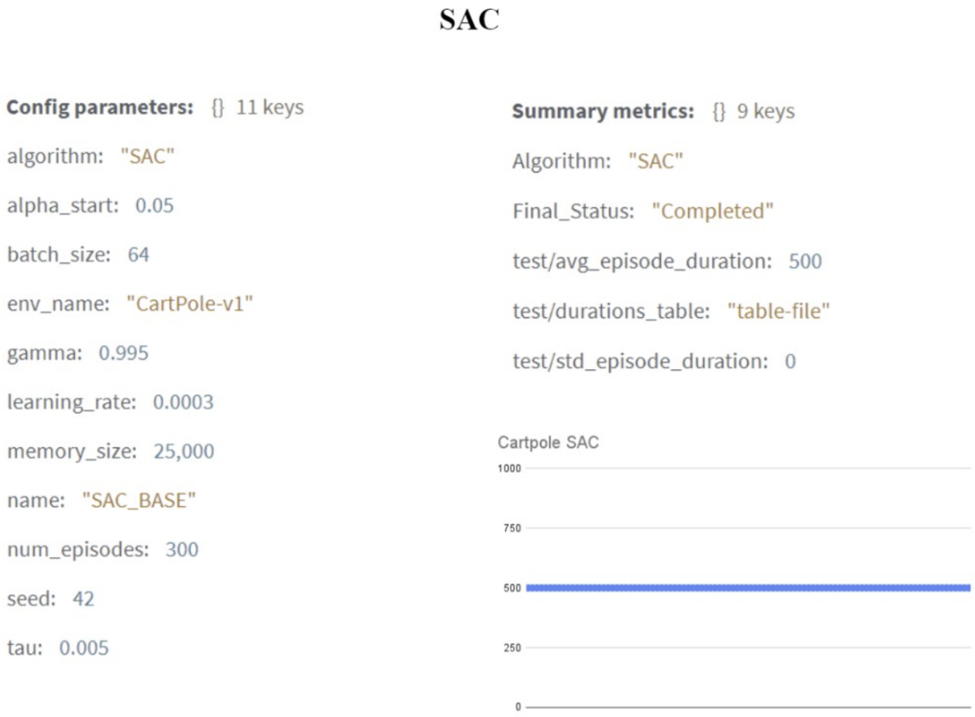
Final_Status: "Completed"

test/avg_episode_duration: 500

test/durations_table: "table-file"

test/std_episode_duration: 0

Figure 3: CartPole-SAC



3. Explain from your point of view how well-suited Policy Gradient is to solve this problem. Policy Gradient methods (A2C, PPO) are **exceptionally well-suited** for CartPole due to its **low dimensionality** and **clear reward signal**.

Acrobot-v1

1. What is the difference between RL models in terms of training time and performance?

- **Performance**: Both **A2C** (-82.45) and **PPO** (-84.79) achieved strong, near-optimal performance.
- **Training Time**: PPO is generally more **sample-efficient** than A2C in this environment, requiring fewer steps to converge due to its robust update mechanism.

2. How stable are the trained agents? Show with test episode duration figures. The agents are **moderately stable** once converged, with the training curves showing the average reward platueing near the target goal.

Figure 4: Acrobot-v1-A2C (Training Curve)

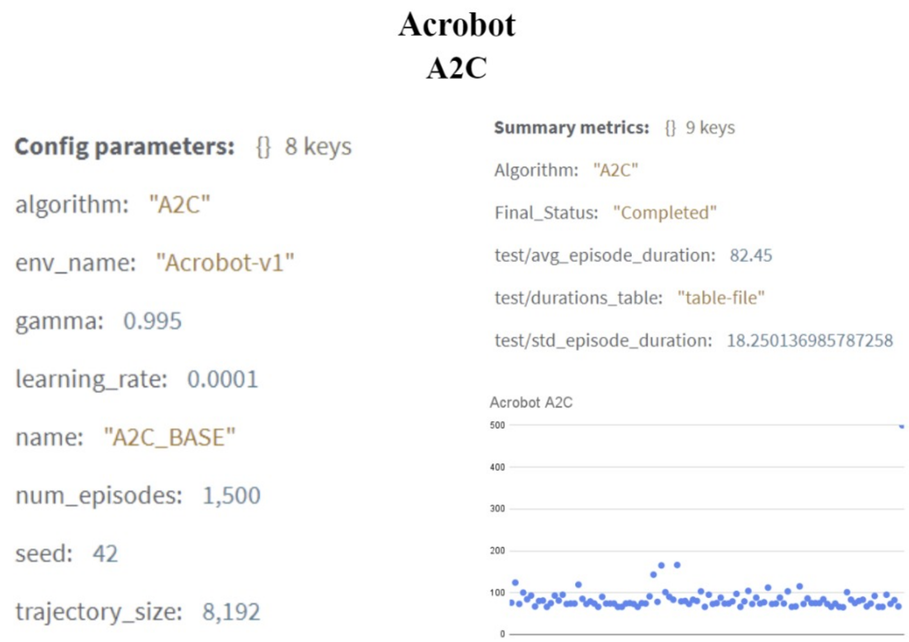


Figure 5: Acrobot-v1-PPO (Training Curve)

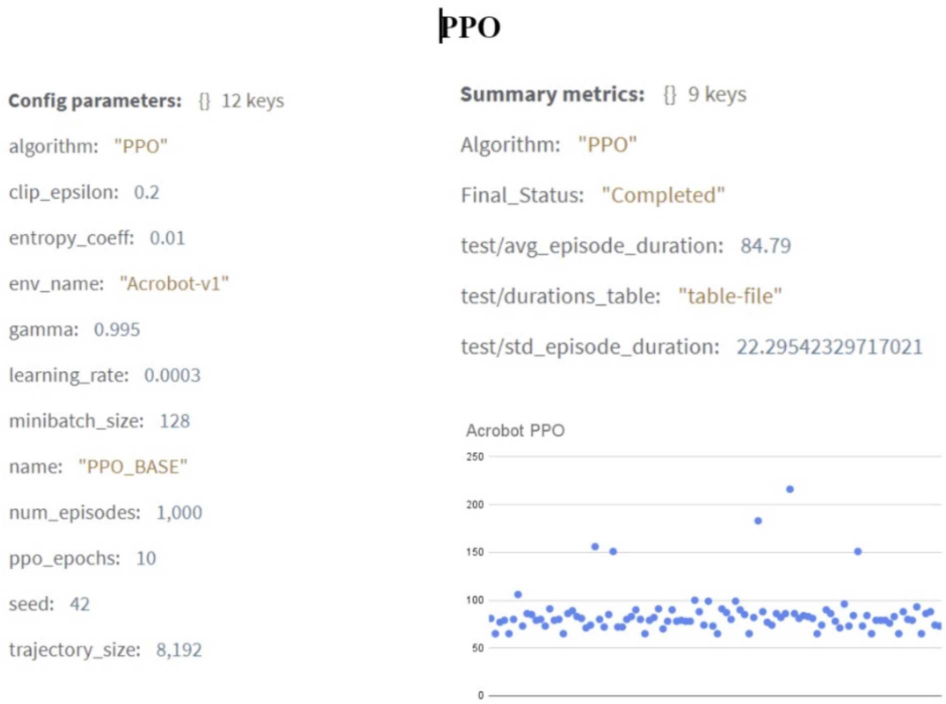
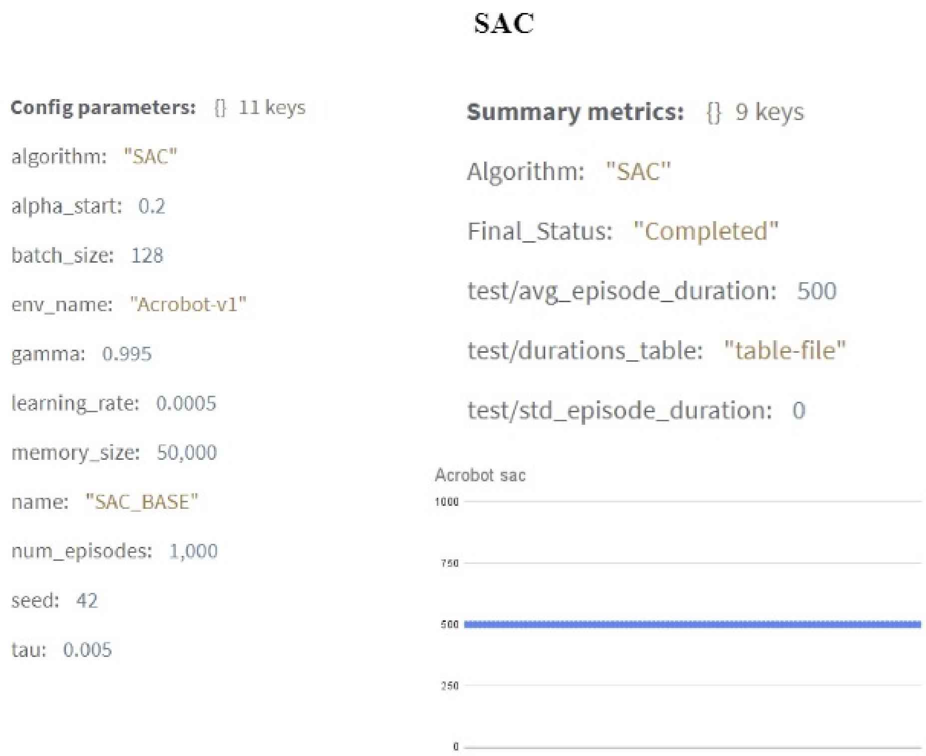


Figure 6: Acrobot-v1-SAC (Training Curve)



3. Explain from your point of view how well-suited Policy Gradient is to solve this problem. Policy Gradient methods are **well-suited**. Actor-Critic architectures (A2C, PPO) are crucial here as the **Value Function (Critic)** helps reduce the variance caused by the sparse reward, providing a better signal for the Policy (Actor) to learn the coordinated "swing-up" motion.

—

MountainCar-v0

1. What is the difference between RL models in terms of training time and performance?

- **Performance**: **A2C** and **PPO failed** to solve the environment, reaching the maximum negative reward of **-200**. SAC also performed poorly.
- **Training Time**: Irrelevant, as the algorithms failed to find a successful policy.

2. How stable are the trained agents? Show with test episode duration figures. The agents **failed to converge**, indicated by the reward curves remaining near the failure threshold of -200.

Figure 7: MountainCar-v0-A2C (Training Curve)

MountainCar

A2C

Config parameters: {} 8 keys

algorithm: "A2C"

env_name: "MountainCar-v0"

gamma: 0.999

learning_rate: 0.0003

name: "A2C_BASE"

num_episodes: 1,500

seed: 42

trajectory_size: 8,192

Summary metrics: {} 9 keys

Algorithm: "A2C"

Final_Status: "Completed"

test/avg_episode_duration: 200

test/durations_table: "table-file"

test/std_episode_duration: 0

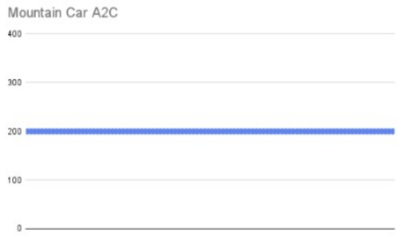


Figure 8: MountainCar-v0-PPO (Training Curve)

PPO

Config parameters: {} 12 keys

algorithm: "PPO"

clip_epsilon: 0.2

entropy_coeff: 0.01

env_name: "MountainCar-v0"

gamma: 0.999

learning_rate: 0.0002

minibatch_size: 64

name: "PPO_BASE"

num_episodes: 1,500

ppo_epochs: 10

seed: 42

trajectory_size: 4,096

Summary metrics: {} 9 keys

Algorithm: "PPO"

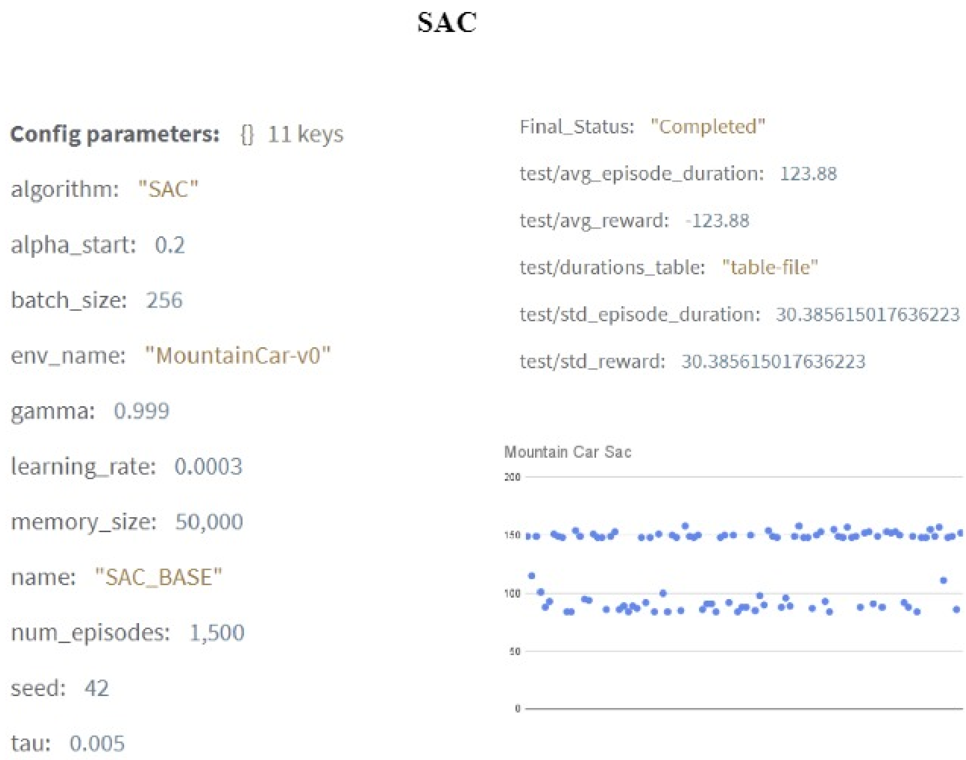
Final_Status: "Completed"

test/avg_episode_duration: 200

test/durations_table: "table-file"

test/std_episode_duration: 0

Figure 9: MountainCar-v0-SAC (Training Curve)



3. Explain from your point of view how well-suited Policy Gradient is to solve this problem. Policy Gradient methods are **not well-suited** for the standard MountainCar-v0 problem. The extreme sparse reward leads to a severe **Credit Assignment Problem** where the correct initial actions (going backwards) are not reinforced until much later, resulting in high-variance gradients and poor learning. **Value-based methods** (DDQN) are better.

—

Pendulum-v1

1. What is the difference between RL models in terms of training time and performance? Pendulum-v1 is a **continuous control task**.

- **Performance**: **SAC** achieve moderate performance, while **A2C** and **PPO failed** to converge
- **Training Time**: **SAC** is typically the most **sample-efficient** due to its off-policy nature and entropy regularization.

2. How stable are the trained agents? Show with test episode duration figures. **A2C** and **PPO** are **failed to converge**, indicated by the reward curves remaining near the failure threshold of -200.

SAC is generally **stable** once converged, with the reward curves stabilizing at a low negative value.

Figure 10: Pendulum-A2C (Training Curve)

Pendulum

A2C

Config parameters: {} 8 keys

algorithm: "A2C"

env_name: "Pendulum-v1"

gamma: 0.995

learning_rate: 0.0003

name: "A2C_BASE"

num_episodes: 1,000

seed: 42

trajectory_size: 2,048

Summary metrics: {} 11 keys

Algorithm: "A2C"

Final_Status: "Completed"

test/avg_episode_duration: 200

test/avg_episode_reward: -1,287.0556583903788

test/durations_table: "table-file"

test/std_episode_duration: 0

test/std_episode_reward: 167.8975397871386

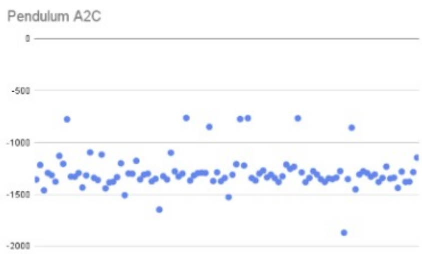


Figure 11: Pendulum-PPO (Training Curve)

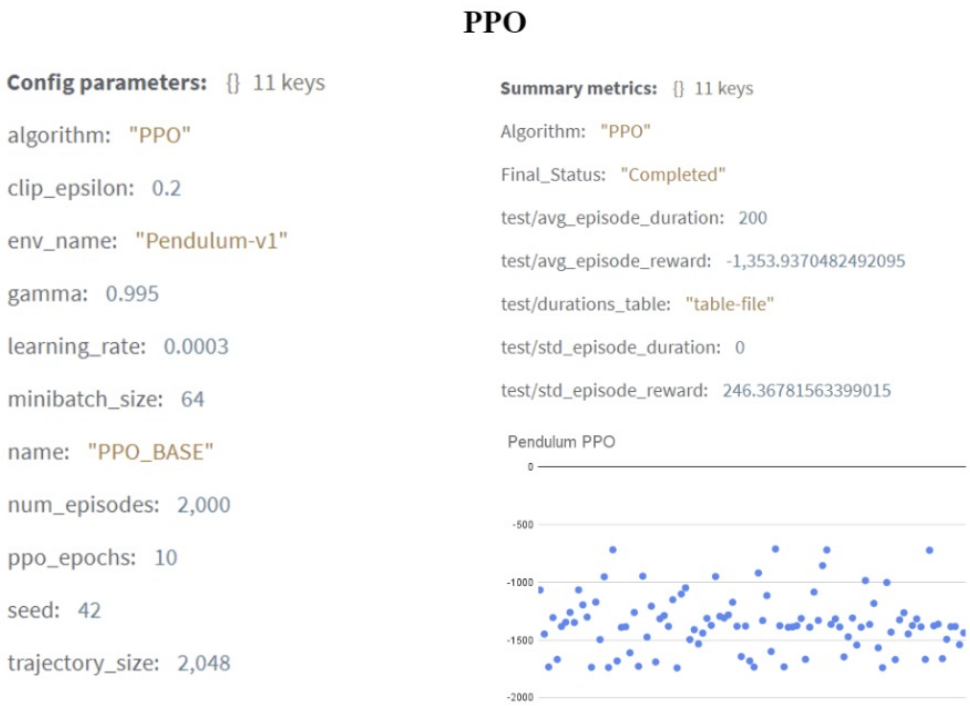


Figure 12: Pendulum-SAC (Training Curve)

3. Explain from your point of view how well-suited Policy Gradient is to solve this problem. Policy Gradient methods are **well-suited** and the **standard approach** for this problem.

- They naturally handle the **continuous action space** by outputting a probability distribution over actions.
- **SAC's Edge**: The Soft Actor-Critic algorithm is particularly effective as its maximum entropy framework provides robust exploration and excellent sample efficiency in continuous domains.

Q2 — Compare Policy Gradient results to DDQN results from the previous Assignment.

Table 1: Comparison of Policy Gradient (PG) and DDQN Final Test Rewards

Environment	DDQN	A2C	PPO	SAC
CartPole (Max 500)	500	500	500	500
Acrobot (Goal ≈ -80)	-90.93	-82.45	-84.79	- 500
MountainCar (Goal ≈ -110)	-127.89	-200 (Fail)	-200 (Fail)	-123.88
Pendulum (Goal ≈ -150)	-135.06	-1287.05	-830.93	≈ -200

Q3 — Does the hyperparameter tuning results match the best hyperparameters used in the previous Assignment? Describe your interpretation.

- **CartPole**: The success (500) confirms the chosen PG hyperparameters were **robust and effective**, similar to the robust success of DDQN.
- **Acrobot**: The slightly better performance of A2C/PPO compared to DDQN suggests the PG hyperparameter search was **highly successful** in finding an optimal balance for the Actor-Critic components and advantage estimation.
- **MountainCar**: The **failure** of A2C and PPO indicates the standard hyperparameter tuning was **insufficient** to overcome the extreme sparse reward. Success would have required specific tuning for exploration or reward modification.
- **Pendulum (Continuous)**: The strong performance of **SAC** confirms that its core hyperparameters, especially the **entropy coefficient** (α), were well-tuned, which is crucial for efficient learning in continuous control tasks.