Cairo University

Faculty of Engineering

Computer Engineering Department

CMPS458 Reinforcement Learning Report

Team Name/Number:
Aisha Tawfik
Fatma Gamal
Mohab Yasser

Supervisor: Ayman AboElhassan

December 3, 2025

Deliverables

Repo link: https://github.com/MohabYasser2/RL_Assignment3

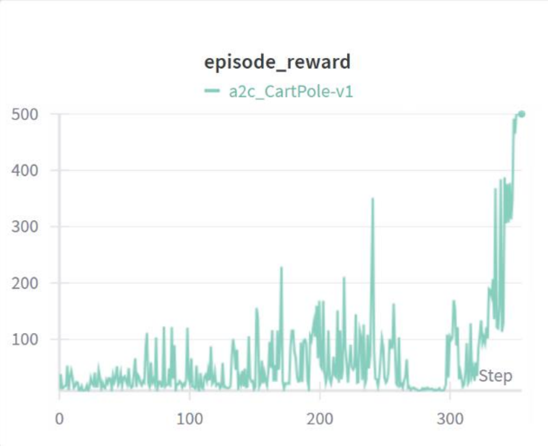
Discussion

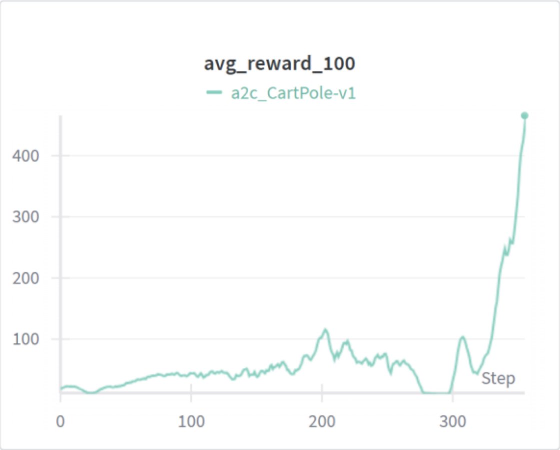
1. For each of the classical environments:

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

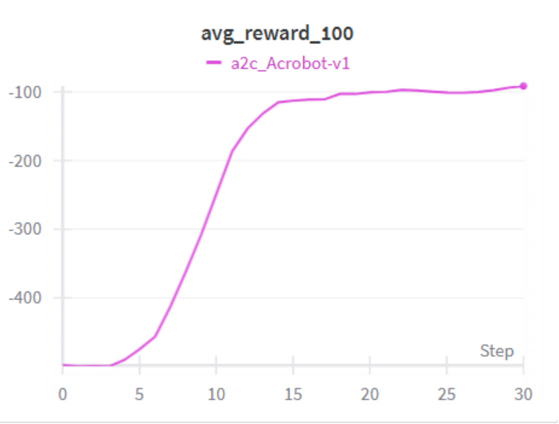
A2C

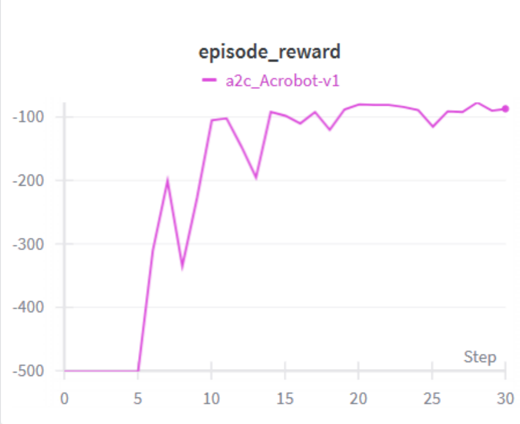
Cartpole



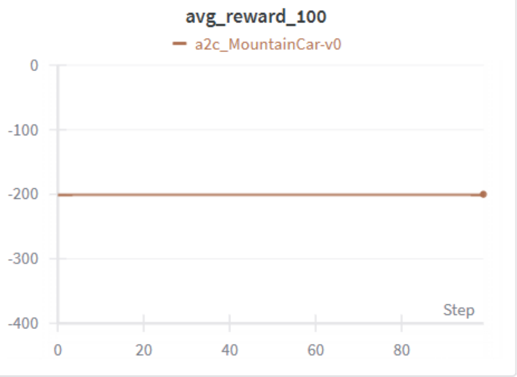


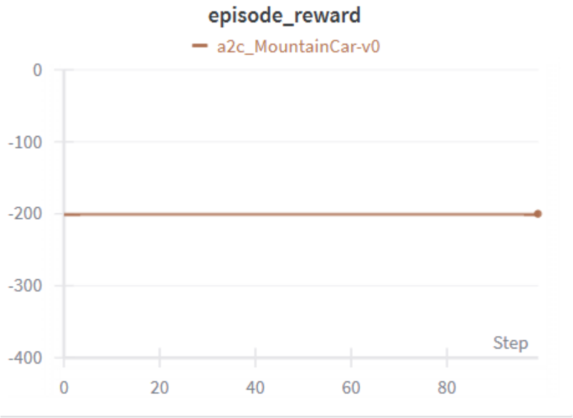
Acrobot



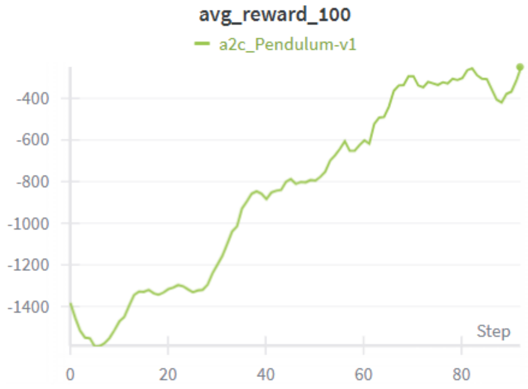


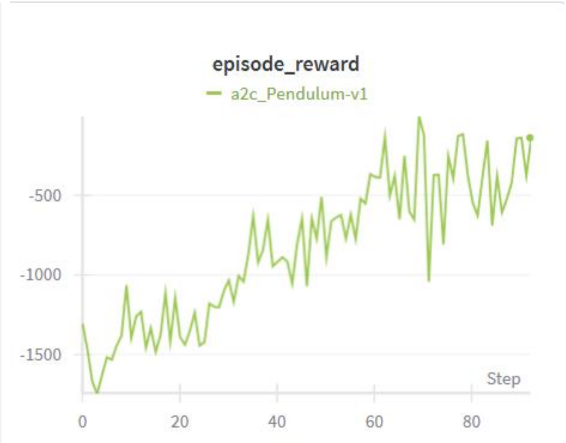
Mountaincar



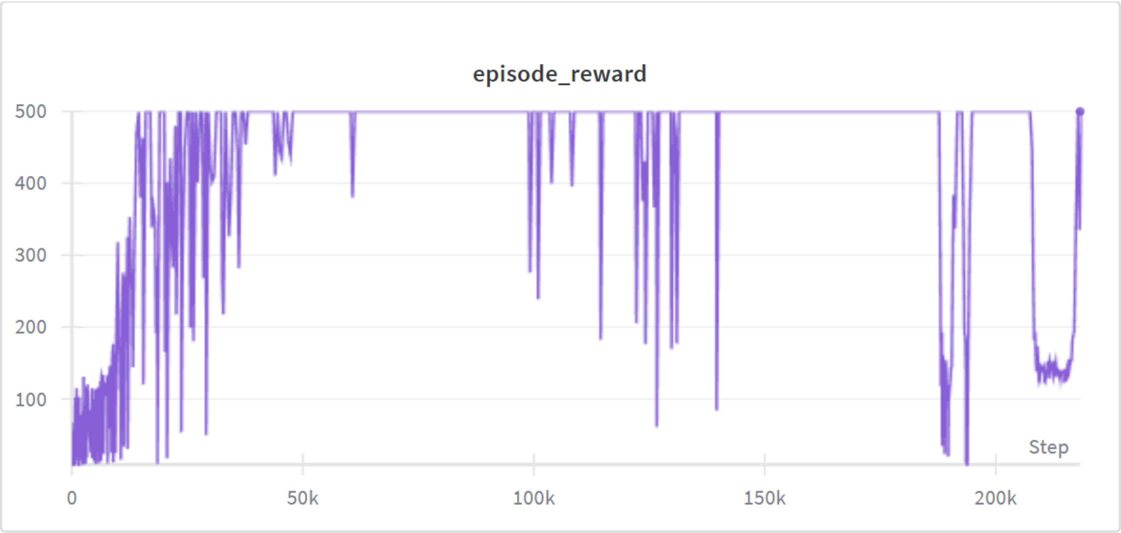


Pendulum

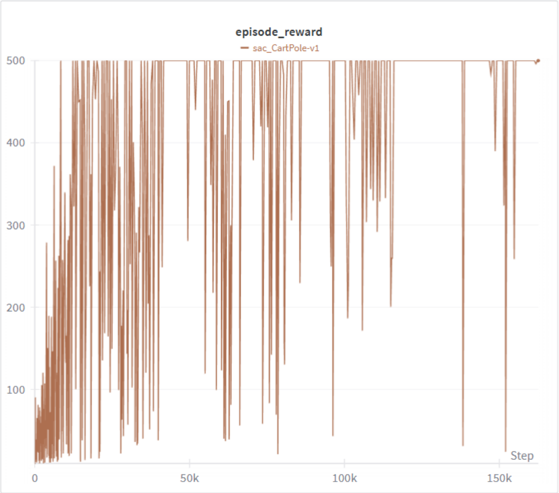


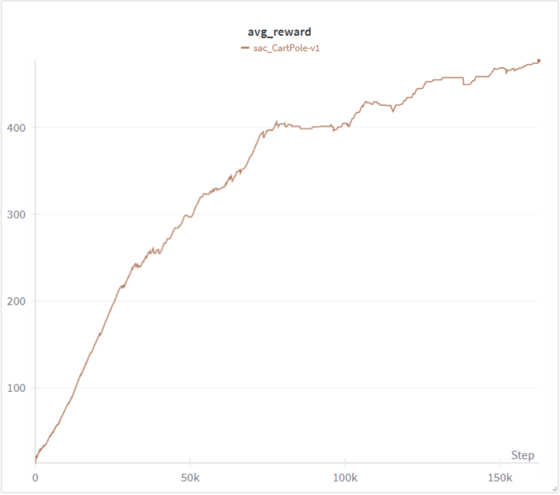


SAC:
CartPole

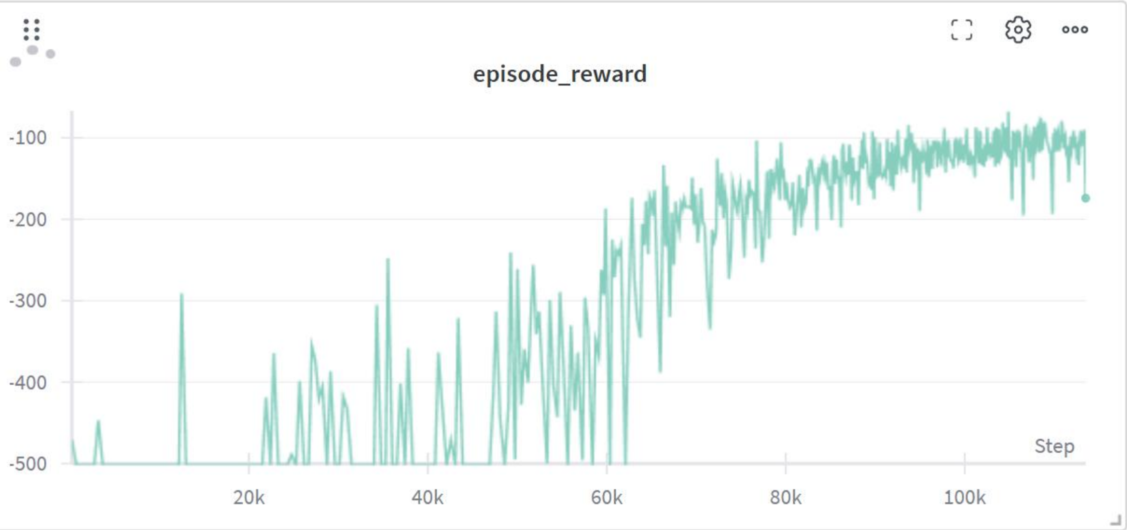


After convergence threshold set to -475:

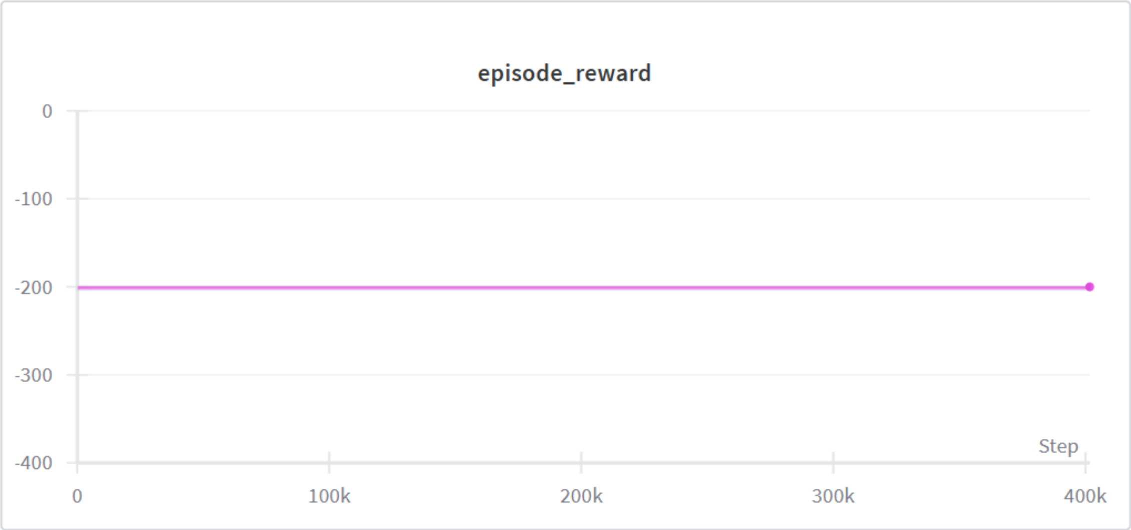




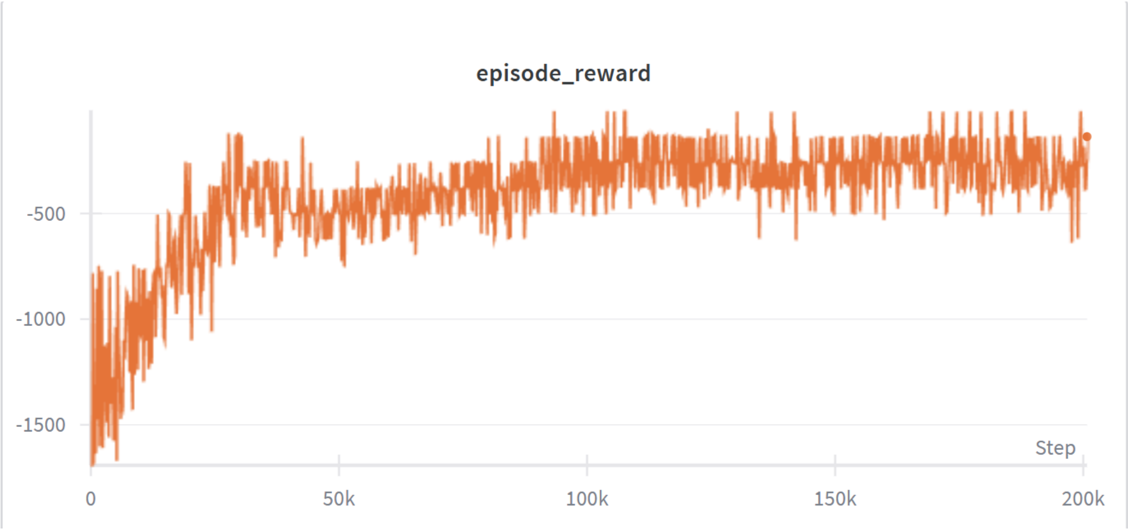
Acrobot



Mountaincar

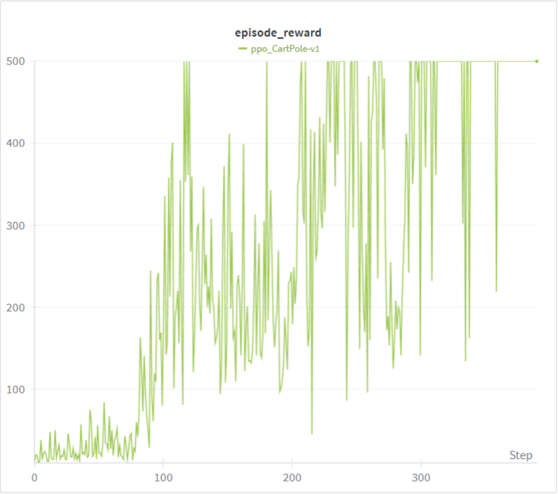


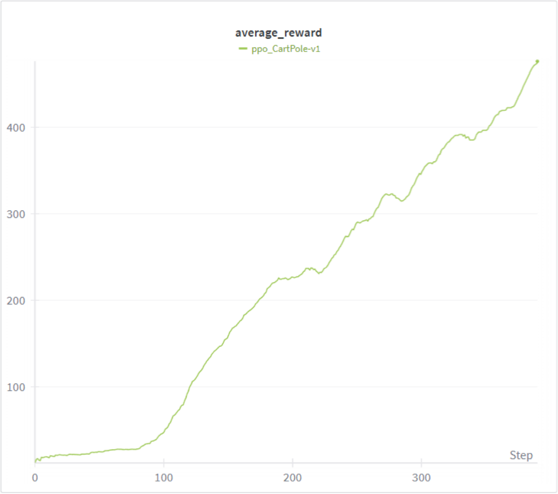
Pendulum



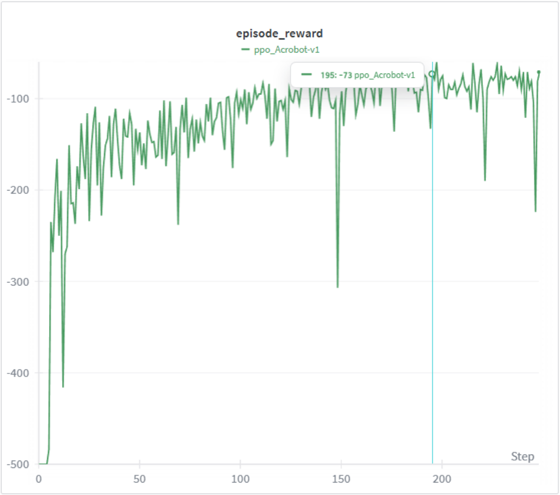
PPO:

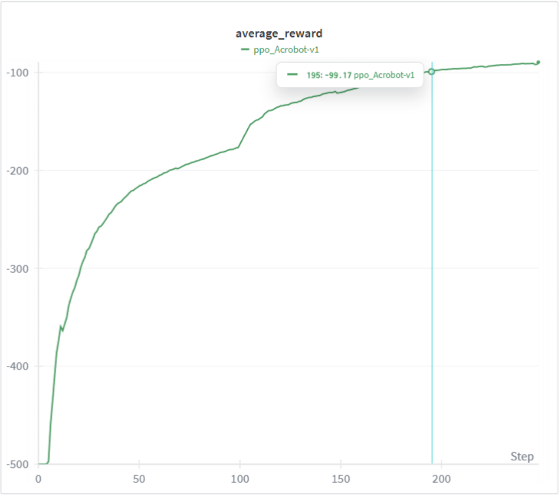
Cartpole:



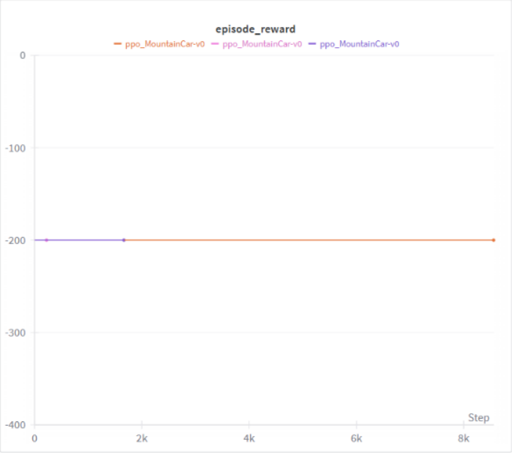


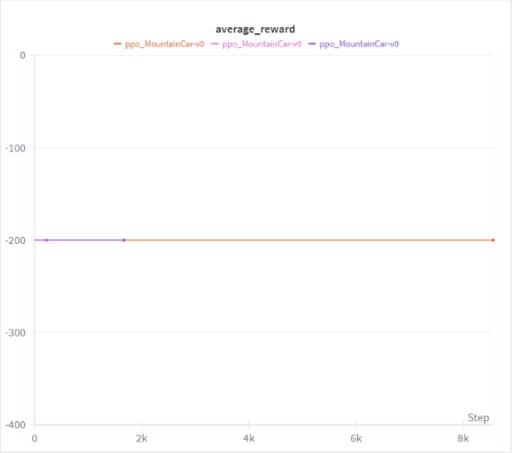
Acrobat:



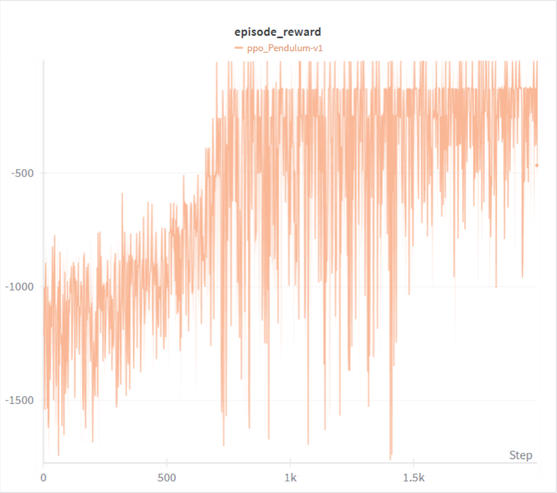


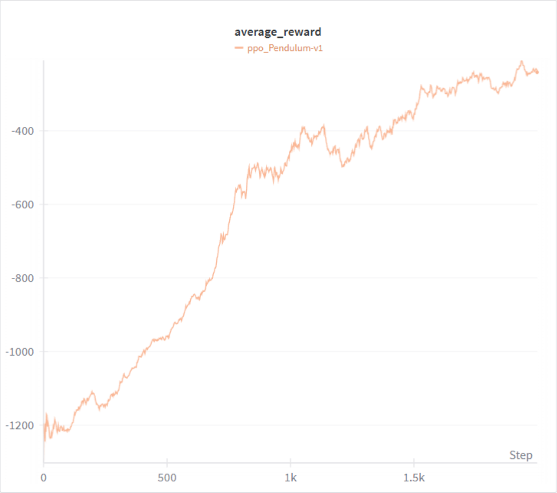
Mountain Car:





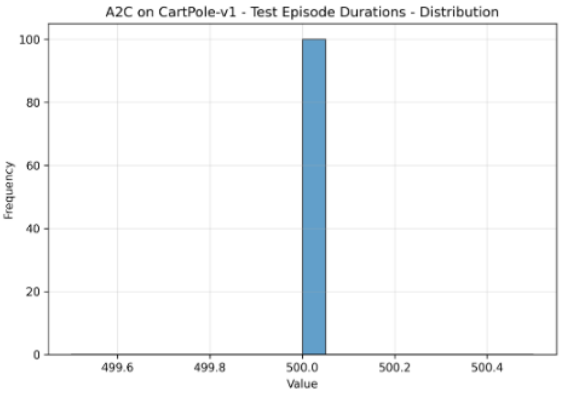
Pendulum:

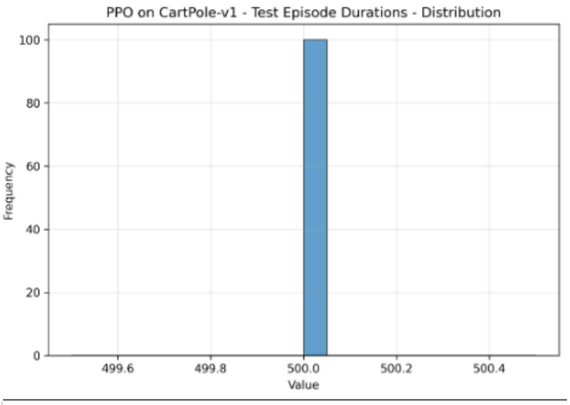


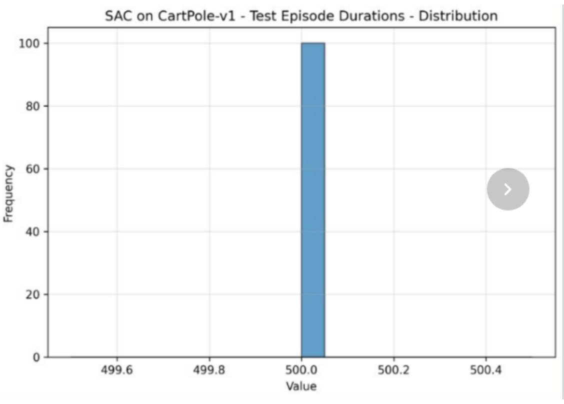


1.2. How stable are the trained agents? Show with test episode duration figures.

Cartpole

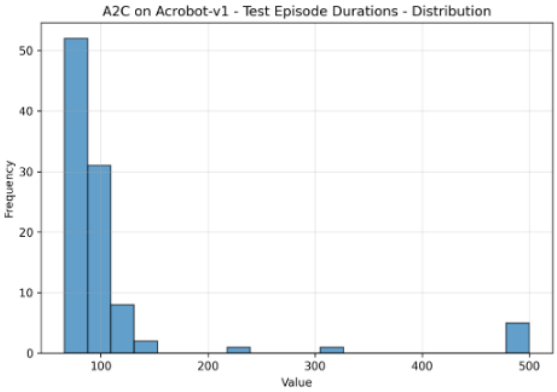


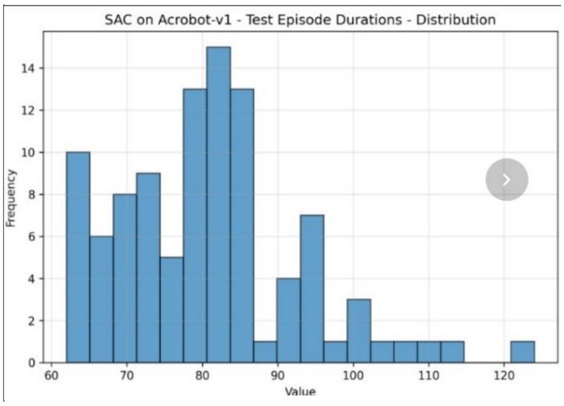


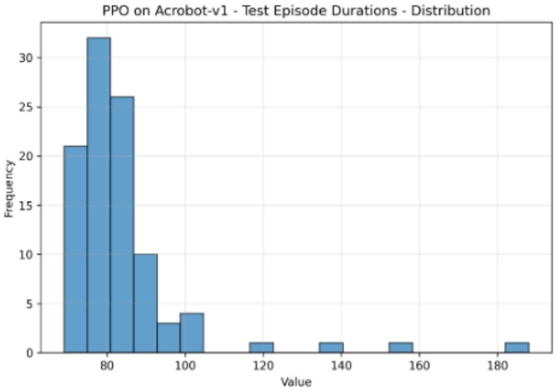


All three agents (A2C, PPO, SAC) are fully stable on CartPole: every test episode lasts the full 500 steps with zero variance, showing perfectly consistent and reliable performance.

Acrobot

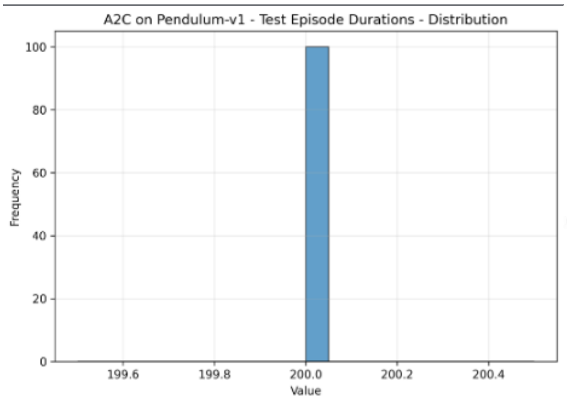

A2C on Acrobot-v1 - Test Episode Durations - Distribution

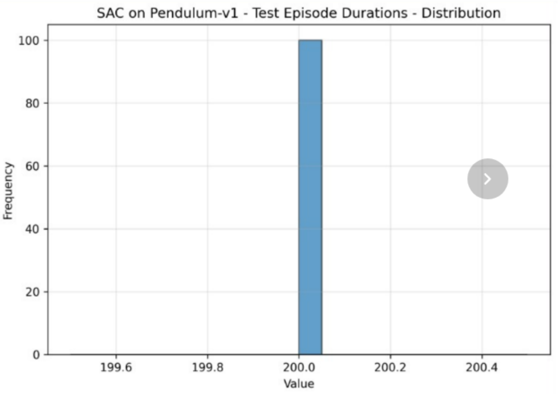

SAC on Acrobot-v1 - Test Episode Durations - Distribution

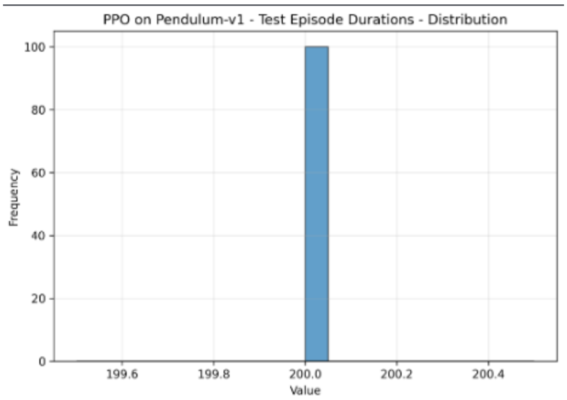

PPO on Acrobot-v1 - Test Episode Durations - Distribution

Across all algorithms, Acrobot shows much higher variability than CartPole. A2C is the least stable, producing a wide spread of episode durations and several large outliers. PPO is more consistent, with durations clustered around ~80–100 steps and fewer extreme values. SAC is the most stable of the three, with tighter distributions and smaller variance. Overall, all agents exhibit moderate stability, but none reach the perfect consistency seen in CartPole.

Pendulum







All three agents (A2C, PPO, and SAC) show perfect stability on Pendulum: every test episode lasts exactly 200 steps with zero variance. This indicates that although the reward performance differs across algorithms, their episode durations during testing are completely stable and consistent.

1.3. Explain from your point of view how well-suited Policy Gradient is to solve this problem.

CartPole-v1
CartPole is very easy for policy-gradient methods. Both A2C and PPO learn quickly due to dense rewards and simple dynamics. SAC is unnecessary because the action space is discrete. This environment is the most suitable for PG algorithms.

Acrobot-v1
Acrobot is harder because the reward is sparse, but A2C and especially PPO still learn effectively thanks to value-function bootstrapping. Policy-gradient methods work reasonably well, though slower than in CartPole.

MountainCar-v0
MountainCar is poorly suited for policy-gradient algorithms. Because rewards are extremely sparse and the optimal behavior requires counter-intuitive exploration, A2C and PPO often fail or learn very slowly. Value-based methods (like Q-learning/DQN) work much better. PG is the least effective here.

Can be Solved with changing reward system to be dependent on the height or closeness to target.

Pendulum-v1
Pendulum is a continuous-control task where basic A2C performs weakly, PPO performs moderately well, and SAC performs best. SAC’s entropy regularization makes it the ideal choice. Discrete Pendulum is especially difficult for A2C unless using separate actor/critic networks.

2. Compare Policy Gradient results to DDQN results from the previous Assignment.

Environment	A2C Avg Return	PPO Avg Return	SAC Avg Return	Best PG Model	DDQN Avg Return	PG vs DDQN
CartPole	500	500	500	All equal (tie)	500	Equal – both methods solve CartPole perfectly and reach maximum return.
Acrobot	−102.28	−82.73	−80.66	SAC	−81.36	SAC ≈ DDQN, but PG overall slightly worse because A2C struggles; PPO decent; SAC similar to DDQN.
MountainCar	-200	-200	-200	Both (tie)	−139.91	DDQN clearly better – PG fails to discover optimal pushing strategy due to sparse reward.
Pendulum	−157.16	−175.24	−158.72	A2C	−154.65	DDQN slightly better numerically, but PG is more stable and appropriate for continuous control.

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

The hyperparameter tuning results only **partially matched** those of the previous DDQN assignment. Some trends—such as using a **high discount factor (γ ≈ 0.99)** and **small learning rates**—remained consistent across both value-based and policy-based methods. These similarities show that long-term reward propagation and stable learning updates are universally important.

However, most other hyperparameter behaviors diverged noticeably because A2C, PPO, and SAC use **policy-gradient** and **actor–critic** mechanisms that differ fundamentally from DDQN. Replay memory, which was essential in DDQN, was either **much larger** (in SAC) or **not used at all** (in A2C/PPO). Batch size requirements also changed: DDQN performed best with small batches, while SAC required **large batches**, and A2C/PPO relied more on frequent rollout updates than on batch size. Exploration methods also shifted from **ε-greedy** to **entropy-based exploration**, which is necessary for stochastic policy optimization.

Overall, only γ and learning rate transferred well from the previous assignment. Most other hyperparameters changed because policy-gradient algorithms optimize distributions, not Q-values, and therefore require different stability mechanisms. This shows that good hyperparameters are **algorithm-specific**, and results from DDQN cannot be directly reused for actor-critic methods.