Actor-Critic Methods on Classic Gym Environments

By Erin Gregoire & John Abramo

Spring 2025

Actor-critic methods are a combination of policy gradient and value-based functions, where the actor and critic each demonstrate one of these aspects. The role of the actor network is to update the policy parameters and choose the action based on the policy. This happens by the actor receiving the state and then creating a distribution of probabilities for each action. The action with the highest probability is said to be the optimal action for that state. Meanwhile, the role of the critic is to evaluate the action that the actor just took by using value-based methods including the true value function, Q-value, and advantage value.

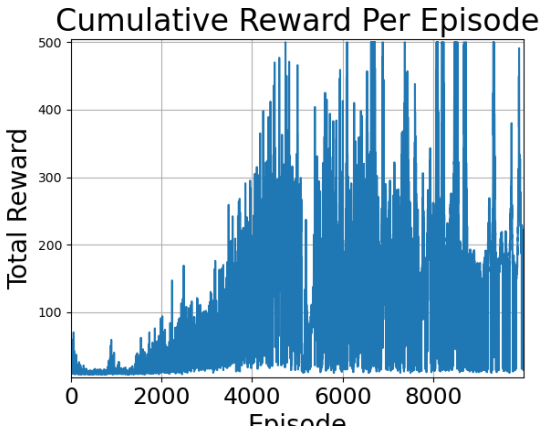
With Advantage Actor Critic (A2C), the critic uses the advantage value. The advantage value comes from the value output of the critic's deep network subtracted from the Q-value that is computed with the state-action's reward, as well as the discount factor. The importance of calculating the advantage value in this way is that it significantly reduces the variance of the policy gradient.

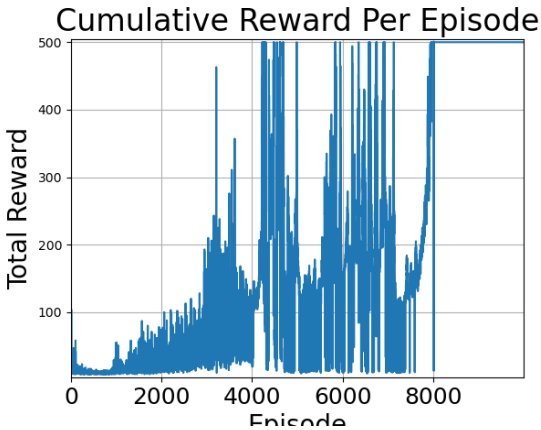
Although in our construction, the actor and critic share the same network, they both have their own loss function, which is then combined for the total loss. For the critic, the loss function is the mean squared error which is the standard loss function used when working with value-based functions. For the actor, the loss is computed as the average of the log of the action probability distribution multiplied by the advantage value.

Cart Pole:

The environment used in part two is the cart pole environment. This environment features an agent that has two possible actions, to push the cart to the left or to the right. This affects the angle and velocity of the pole by moving it slightly in either direction. The observation space consists of the cart's position and velocity, as well as the pole's angle and angular velocity. Together, the observation space and the action make up the environment's state. The episode terminates if the pole angle is negatively or positively 12 degrees, meaning that the pole has tipped over to the right or left, or the cart position is reaching the edge of the render's display so that the cart does not move completely out of bounds. The episode truncates after 500 timesteps have been reached. The reward structure is very simple and includes a reward of one for every action that the agent makes, including the termination state. This means that the maximum reward that the agent can receive in a single episode is 500. The environment is considered to be solved if the agent reaches a reward of at least 470 over 100 episodes.

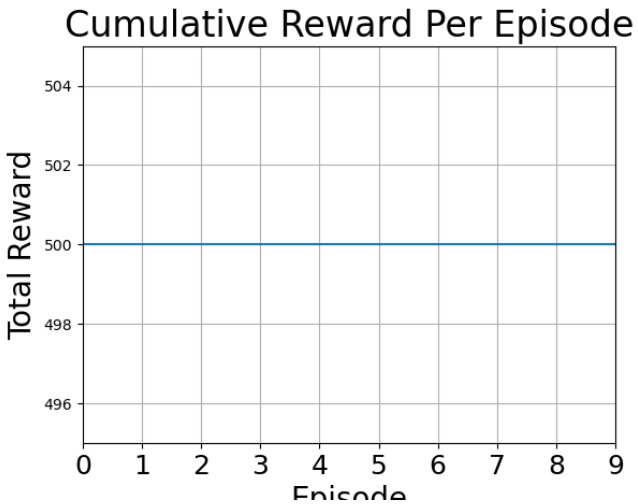
Training was conducted over 1,000 episodes and included multi-threading where there are two agents simultaneously training on the Cart Pole environment. The results of training can be seen in the graphs here depicting the cumulative reward for actor-critic agent per episode. Each graph relates to each of the actor-critic threads.





These results show that both agents are able to reach the maximum reward over the course of the training episodes, however, only one agent fully converges and ends training with a perfect score each episode.

To evaluate the advantage actor critic model, the algorithm was fed the learned policy and chose only greedy options. Greedy options are chosen by selecting the action with the highest probability from the probability distribution given by the actor. The evaluation was conducted for ten episodes with 10,000 steps per episode. The cumulative reward per episode can be seen to the right. Based on this graph, the A2C agent was able to learn the best parameters and is able to solve the Cart Pole environment during every evaluation episode. Despite the fact that only one of the actor-critic threads converged, the combination of the weights learned between both agents, created results that result in perfect evaluation.

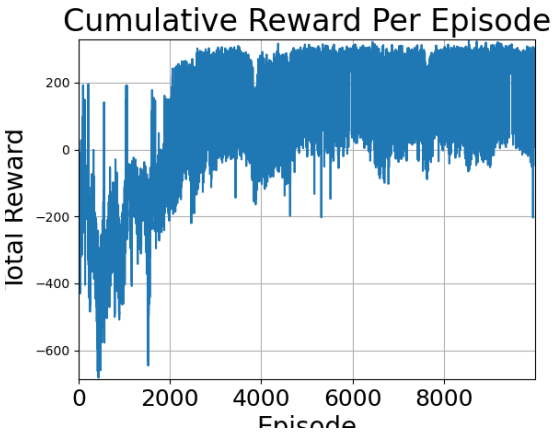


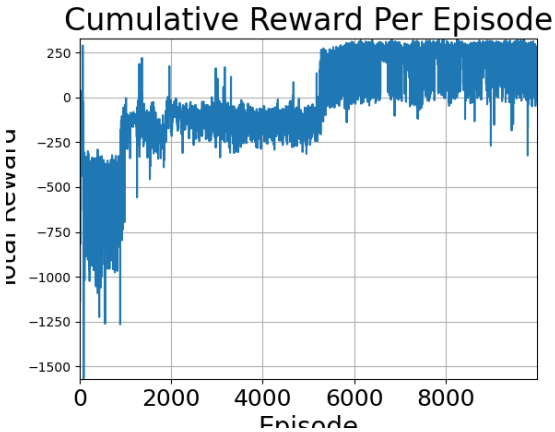
Lunar Lander:

The lunar lander environment demonstrates a rocket that must land within the landing pad that is denoted by two flag poles. The lunar lander environment consists of four discrete actions, to fire the right or left engine, to fire the main engine, or to do nothing. The observation space consists of five main elements that form an eight-dimensional vector. These include the coordinates of the rocket, the linear velocities, the angle position of the rocket, the angle's velocity, and whether or not the legs of the craft are touching the ground. The reward structure is a bit complex and I believe it made the lunar lander learn rather well. Negative rewards are provided if the rocket moves away from the landing pad or crashes. Positive rewards are given if the craft begins landing by touching one of its legs to the ground, or fully lands. However, with every action that fires one of the engines, the lunar lander loses a small amount of reward to

assist with timeliness and prevent unnecessary extra movement. The environment is considered solved if the agent earns a score of 200 points. The episode terminates if the lunar lander crashes into the moon, the craft moves out of the frame, or the rocket is considered to be not awake.

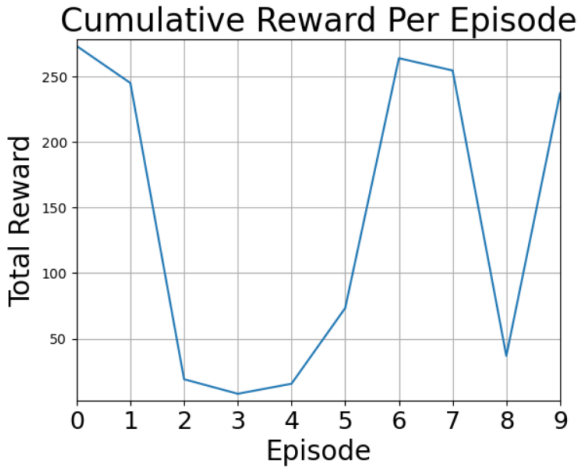
Training consisted of two multi-processing threads conducted for 1,000 episodes with 10,000 steps each episode. The graphs below show the cumulative reward per episode earned by each thread during the training episodes.





These results show that the A2C agent was able to learn the maximum reward many times during the training episodes. These graphs also show a strong sense of convergence. The main challenge though, is that even though the agent is regularly reaching well above 200 points during training, the convergence still leaves variance between the reward scores of approximately 0 to 250. True convergence of solving the Lunar Lander environment would ideally be a score of 200 or more during each episode.

To evaluate the A2C agent on the Lunar Lander environment, the agent was run for ten episodes, where it chose only the greedy options, which is the highest probability action from the action distribution created by the actor. The cumulative reward per episode graph can be seen here. As predicted based on the training results, the agent has not fully learned how to solve the Lunar Lander environment. The agent is able to get positive rewards during each episode and received a score over 200 multiple times. However, its inability to receive more than 200 points across all ten evaluation episodes shows that the optimal policy was not yet learned by the advantage actor-critic agent.



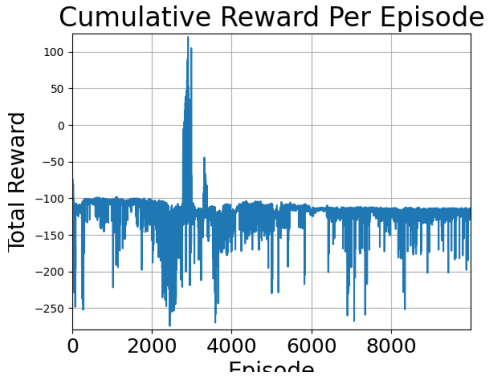
Bipedal Walker:

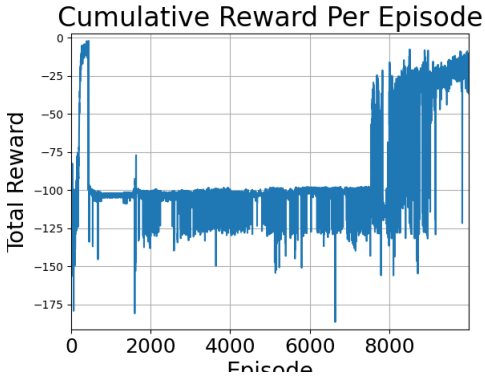
The Bipedal Walker environment provides a robot agent that is attempting to walk with two leg-like attachments. The environment provides a continuous action space consisting of 4 values in the range [-1, 1] that each correspond to a motor speed at the hip and knee for each leg. The observation space consists of hull angle speed, angular velocity, horizontal speed, vertical speed, position of joints and joints angular speed, legs contact with ground, and 10 lidar rangefinder measurements.

The reward for moving the robot to the far end of the environment is +300 points. If the robot falls down, it receives -100 points. The robot also receives a small negative reward for each torque amount applied to the motors. This is to ensure that the robot is running efficiently and making progress toward the goal state.

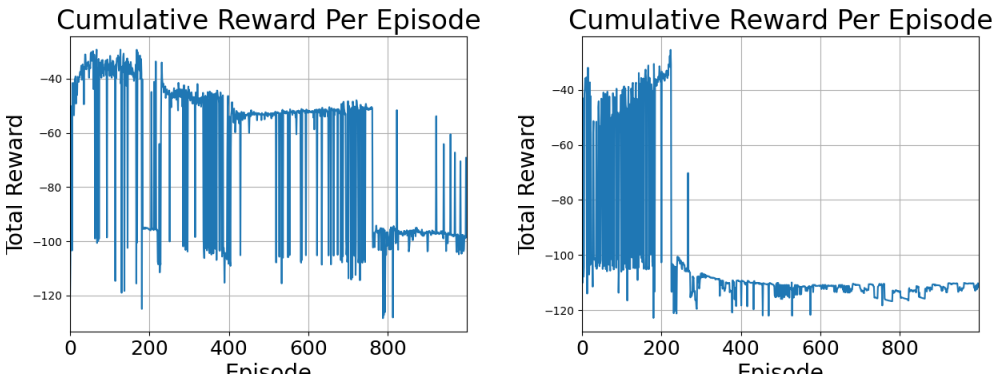
The robot starts in the same position every time and terminates once it reaches the goal state at the far end of the environment or touches the ground.

We tried multiple different methods for training the model. In our first attempt, we trained over 10,000 episodes on two threads. This led to a model that converged to a reward of about -25. There were a couple of spikes during training where the agent achieved a number of somewhat good episodes in a row, getting about 100 points per episode.

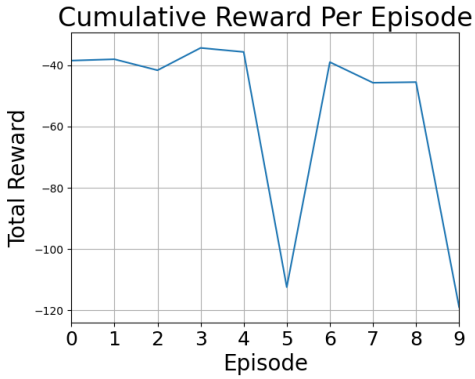




We made some changes based on advice given in office hours to improve the results. Based on that we ran two threads for 1,000 episodes with a slightly changed model and received the updated results. These results were still similar to the previous results in that they seemed to converge to a negative reward.



For our evaluation, we ran the agent for 10 episodes. The results are shown below. Ultimately the agent did not solve the environment like we had hoped it would instead, it seems to have made some progress toward making better decisions. On the evaluation, the most common outcome was for the agent to get -40 points. This is significantly better than that -100 average number of points that the agent got before training, but still short of the intended 300 points that would be necessary for an optimal agent to have solved the environment.



References:

"ActorCriticOperator." *PyTorch, 2022.* Accessed 13 Apr 2025. https://pytorch.org/rl/main/reference/generated/torchrl.modules.tensordict_module.ActorCriticOperator.html

"Cart Pole." *Gym Documentation.* https://www.gymlibrary.dev/environments/classic_control/cart_pole/. Accessed 23 Mar 2025.

Dixitaniket. "Advantage Actor-Critic (A2C) Algorithm Explained and Implemented in PyTorch." *Medium,* 16 July 2024. Accessed 16 Apr 2025. https://medium.com/@dixitaniket76/advantage-actor-critic-a2c-algorithm-explained-and-implemented-in-pytorch-dc3354b60b50

Gupta, Mehul. "Advantage Actor-Critic (A2C) algorithm in Reinforcement Learning with Codes and Examples using OpenAI Gym." *Medium,* 14 Apr 2023. Accessed 16 Apr 2025. https://medium.com/data-science-in-your-pocket/advantage-actor-critic-a2c-algorithm-in-reinforcement-learning-with-codes-and-examples-using-e810273c0c9e

Klimov, Oleg. "Lunar Lander." *Gym Documentation.* https://www.gymlibrary.dev/environments/box2d/lunar_lander/. Accessed 24 Mar 2025.

"Multiprocessing package - torch.multiprocessing." *PyTorch,* 2024. Accessed 15 Apr 2025. https://pytorch.org/docs/stable/multiprocessing.html