<u>Reinforcement Learning Report-3</u>

Aniket Kumar
Kumar Satyam

<u>PART 2:</u>

<u>2. Environment Description: CartPole-v1 (Gymnasium)</u>

<u>What are the roles of the actor and the critic networks?</u>

In the notebook,we have implemented a single ActorCritic class that contains both:

- **Actor Network** (policy layer):
  - Outputs a probability distribution over possible actions using a softmax layer.
  - Responsible for **action selection** during interaction with the environment.
- **Critic Network** (value layer):
  - Outputs a scalar value estimating the **expected return** from the current state.
  - Used to **evaluate how good** the current state is and guide the actor's learning.

  This shared architecture allows the model to reuse feature representations from the input state for both policy and value predictions.

<u>What is the "advantage" function in A2C, and why is it important?</u>

Purpose of the Advantage:

- Measures how much better or worse an action is compared to the critic’s baseline.
- Reduces variance in policy gradient updates.
- Ensures only better-than-expected actions are reinforced.

Stabilizes Policy Learning:
Advantage values remove noise from raw returns, allowing the policy to converge more reliably.

Balances Exploration and Exploitation:
Combined with entropy regularization, it helps the agent explore meaningfully without clinging to early suboptimal strategies.

<u>Describe the loss functions used for training the actor and the critic in A2C.</u>

Actor Loss (Policy Gradient):

- Encourages the actor to assign higher probabilities to actions that yield positive advantages.
- logp is the log-probability of the action taken, and it's scaled by the advantage.
- .detach() ensures that the advantage is treated as a constant during backpropagation.

Critic Loss (Value Function):

- A Mean Squared Error (MSE) between the predicted value and the actual return.
- This trains the critic to reduce error in state-value estimates.

Environment Overview:
The CartPole-v1 environment tasks an agent with balancing a pole on a cart that moves along a frictionless 1D track. The agent applies forces to the left or right to keep the pole upright.

Observation Space (4-D vector):

- Cart Position (x)
- Cart Velocity ($\dot{x}$)
- Pole Angle (θ)
- Pole Angular Velocity ($\dot{\theta}$)

Action Space (Discrete: 2 actions):

- `0`: Push cart to the left
- `1`: Push cart to the right

Reward Function:

- +1 reward is given at every timestep the pole remains upright.
- Episode ends if:
  - Pole angle exceeds ±12°
  - Cart moves beyond ±2.4 units from the center
  - Episode exceeds 500 steps (max episode length)

Goal:
Maximize cumulative reward (up to 500). A reward of 475+ typically indicates the task is solved.

2. Training Results and Analysis

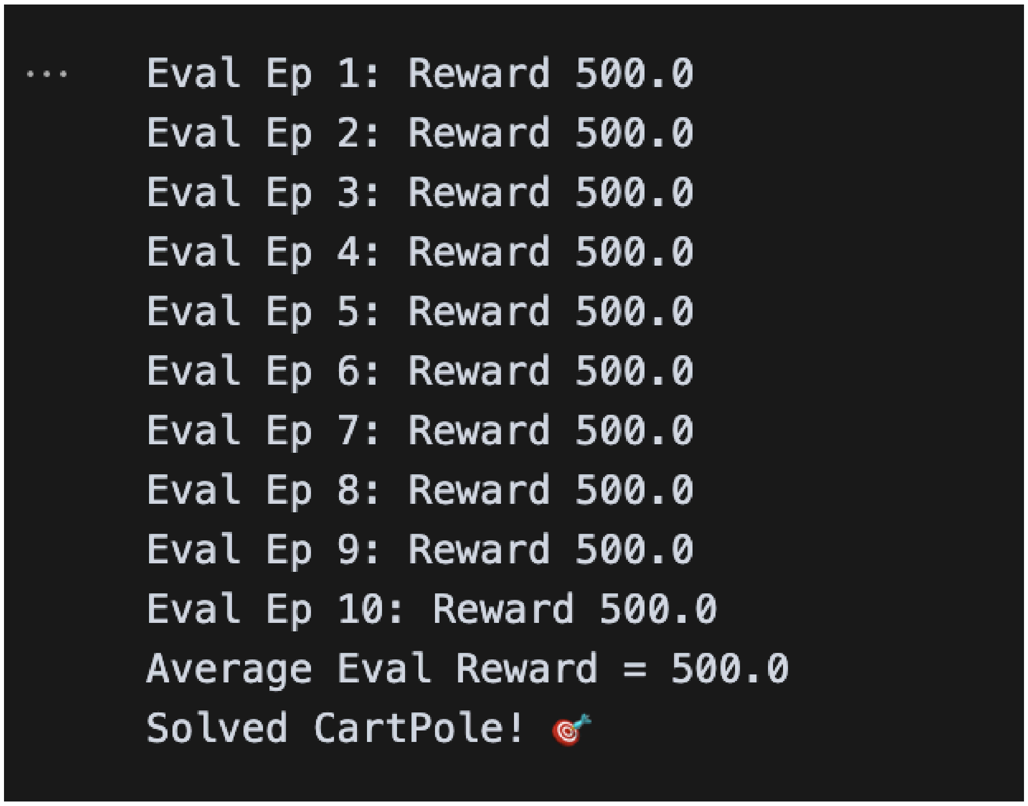
The A2C agent was trained on the CartPole-v1 environment for **2000 episodes** using **2 worker threads** and synchronized updates every 20 steps. The actor-critic model comprises a shared hidden layer feeding into separate policy (actor) and value (critic) heads.

Training Parameters:

- Learning Rate: `1e-3`
- Gamma (Discount Factor): `0.99`
- Entropy Coefficient: `0.01`
- Hidden Layer Size: `256`

Analysis:

- In the first 200–300 episodes, rewards were low (50–150) as the agent explored randomly.
- The model began to stabilize after ~500 episodes.
- By episode ~1500, the agent regularly achieved near-optimal rewards (400–500), successfully learning to balance the pole.
- The entropy term helped promote exploration in early training.



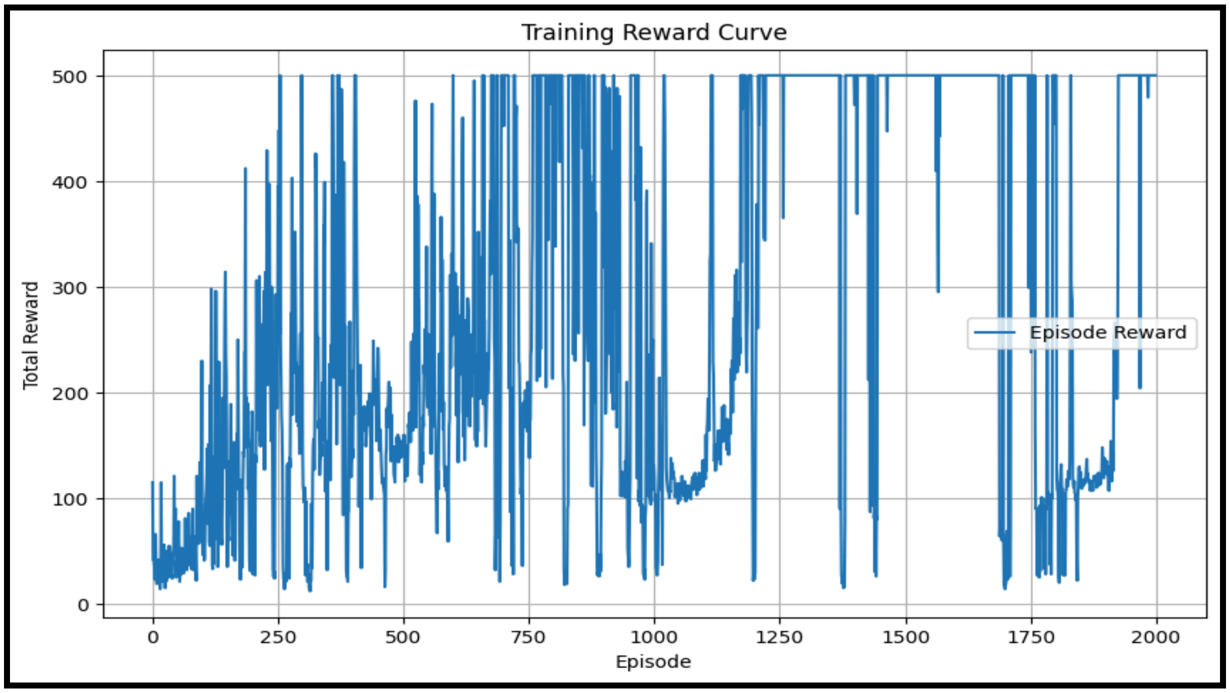
```
...   Eval Ep 1: Reward 500.0
      Eval Ep 2: Reward 500.0
      Eval Ep 3: Reward 500.0
      Eval Ep 4: Reward 500.0
      Eval Ep 5: Reward 500.0
      Eval Ep 6: Reward 500.0
      Eval Ep 7: Reward 500.0
      Eval Ep 8: Reward 500.0
      Eval Ep 9: Reward 500.0
      Eval Ep 10: Reward 500.0
      Average Eval Reward = 500.0
      Solved CartPole! 🎯
```

Model Behavior:

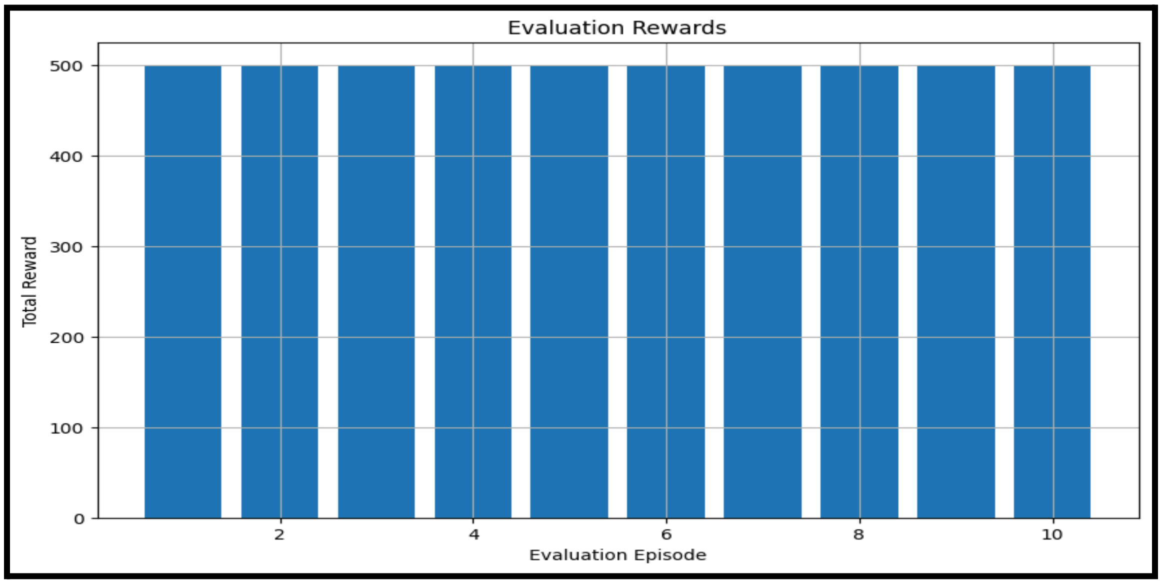
- Smooth balancing without abrupt actions
- Quick reaction to pole deviations
- Reduced oscillation in the cart's movement

3. Evaluation Results

Post-training, the model was evaluated over **10 episodes** using greedy (deterministic) actions, i.e., selecting the most probable action predicted by the policy network.



Evaluation Rewards:



Observations:

- The agent consistently keeps the pole balanced for nearly the full 500 steps.
- High reward stability indicates robust generalization to varying start states.
- All episodes met the success threshold (>475).

PART 3:

1. Environment Description: LunarLander-v3 (Gymnasium)

Environment:
We used the LunarLander-v3 environment from the Gymnasium library. It simulates a lunar module attempting to land smoothly on a designated landing pad.

State Space:
An 8-dimensional vector containing:

- X, Y position
- X, Y velocity
- Angle
- Angular velocity
- Left and right leg ground contact flags

Action Space:
Discrete with 4 actions:

- 0: Do nothing
- 1: Fire left orientation engine
- 2: Fire main engine
- 3: Fire right orientation engine

Goal:
Land the module safely at the center of the landing pad with minimum fuel usage and without crashing.

Reward Function:

At each timestep, the agent receives a reward based on:

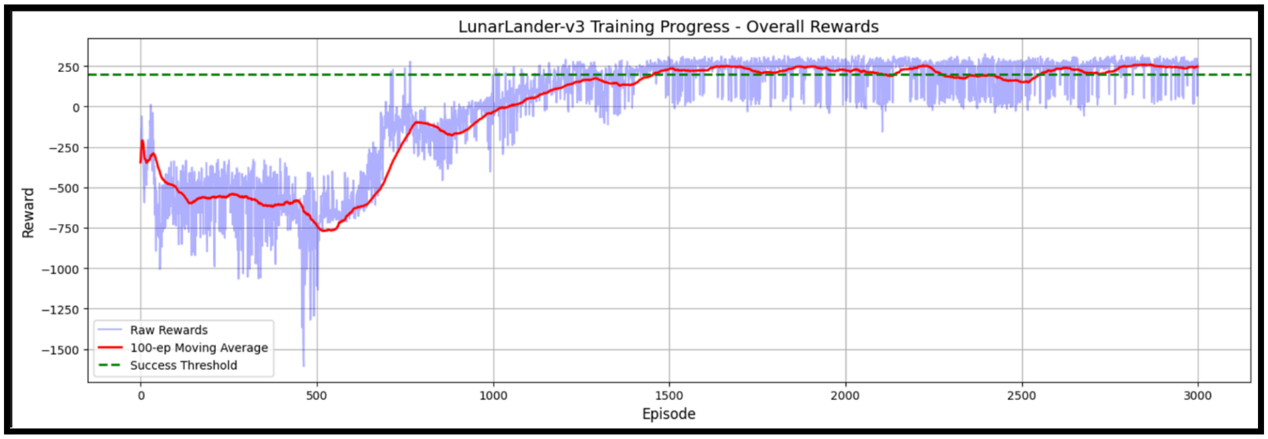
- **Proximity to landing pad:** Closer distance yields higher rewards.
- **Velocity:** Slower speeds are rewarded; higher speeds incur penalties.
- **Angle:** Upright orientation is rewarded; tilting is penalized.
- **Leg contact:** +10 points for each leg in contact with the ground.
- **Engine usage penalties:**
    - -0.03 points per frame for side engines.
    - -0.3 points per frame for the main engine.
- **Terminal rewards:**
    - +100 points for a successful landing.
    - -100 points for crashing.

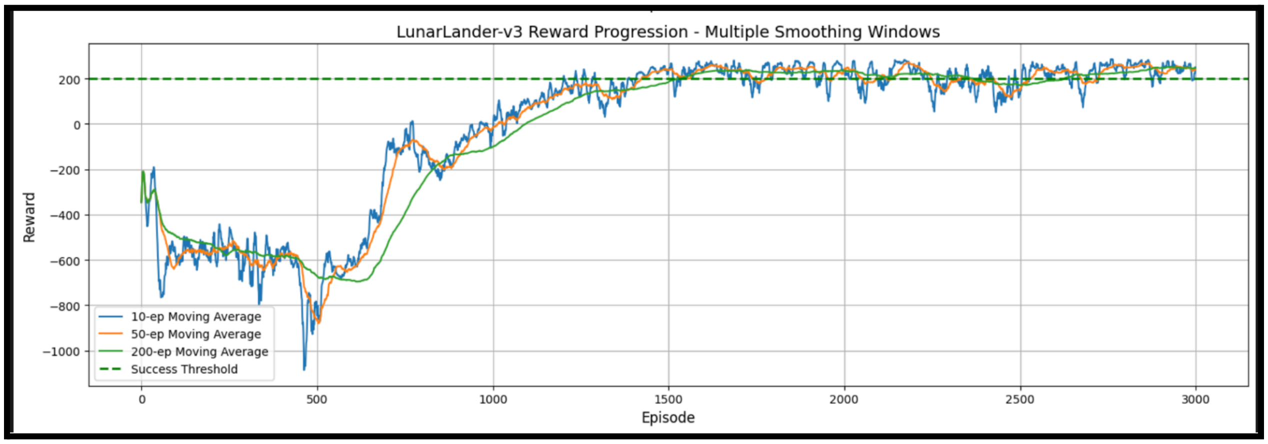
Agent:
The Advantage Actor-Critic (A2C) agent is implemented with two separate neural networks (Actor and

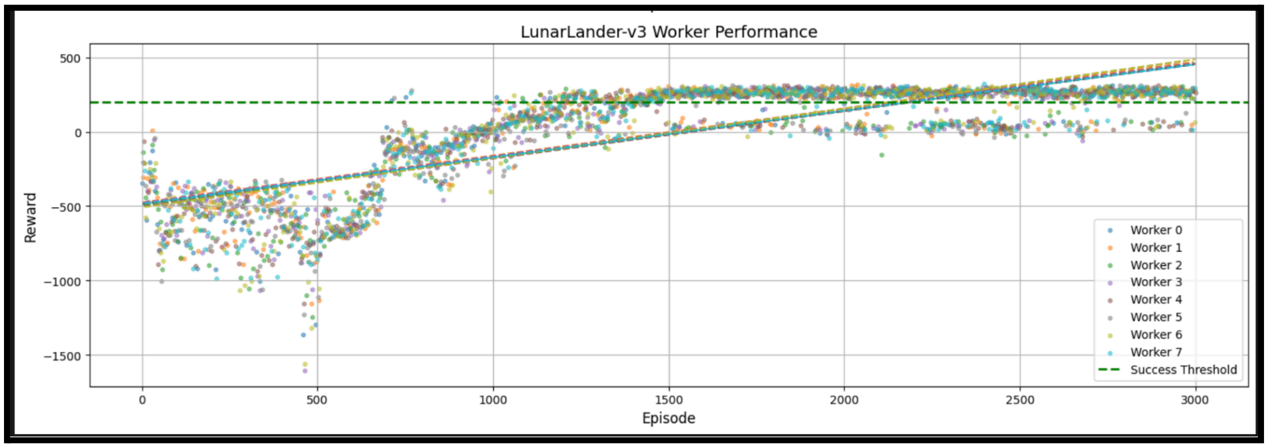
Critic) trained synchronously using shared gradients. The agent receives continuous feedback from the environment and updates its policy based on estimated advantages.

2. Training Results and Analysis

During training, the agent was run for 3000 episodes using multithreaded workers. The figure below shows the **Total Reward per Episode** during training.

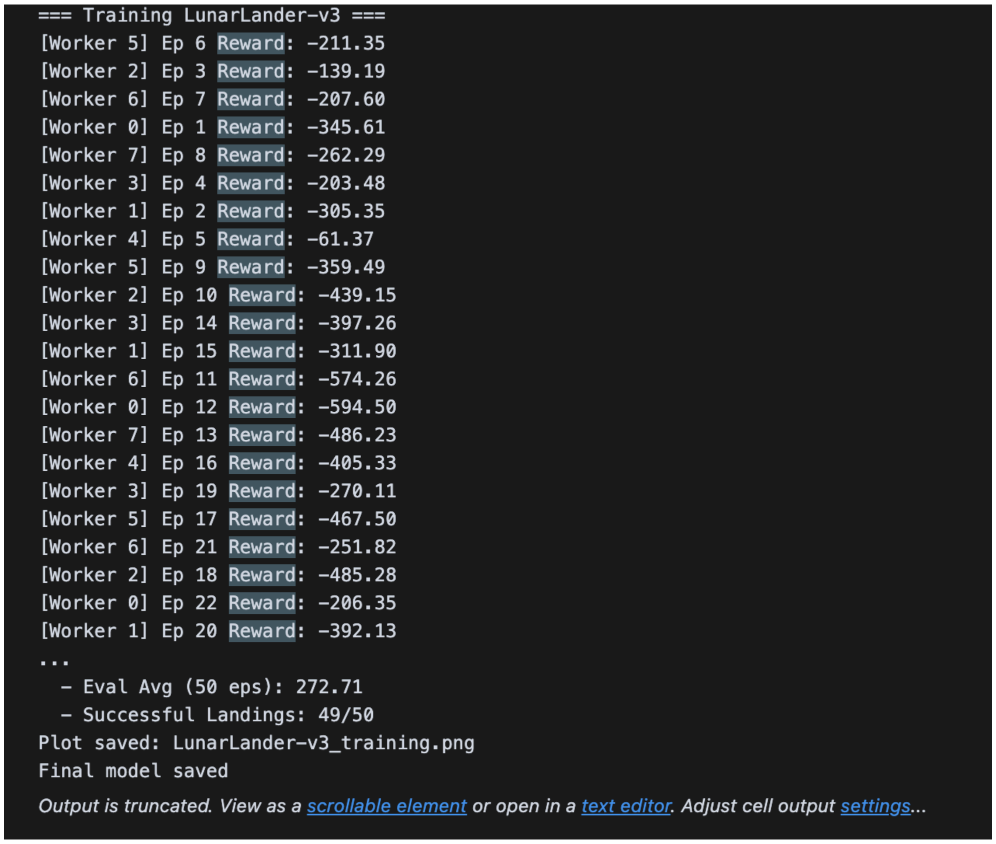






Observations:

- The agent initially explores randomly, resulting in negative rewards.
- With increasing episodes, the average reward trends upward, stabilizing above 200 after ~2000 episodes.
- Occasional dips in reward indicate exploration or unstable policy updates, which is expected in A2C.



```
=== Training LunarLander-v3 ===
[Worker 5] Ep 6 Reward: -211.35
[Worker 2] Ep 3 Reward: -139.19
[Worker 6] Ep 7 Reward: -207.60
[Worker 0] Ep 1 Reward: -345.61
[Worker 7] Ep 8 Reward: -262.29
[Worker 3] Ep 4 Reward: -203.48
[Worker 1] Ep 2 Reward: -305.35
[Worker 4] Ep 5 Reward: -61.37
[Worker 5] Ep 9 Reward: -359.49
[Worker 2] Ep 10 Reward: -439.15
[Worker 3] Ep 14 Reward: -397.26
[Worker 1] Ep 15 Reward: -311.90
[Worker 6] Ep 11 Reward: -574.26
[Worker 0] Ep 12 Reward: -594.50
[Worker 7] Ep 13 Reward: -486.23
[Worker 4] Ep 16 Reward: -405.33
[Worker 3] Ep 19 Reward: -270.11
[Worker 5] Ep 17 Reward: -467.50
[Worker 6] Ep 21 Reward: -251.82
[Worker 2] Ep 18 Reward: -485.28
[Worker 0] Ep 22 Reward: -206.35
[Worker 1] Ep 20 Reward: -392.13
...
  - Eval Avg (50 eps): 272.71
  - Successful Landings: 49/50
Plot saved: LunarLander-v3_training.png
Final model saved
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Comparison to Other Environments: If we compare LunarLander to an Atari-like environment Ping-Pong we observe:

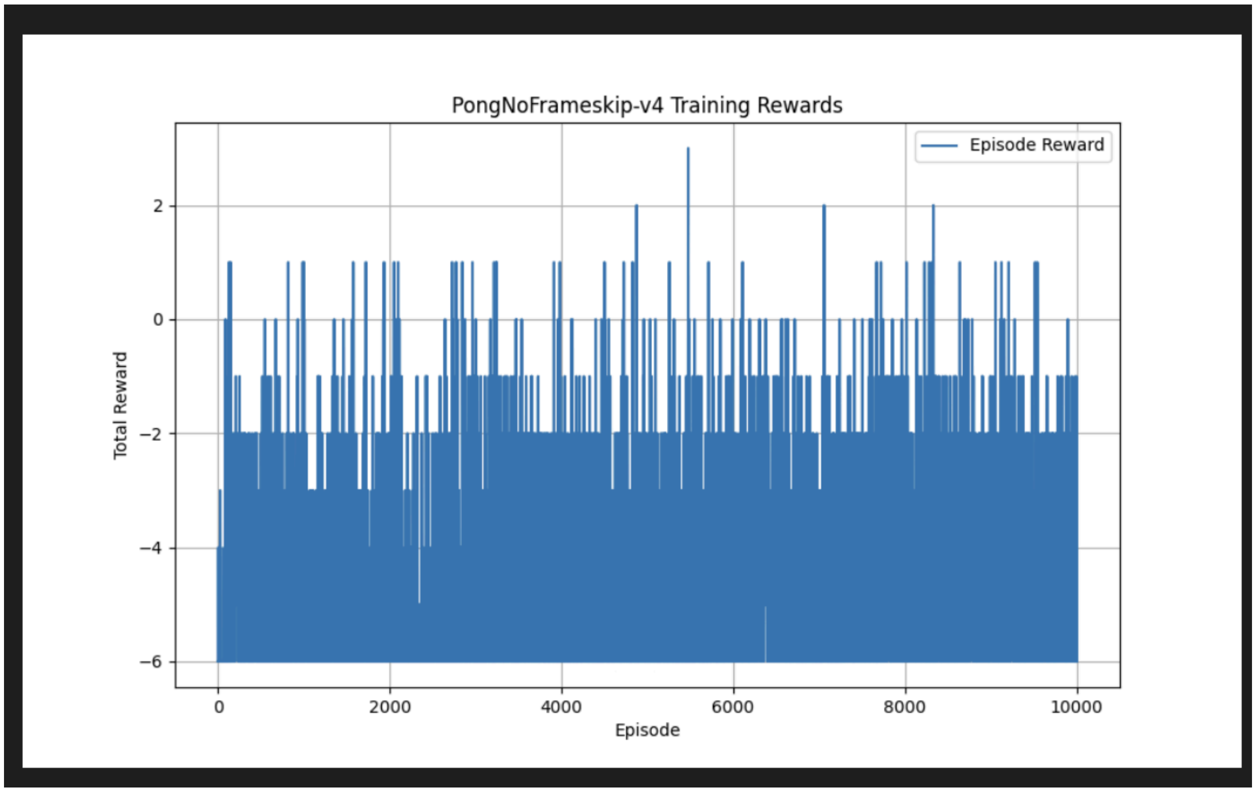
- LunarLander has a smaller, structured state space with physical dynamics.
- Breakout has image-based input and sparse rewards, requiring more exploration and CNNs.
- A2C converges faster on LunarLander due to shaped rewards and simpler dynamics.

2. Environment: PongNoFrameskip-v4

- **State Space**: 4 stacked grayscale image frames of size 84×84. Each frame is preprocessed from raw game images (grayscale + cropped + resized).
- **Action Space**: 6 discrete actions (0-5) — actions like NOOP, FIRE, MOVE_LEFT, MOVE_RIGHT, etc.
- **Agent Goal**: Maximize the game score by learning to return the ball and defeat the opponent.
- **Reward Signal**:
  - **+1**: when the agent wins a point

- **-1**: when the opponent wins a point
- Cumulative reward per episode ranges from -21 to +21

2. Training Results on PongNoFrameskip-v4



Training Curve:

- The plot shows high variance in rewards during training.
- Majority of episodes stay in the **negative reward region** (around -6 to -2).
- The agent exhibits **slow convergence** — common in Pong due to sparse and delayed rewards.
- Occasional spikes indicate learning progress, but not consistently maintained.

Discussion:

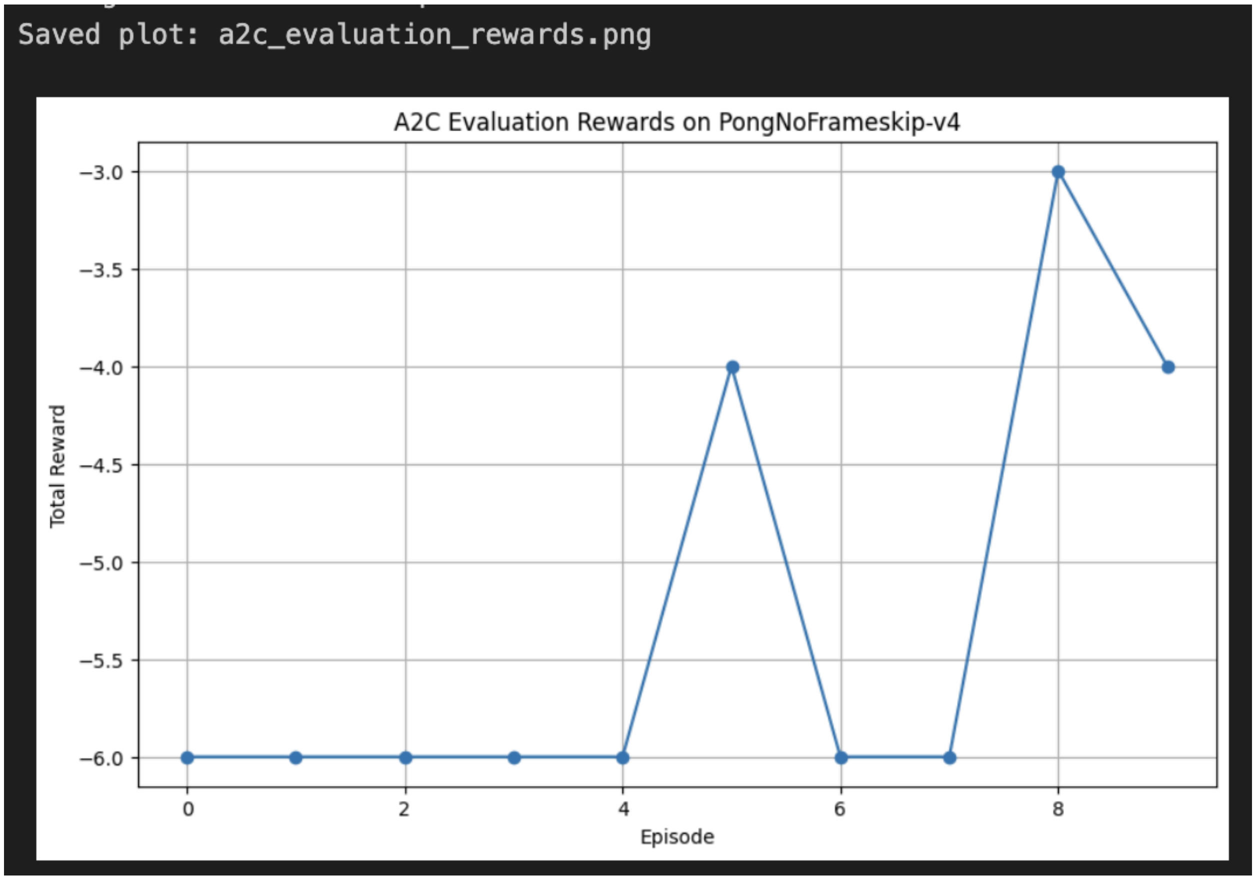
- Despite training with A2C for ~10k episodes, the policy struggles to reach consistently positive rewards.
- Improvements could be made by:
    - Increasing training episodes
    - Using more workers
    - Tuning entropy coefficient to balance exploration

3. Evaluation Results on Pong (Greedy Policy)

Evaluation Plot (10 Episodes):

- Evaluated using **greedy actions** from the learned policy (i.e., sampling the max probability action).
- Rewards mostly hover around **-6**, with some episodes slightly better (~-3).

- This suggests **limited generalization** and the policy likely overfits to sub-optimal trajectories during training.



```
Saved plot: a2c_evaluation_rewards.png
```

- Applied the frame stack ,used cnn , grayscaled the image and normalize[0,1]

Bonus Advanced Actor-critic:

1. CartPole-v1

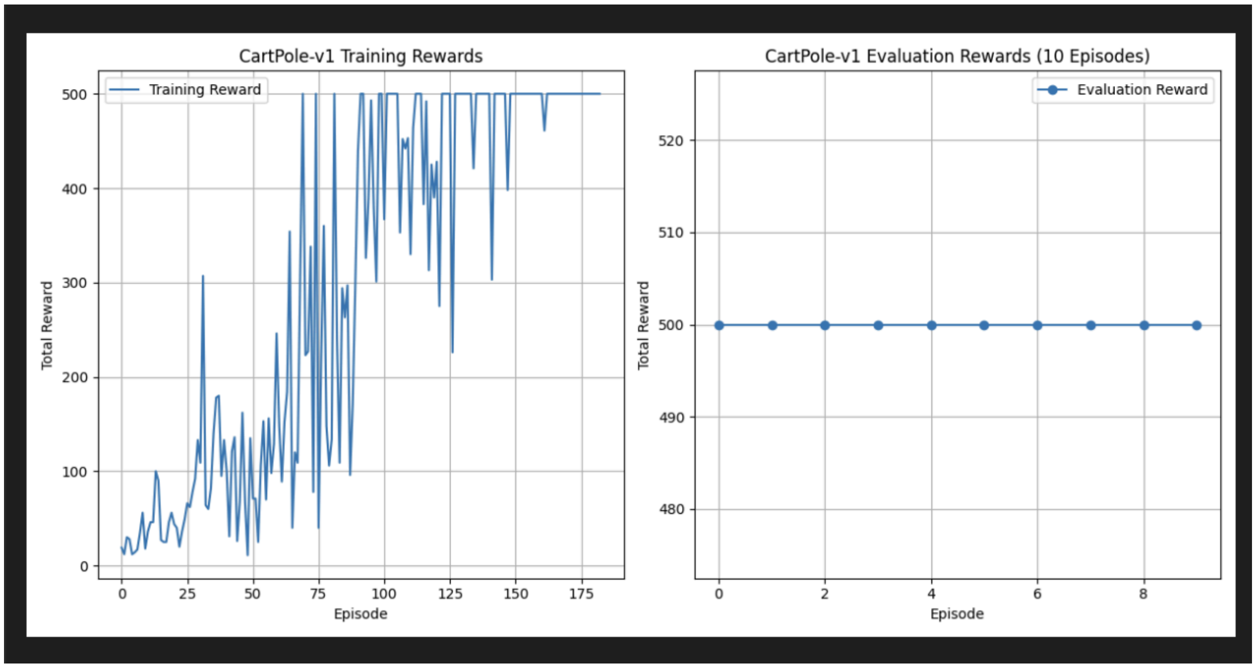
Environment Description:

- Observation Space: 4D continuous vector (cart position, velocity, pole angle, angular velocity).
- Action Space: 2 discrete actions (left or right).
- Goal: Keep the pole upright by moving the cart.
- Reward: +1 per time step pole is balanced. Max reward = 500.

Training and Evaluation:

- Training reward steadily increased to 500 (maximum achievable).
- Evaluation: 10/10 episodes scored 500.

Training and Evaluation Plot: [CartPole Training + Eval]



Analysis:

- PPO successfully solves CartPole.
- Rapid convergence within ~150 episodes.
- Evaluation shows perfect generalization.

2. LunarLander-v3

Environment Description:

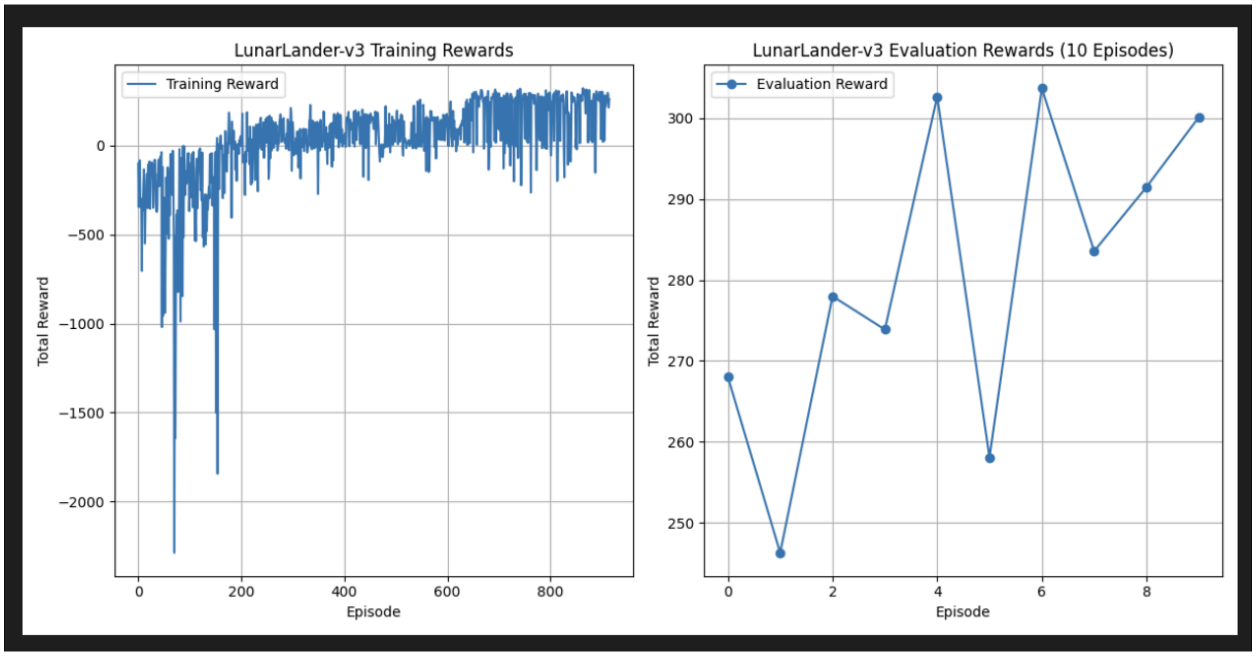
- Observation Space: 8D vector (position, velocity, angle, leg contacts).
- Action Space: 4 discrete actions.

- Goal: Land smoothly without crashing.
- Reward: Positive for soft landings, negative for crashes, bonus for leg contact.

Training and Evaluation:

- Training reward improved from ~-1000 to ~200+.
- Evaluation rewards ranged from 245 to 305.

Training and Evaluation Plot: [LunarLander Training + Eval]



Analysis:

- Steady learning curve.
- Final policy demonstrates controlled landing.
- Slight variability in evaluation rewards.

Mujoco Env

3. HalfCheetah-v4

Environment Description:

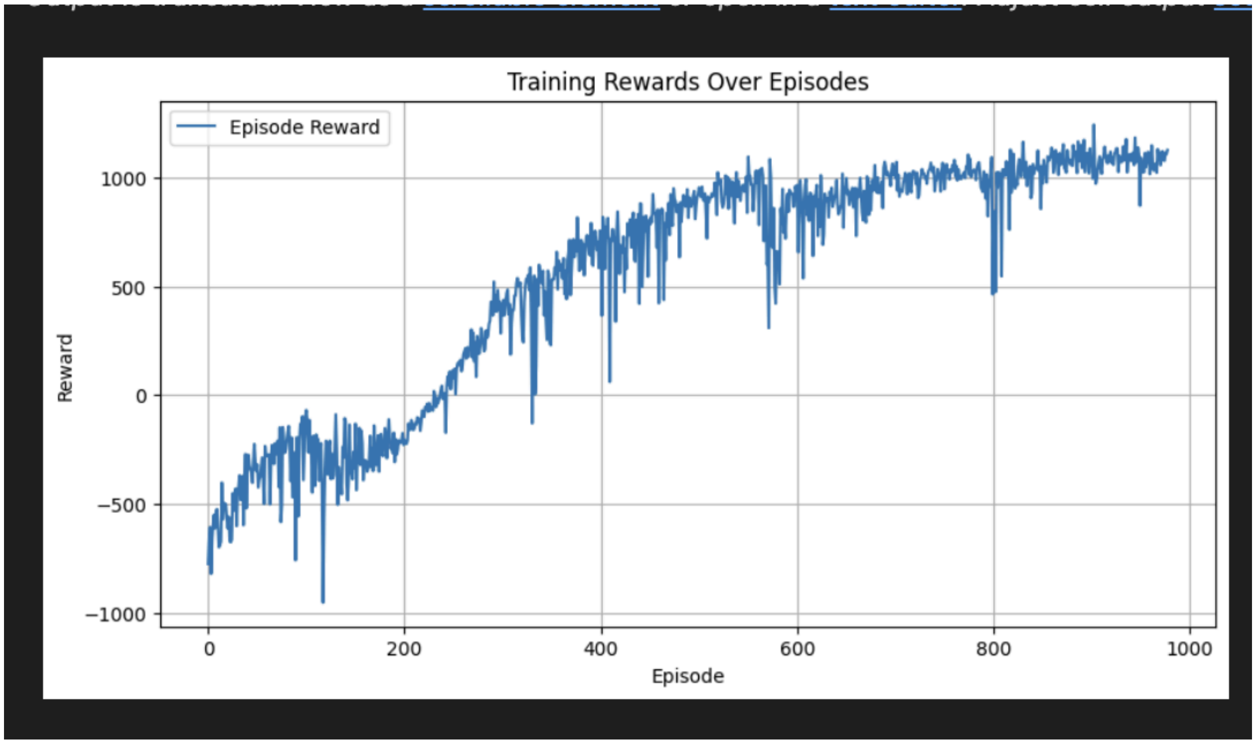
- Observation Space: 17D continuous vector.
- Action Space: 6D continuous (motor torques).
- Goal: Maximize forward velocity with control.
- Reward: Movement bonus minus energy cost.

Training and Evaluation:

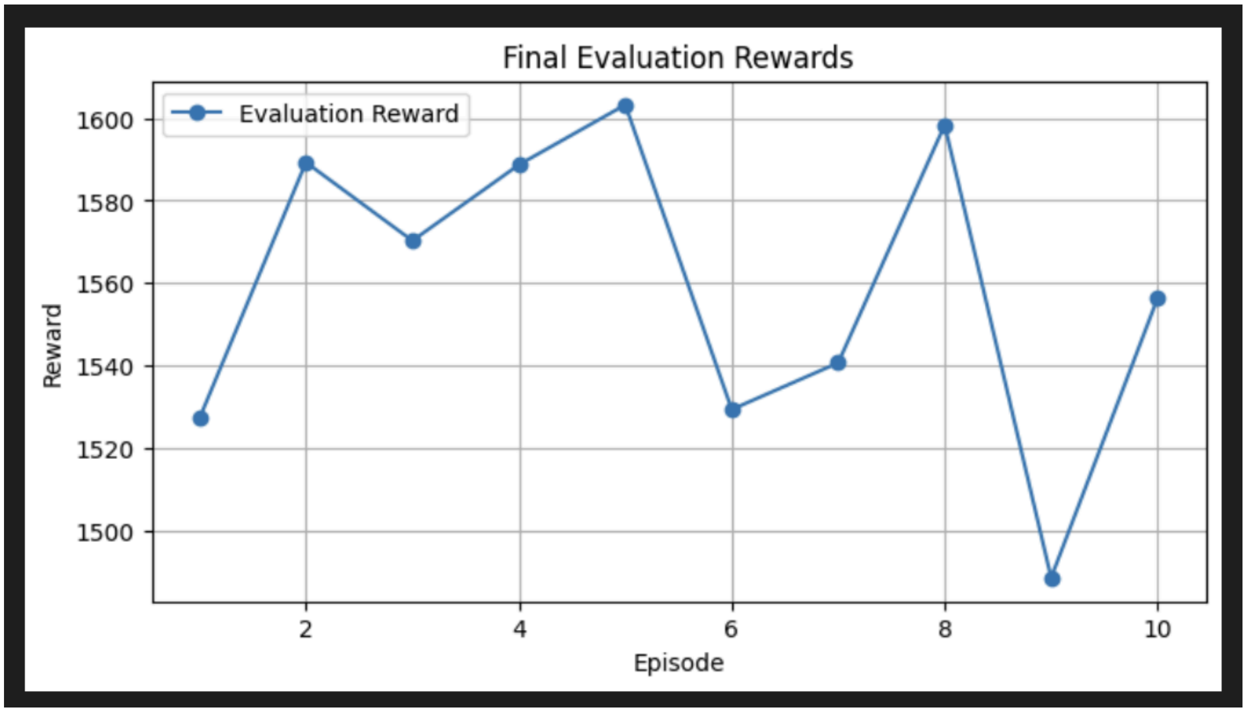
- Training reward climbed steadily to ~1200+.

- Evaluation rewards across 10 episodes averaged ~1556.

Training Curve: [HalfCheetah Training}:



Evaluation Plot: [HalfCheetah Eval]:



Analysis:

- PPO scales well to high-dimensional, continuous environments.
- Learned policy shows stable, natural locomotion.
- Some variation due to complex dynamics.

Summary Table

Environment	Final Train Reward	Evaluation Performance
CartPole-v1	500	(500)
LunarLander-v3	~200+	(270–310 avg)
HalfCheetah-v4	~1200–1600	(~1556 avg)

Conclusion

PPO demonstrated strong performance across tasks of varying difficulty and action spaces. It consistently learned competitive policies, especially when tuned correctly. Notably:

- Fast convergence on CartPole
- Robust learning on LunarLander
- High performance on complex HalfCheetah dynamics

This validates PPO as a strong general-purpose RL algorithm.

References:

- https://gymnasium.farama.org/environments/mujoco/half_cheetah/
- https://www.gymlibrary.dev/environments/atari/pong/
- Gymnasium environments

Team Member	**Assignment**	**Contribution**
Aniket Kumar	Part-2	40
Kumar Satyam	Part-2	60
Aniket Kumar	Part-3	60
Kumar Satyam	Part-3	40
Aniket Kumar	Bonus	50
Kumar Satyam	Bonus	50