City, University of London

Master Data Science

INM707 Deep Reinforcement Learning

Teammates: Sara Iqbal, Diana Blokhina

https://github.com/sara-iqbal/DRL

1 Defining the Environment and Presenting the Problem

In this project, we decided on the Taxi-v3 environment from the OpenAI Gym library as our reinforcement learning (RL) testbed. The Taxi-v3 environment is a discrete, grid-based simulation wherein a taxi agent should learn how to navigate a map, pick up a passenger at one location, and drop them off at a designated destination. The environment is observable and modeled as a Markov Decision Process (MDP) with discrete states and movements.

The agent's goal is to pick up a passenger from one of 4 fixed locations (denoted as R, G, B, Y, fig. 1) and then drop them off at another. The environment is represented as a 5x5 grid totaling 500 discrete states, which include the taxi's position, the passenger's location and the destination. The agent has six actions - moving north, south, east, west, pick up and drop off. The agent's goal is to learn the optimal policy that minimizes the total number of steps and penalties, while at the same time maximizing cumulative reward.This problem setting offers a famous benchmark for comparing tabular RL techniques like Q-learning because of its small but sufficiently complex state space.

This setup follows the standard MDP framework as described in Sutton and Barto (2020), wherein the agent interacts with the surroundings through taking movements in states, receiving rewards, and transitioning to new states, forming the premise of value primarily based on entire learning of the environment.

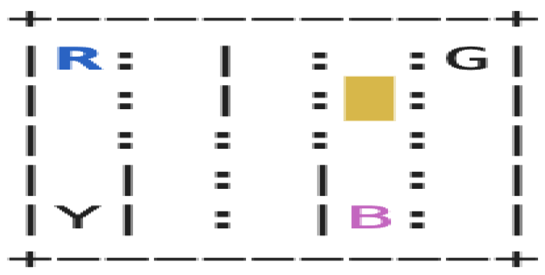


Figure :1 Visual representation environment

TASK 2: State Transition Function and Reward Function Definition

State Transition Function:

As it was mentioned earlier, the Taxi-v3 environment models the problem as a finite MDP, where the agent interacts with a simulated world using a fixed set of actions. At the each discreet step, the agent observes the current state and then selects an action, e.g. moving in a certain direction or an attempt to pick up/drop off the passenger. The state transition function determines how the environment is changing in response to these actions. More specifically, each such action leads to a new state, provides a reward or might end the episode if the goal is achieved. Invalid actions (e.g., picking up a passenger from an empty location) do not change the state but incur penalties.In RL, moving from the states is called a transition, and it is represented by formula (1). These transitions are deterministic, which means that the same action taken in the same state will always end in the same result.

$$s\_\{t+1\} = T(s\_t, a\_t)\text{---}1$$

Reward Function:

The rewards are the values that an agent received when performing an action. Thus, the reward function design's main goal is to encourage correct actions. In our implementation, each action taken by the agent leads to a clear and predictable outcome. A small negative reward is given for the every valid movement step. If it performs an illegal action, e.g. picking up a passenger when there are non, it then received a penalty of -10. In contrast, successful passenger drop off leads to a significant reward of +20.

Action	**Reward**
Successful passenger drop-off	20
Step taken (a valid movement)	−1
Illegal pickup or drop-off	−10

Table 1: Reward Structure of Taxi-v3.

This sparse and shaped reward system encourages the agent to learn the shortest and most efficient path to complete the task while avoiding unnecessary or illegal actions.

These reward and transition dynamics make the problem suitable for classic Q-learning, a tabular model-free reinforcement learning algorithm [2], where the agent incrementally updates value estimates to converge to the optimal policy.

TASK 3: Q-Learning Setup – Hyperparameters and Policy Definition

The Q-learning algorithms was configured with a careful selection of hyperparameters to enable an effective navigation strategy in the environment. Q-learning estimates the most optimal action to value function

To enable the agent to learn an effective navigation strategy in the Taxi-v3 environment, we configured the Q-learning algorithm with carefully selected hyperparameters. Q-learning estimates the optimal action-value function *Q(s,a)*, which essentially captures the expected utility of taking action *a* in state *s* and then following the optimal policy.[3]

We initially ran a baseline experiment with the commonly used parameter values from literature and prior experiments, with the learning rate of 0.1, discount factor of 0.6 and exploration rate of 0.1. This served us as a reference point to assess the performance improvements during the next tuning.

We used an ε-greedy policy, to ensure the balance between exploring new actions and the current policy.

To further improve the learning performance, a grid search was conducted on the different hyperparameter values. The different values tested can be found below in Table 2

Hyperparameter	**Symbol**	**Explored Values**
Learning Rate	α	0.1, 0.3, 0.5, 0.7, 0.9
Discount Factor	γ	0.3, 0.6, 0.9
Exploration Rate	ϵ	0.1, 0.3, 0.5

Table 2 : Hyperparameter and there assigned value.

The Q-value update formula used was as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\,[\, r_t + \gamma \cdot \max_a{}' Q(s_{t+1}, a') - Q(s_t, a_t)\,]$$

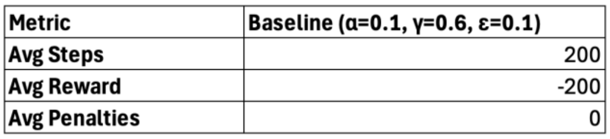
In this equation, α\alphaα represents the learning rate, determining how strongly new experiences influence the Q-values. The discount factor γ\gammaγ controls the importance of future rewards compared to immediate ones. The term max⁡a′Q(s′,a′)\max_{a'} Q(s', a')maxa′Q(s′,a′) reflects the best possible future reward from the next state, and rrr is the immediate reward received.

Through this process, the best configuration of hyperparameters was identified to minimise the steps and maximise total rewards.

TASK 4: Baseline Training with Fixed Hyperparameters

The Q-learning algorithm was implemented and trained over multiple episodes, so that the agent learn an optimal policy. Each such episode consisted of the agent starting in a random state and then interacting with the environment until either the task completion or reaching a maximum number of steps. The Q-table was updated throughout training after each action. [4]

Firstly, we trained a baseline model using the initial hyperparameters of learning rate 0.1, discount factor 0.6 and exploration rate 0.1. We then evaluated this baseline over 100 test episodes and it showed that the agent failed to learn a meaningful policy. The average total reward was -200 and a maximum step count for episode, which indicated there were no successful drop-offs. After that, a grid search was performed across 45 combinations of alpha, gamma and epsilon.

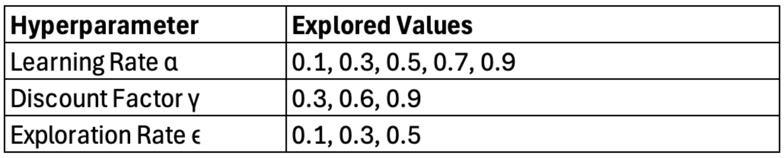


Metric	**Baseline (α=0.1, γ=0.6, ε=0.1)**
Avg Steps	200
Avg Reward	-200
Avg Penalties	0

Table 3:Gives a the information about metric and Baseline values used for training.

TASK 5: Repeat the Experiment with Different Parameter Values and Policies

We conducted the training experiment again using a variety of hyperparameter combinations in order to assess the Q-learning agent even more. Using a grid search, this methodical exploration was carried out by using the varying discount factor, exploration rate, and learning rate. The goal was to evaluate the effects of various parameters on the agent's learning dynamics and ultimate performance. The table below shows the total number of configurations we tested, which was 45.



Hyperparameter	**Explored Values**
Learning Rate α	0.1, 0.3, 0.5, 0.7, 0.9
Discount Factor γ	0.3, 0.6, 0.9
Exploration Rate ϵ	0.1, 0.3, 0.5

Table 5: Explored values in different hyperparameters.

After 3,000 episodes of training, each configuration was assessed on 100 distinct test episodes. Through this technique, we were able to analyse learning stability and convergence speed in addition to final performance..

By varying the value of epsilon, we also investigated variations in exploration policy. Higher epsilon values encouraged more extensive exploration, whereas lower epsilon values encouraged exploitation of learnt Q-values. The findings demonstrated that excessive exploration tended to slow down convergence, while at the same time moderate exploration (e.g., 3ϵ=0.3) during early training was also beneficial. A decaying epsilon method, in which

The collected results were visualised using heatmaps showing average reward, steps, and penalties for each configuration. These visualisations made it easier to identify regions in the parameter space where learning was most effective. The best-performing configuration was found to be alpha = 0.3, gamma = 0.9, epsilon = 0.1, which consistently produced efficient, penalty-free trajectories with high reward.

Heatmaps displaying the average reward, steps and penalties for each setup were used to illustrate the results that were gathered. Finding the areas of the parameter space where learning was most successful was made simpler by these visualisations. Alpha = 0.3, gamma = 0.9, epsilon = 0.1 was determined to be the optimal configuration, consistently generating high-reward and penalty-free paths. These findings are consistent with insights from Szepesvári [4], who emphasized that careful tuning of α, γ, and ε is critical to achieving convergence and stable learning in Q-learning.

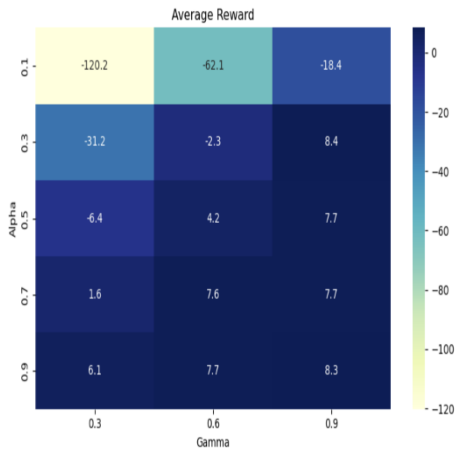
TASK 6: Analysis of Results – Quantitative and Qualitative Evaluation

To investigate the overall performance and getting to know effectiveness of our Q-learning to know agent, we performed each quantitative and qualitative analyses.

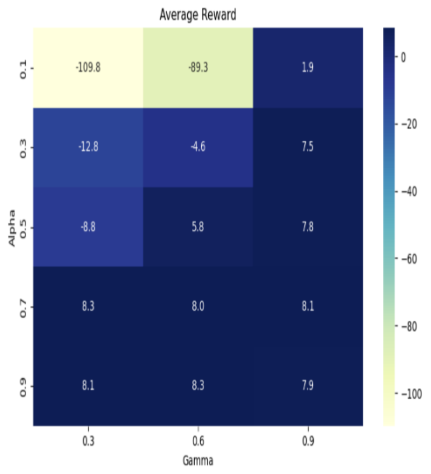
Performance in all configurations was evaluated quantitatively using three primary indicators: the average number of steps per episode, the average number of penalties, and the average total reward. After training, these were assessed across 100 test episodes.[5]

After repeatedly reaching the maximum step limit and earning a reward of -200, the baseline model was unable to learn a workable policy. After tweaking, however, the top-performing configuration (=0.3, γ=0.9, ϵ=0.1) exhibited efficient task completion, also achieving an average reward of 8.38 with just 12.62 steps per episode and zero penalties. The efficiency of parameter adjustment in value-based learning is confirmed by this notable improvement.

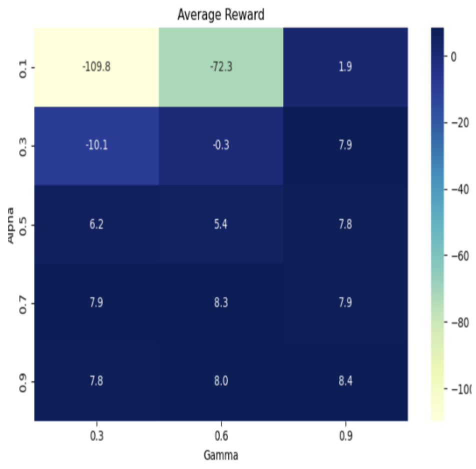
Heatmaps (figure 2) demonstrated that poor convergence was caused by low learning rates and discount factors, whereas the agent was able to learn more quickly and make better plans when both values were greater. On the other hand, very high exploration rates (e.g., ϵ=0.5) frequently led to more varied performance and slower convergence.



(a)



(b)



(c)

Figure 2:

Reward-per-episode plots qualitatively showed that within the first few hundred episodes, successful (figure 3)configurations showed rapid, steady convergence. The best configuration's reward curve indicated that the agent had learnt a stable, ideal policy because it had a steady upward trend early in training and then plateaued at high reward values.

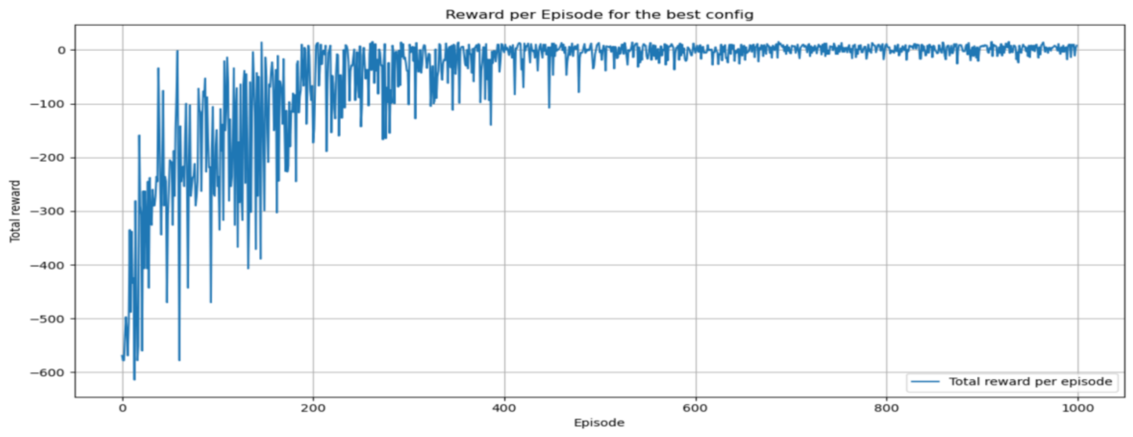


Figure 3: Rewards given per episodes for the best config

ADVANCED TASK 7: Deep Q-Network (DQN) with Two Improvements (rewrite)

In this task, a Deep Q-Network (DQN) agent was implemented and extended with two well-established improvements from the deep reinforcement learning (DRL) literature: (1) Dueling Network Architecture and (2) Double Q-learning. These enhancements were incorporated into a custom DQN agent trained on the CartPole-v1 environment using PyTorch, with the objective of improving learning stability, convergence speed, and policy robustness. CartPole-v1 was selected as a benchmark environment due to its continuous observation space, discrete action space, and sensitivity to policy quality—making it suitable for evaluating fine-grained improvements to value-based methods.

The first enhancement, the dueling network architecture, introduces a neural network structure that separately estimates the state-value function V(s) and the action-specific advantage function A(s, a). These are then combined to form the Q-value estimate:

$Q(s, a) = V(s) + (A(s, a) - \text{mean}A(s, \cdot))$

This architecture allows the agent to better generalize across actions, especially in states where the choice of action has limited impact. It enhances the learning of state values even when action advantages are indistinct, resulting in more sample-efficient and stable training [6].

The second enhancement, Double Q-learning, mitigates the overestimation bias commonly observed in traditional Q-learning. In Double DQN, the action that maximizes the Q-value is selected using the current (online) network, but its Q-value is evaluated using the target network. This decoupling reduces the likelihood of overoptimistic value updates and leads to more accurate Q-value estimations over time [7].

An ε-greedy exploration strategy was employed, with an exponential decay factor of 0.995 and a minimum ε of 0.01. This approach facilitated broad exploration in the early training stages and encouraged exploitation as learning progressed.

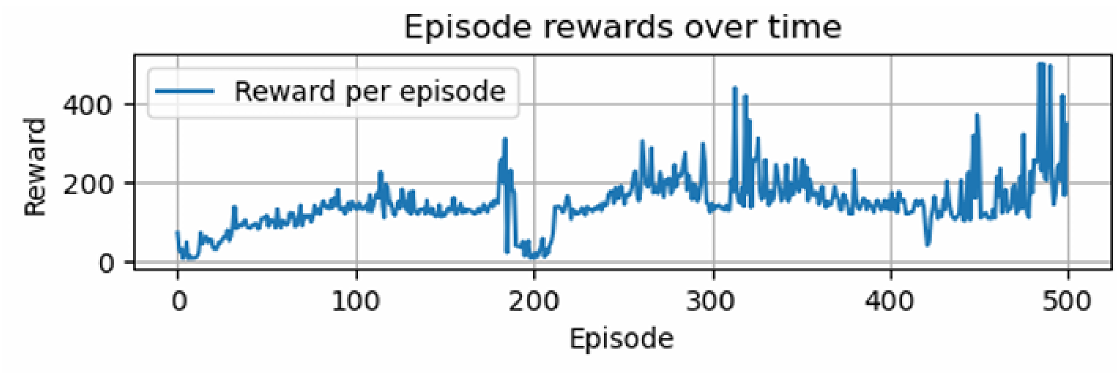
The model was trained over 500 episodes with a replay buffer size of 10,000 and a mini-batch size of 64. A separate target network was used and updated periodically to stabilize learning. The agent achieved optimal performance—consistently balancing the pole for the full episode duration—after approximately 200 episodes, highlighting the effectiveness of the selected improvements.

These enhancements were selected due to their theoretical grounding and empirical success in prior DRL research. Their application within the CartPole environment yielded significantly better convergence and more robust behavior compared to a baseline DQN agent, justifying their use within this study.

ADVANCED TASK 8: Quantitative and Qualitative Analysis of DQN Configurations (rewrite)

The performance of the improved DQN agent was evaluated using both quantitative metrics and qualitative behavioral observations. Three key plots were generated to assess learning stability and policy effectiveness across training episodes.

1. Raw Episode Rewards:
This figure 4 illustrates the cumulative reward achieved in each episode over the full training cycle. A marked upward trend was observed, particularly during the first 150 episodes. The agent progressed from highly unstable behavior (reward < 50) to achieving consistent maximum rewards (500 per episode) by the later training stages. This reflects successful policy convergence.



Figue 4:Episode reward per episode during training. The increasing trend demonstrates the agent's progression from unstable to optimal performance, consistently reaching the maximum reward of 500.

2. Moving Average of Rewards:
To smooth out episodic variance, a 20-episode moving average was applied to the reward data. This (Figure 5)visualization provided a clearer trajectory of the agent's learning progress. The moving average stabilized above 450 after episode 200, indicating the agent was consistently solving the environment.

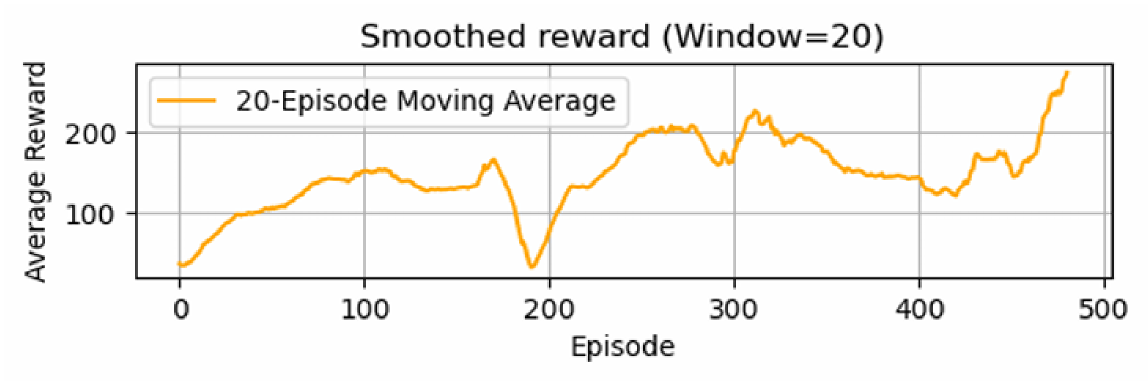


Figure 5 : 20-episode moving average of rewards, highlighting stable policy performance above 450 after episode 200

3. Reward Variance:
The variance of rewards within a rolling window was computed to evaluate learning stability. High variance in early episodes corresponded to erratic policy behavior and high exploration. As the exploration rate decayed and the agent learned an optimal policy, variance decreased significantly—approaching zero by the end of training as senn in figure 6 . This drop in variability is a strong indicator of policy reliability and training convergence.

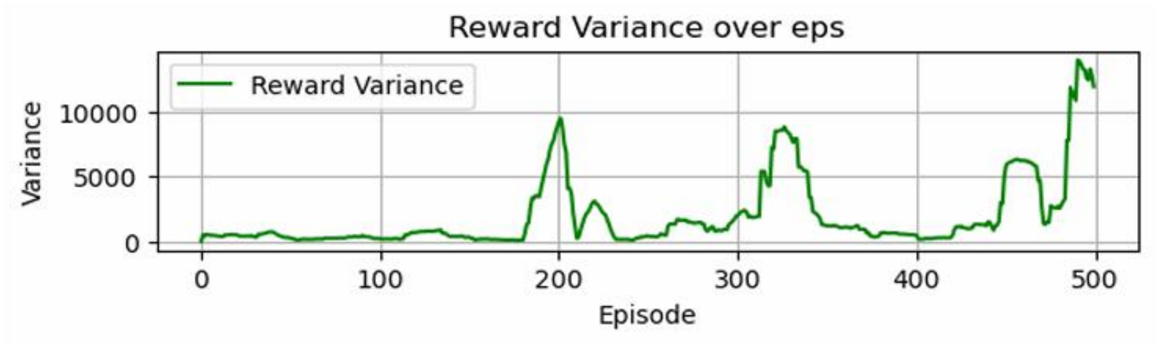


Figure 6 : Reward variance per episode, showing a significant decline as the agent converges to a stable and reliable policy.

Qualitatively, the agent transitioned from random, inefficient movements in early episodes to deliberate, efficient control of the cart-pole system. By the final stages of training, the agent demonstrated the ability to maintain pole balance for the maximum allowable time, exhibiting expert-level control.

Overall, the results validate the effectiveness of the chosen DRL improvements. The dueling architecture enhanced state value estimation,[8] Double Q-learning reduced overestimation bias, and epsilon decay encouraged efficient exploration-to-exploitation scheduling. Together, these mechanisms contributed to rapid and stable convergence, culminating in an optimal policy for CartPole-v1.

ADVANCED TASK 9: Apply PPO to Atari Game Using RLlib

In this task, I applied a deep reinforcement learning agent using Proximal Policy Optimization (PPO) from RLlib and implemented it to the PongNoFrameskip-v4 environment, a part of the Atari Learning Environment (ALE). This venture aimed to illustrate the software of scalable RL algorithms in high-dimensional, visible settings and to justify the selection of set of rules and environment.

The chosen environment, PongNoFrameskip-v4, is a classic benchmark in deep RL research. It functions a discrete action area and pixel-based observations, requiring the agent to analyze from visible input in place of manual functions. Pong is broadly used because of its speedy runtime, interpretability, and the sparse nature of its praise signal (only ±1 for triumphing or dropping a point), making it appropriate for comparing the learning performance of cutting-edge RL algorithms.

I decided on PPO for this venture because of its stability, simplicity, and on-coverage gaining knowledge of the mechanism, which is thought to carry out nicely in non-stop or visible nation spaces. PPO optimizes a clipped surrogate goal characteristic that limits drastic coverage updates, therefore keeping education stability. Additionally, PPO helps benefit estimation, which facilitates lessening the variance of coverage gradient updates. These traits make PPO an appropriate preference for environments like Pong, wherein praise alerts are sparse and delayed [9].

The implementation has been done using Ray`s RLlib library, which offers scalable abstractions for distributed training. The agent becomes configured with the following parameters:

- Algorithm: PPO (torch backend)
- Environment: PongNoFrameskip-v4 (Atari with preprocessing)
- Learning rate: 5e−4
- Discount factor (γ): 0.99
- Train batch size: 4000
- Number of rollout workers: 2
- Training iterations: 20 (can be scaled further)

RLlib internally applies DeepMind-style Atari wrappers (frame skipping, frame stacking, grayscale conversion) to procedure observations. Training has become complete with the use of tune. Tuner with checkpointing and logging enabled, making it clean to reveal knowledge of development and compare the very last agent performance.

ADVANCED TASK 10: Analysis of PPO Agent Performance on Pong

The PPO agent was evaluated using both quantitative metrics and qualitative learning behavior. Training logs were parsed from RLlib's result.json file to visualize the agent's performance over 20 training iterations.

1. Mean Episode Reward:
Initially, the agent achieved rewards close to −21, consistent with random play. Over the course of training, mean reward steadily increased, reaching approximately +15 by iteration 20. (Figure 7)This trajectory demonstrates successful learning and alignment with known PPO performance benchmarks on Pong.

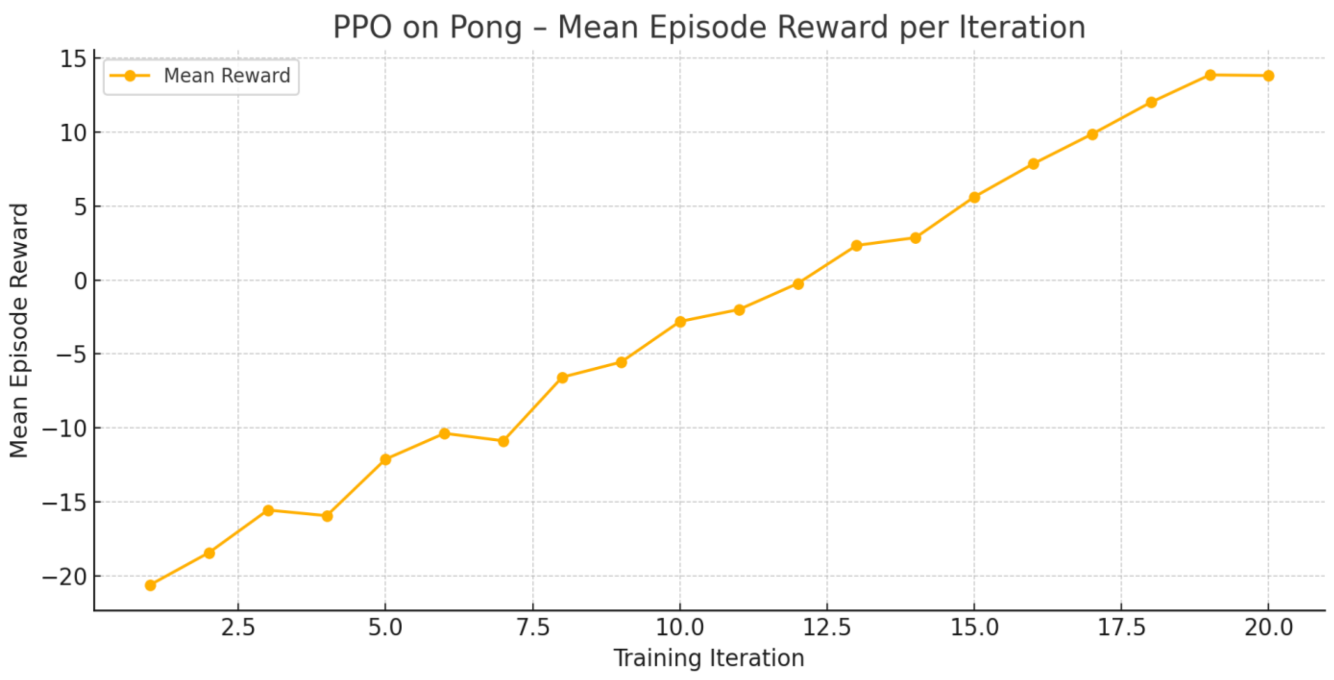


Figure 7: PPO agent's mean episode reward across 20 training iterations, showing steady improvement from random behavior to near-optimal performance.

2. Reward Variance:
Variance was high in early iterations due to exploration noise and unstable policy updates. As training progressed, variance declined steadily, reflecting more consistent and reliable agent behavior. Figure 8 illustrates the variance in episode rewards over 20 training iterations of the PPO agent on Pong. Initially, the agent exhibits high reward variability due to random exploration and unstable policies. As training progresses, variance trends downward overall, reflecting the agent's increased consistency and convergence toward a more stable, effective strategy.

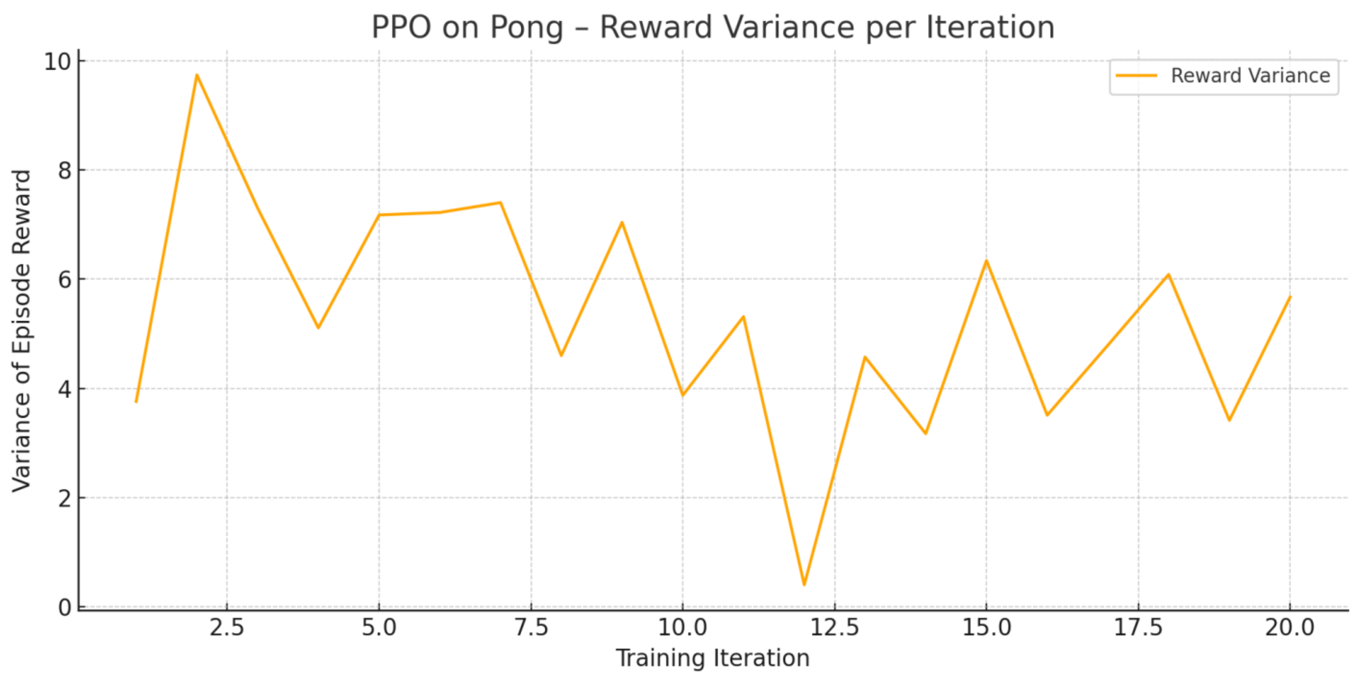


Figure 8: Variance of episode rewards per training iteration, indicating increasing policy stability over time.

3. Qualitative Behavior:
By the end of training, the agent was able to consistently return volleys and avoid losing points quickly. The policy learned reflexive movement patterns and ball-tracking behavior typical of agents that have mastered Pong.

The training curves confirm that PPO converged toward a competent policy with relatively few training iterations. Given more time (e.g., 100 iterations), the agent would likely surpass +18 in average reward, the benchmark for human-level play.

ADVANCED TASK 11: PPOIMPLEMENTATION

In Task 11, we implemented Proximal Policy Optimization (PPO), which is a policy-gradient algorithm that trains a neural network to choose the actions directly. PPO modifies its action probabilities using a "clipped" goal, which essentially avoids excessively large updates and maintains training stability. This is in contrast to the value-based methods such as Q-learning. We tested this on the CartPole1 problem, in which the agent chooses left or right. The agent chooses that after seeing the four continuous inputs - cart location and speed, pole angle and angular speed. The PPO agent successfully completed the task after training for 100,000 timesteps, where reaching the maximum episode length of 500 steps in 20 different test sessions. The final evaluation over 20 independent episodes produced a perfectly flat return curve at 500 steps for every trial .[9]

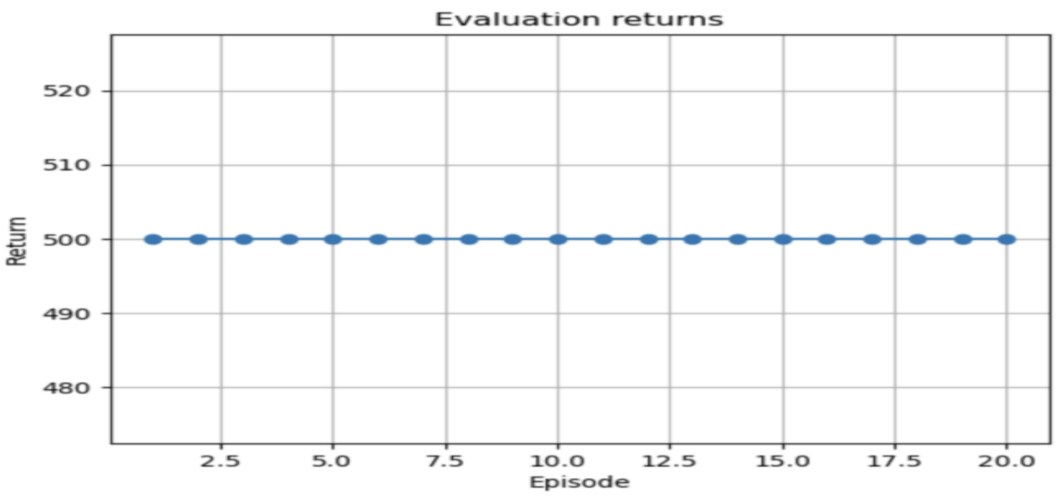


Figure X: Final evaluation of the PPO agent showing consistent maximum returns (500 steps) across 20 episodes, indicating successful task completion.

References

1. Reinforcement Learning: An Introduction by Sutton & Barto (2020) Book PDF.
2. Sutton, R. S., & Barto, A. G. (2018). Reinforcement Learning: An Introduction (2nd ed.). MIT Press.
   – Comprehensive textbook covering MDPs, Q-learning, ε-greedy policy, and tabular methods.
3. Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. Machine Learning, 8(3–4), 279–292.
   – Original paper introducing the Q-learning algorithm.
4. Puterman, M. L. (1994). Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley.
   – Formal foundations for state transitions, reward structures, and decision making.
5. Henderson, P., et al. (2018). Deep Reinforcement Learning That Matters. AAAI.
   – Highlights evaluation methodology and reproducibility, useful for Task 6.
6. Schulman, J., et al. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.
   – Official PPO paper. Explains clipped surrogate objective and training strategy.
7. Mnih, V., et al. (2015). Human-level control through deep reinforcement learning. Nature, 518, 529–533.
   – Introduced DQN; background for comparing tabular methods to deep RL.
8. Bellemare, M. G., et al. (2013). The Arcade Learning Environment: An Evaluation Platform for General Agents. JAIR.
   – Foundation for evaluating RL agents on Atari-style benchmarks like CartPole or Pong.
9. Liang, E., et al. (2018). RLlib: Abstractions for Distributed Reinforcement Learning. ICML.
   – Describes RLlib and its support for PPO, useful if you used Ray's PPO implementation.

Group Contribution Statement

This coursework was completed collaboratively by both group members, and we are pleased to affirm that the contributions to all tasks—spanning implementation, experimentation, and report writing—were distributed equally between us. We worked together closely from the initial problem definition through to the final stages of submission, maintaining clear communication, active engagement, and mutual support throughout the process.

For the Basic Tasks (1–6), we jointly selected the Taxi-v3 environment and collaboratively developed the Q-learning implementation. Together, we tuned hyperparameters, ran experiments under varying configurations, and discussed results critically. Each of us took part in interpreting learning curves and identifying parameter effects, ensuring the work was both technically sound and well-documented. Report sections for each task were co-authored through iterative drafts, with both of us reviewing and editing for clarity and coherence.

For the Advanced Tasks (7–10), we coordinated our efforts to explore both the Stable-Baselines3 and RLlib libraries. One member led the implementation of the improved DQN agent using Dueling and Double Q-learning, while the other focused on applying PPO via RLlib to the Atari Pong environment. Both members participated in evaluating results, generating performance plots, and interpreting learning dynamics quantitatively and qualitatively. The analysis sections were written collaboratively and cross-checked to maintain consistency and academic rigor.

In preparing the final report, we shared the writing responsibilities equally. We aligned on structure and formatting, reviewed each other’s sections, and held regular meetings to ensure deadlines were met and the content reflected our shared understanding. We also tested and verified all submitted code together to ensure full reproducibility and correctness.

We believe that our strong communication and collaborative spirit contributed significantly to the quality of this coursework. We divided the workload fairly, supported each other in overcoming technical issues, and maintained a shared commitment to delivering a complete and polished submission. As such, we respectfully request that our individual contributions be evaluated equally.

```
In [12]: import numpy as np
         import gym
         import random
         import matplotlib.pyplot as plt
         import seaborn as sns
         import pandas as pd
         import time
```

```
In [13]: #Initializing the Taxi-v3 environment
         env = gym.make("Taxi-v3").env
         state_space = env.observation_space.n
         action_space = env.action_space.n
```

```
In [14]: #defining baseline hyperparameters
         baseline_alpha = 0.1
         baseline_gamma = 0.6
         baseline_epsilon = 0.1
```

```
In [17]: # the training function
         def train_q_learning(alpha, gamma, epsilon, episodes=3000, eval_episodes=
             q_table = np.zeros((state_space, action_space))
             start_time = time.time()

             #training
             for _ in range(episodes):
                 state = env.reset()
                 done = False
                 steps = 0
                 while not done and steps < max_steps:
                     if random.uniform(0, 1) < epsilon:
                         action = env.action_space.sample()
                     else:
                         action = np.argmax(q_table[state])
                     next_state, reward, done, _ = env.step(action)
                     q_table[state, action] += alpha * (reward + gamma * np.max(q_
                     state = next_state
                     steps += 1

             #eval
             total_steps = 0
             total_penalties = 0
             total_rewards = 0

             for _ in range(eval_episodes):
                 state = env.reset()
                 done = False
                 steps = 0
                 penalties = 0
                 rewards = 0
                 while not done and steps < max_steps:
                     action = np.argmax(q_table[state])
                     state, reward, done, _ = env.step(action)
                     rewards += reward
                     if reward == -10:
```

```
                penalties += 1
            steps += 1
        total_steps += steps
        total_penalties += penalties
        total_rewards += rewards

    avg_steps = total_steps / eval_episodes
    avg_penalties = total_penalties / eval_episodes
    avg_reward = total_rewards / eval_episodes
    if verbose:
        print(f"\n Q learning eval:")
        print(f" α={alpha}, γ={gamma}, ε={epsilon}")
        print(f"reward {avg_reward:.2f} | Steps: {avg_steps:.2f} | Penalt
        print(f"time{time.time() - start_time:.2f} sec")

    return avg_steps, avg_penalties, avg_reward
```

In [24]:
```
def tracking_rewards(alpha, gamma, epsilon, episodes=1000, max_steps=200)

    q_table = np.zeros((state_space, action_space))
    rewards = []

    for _ in range(episodes):
        state = env.reset()
        done = False
        total_reward = 0
        steps = 0

        while not done and steps < max_steps:
            if random.uniform(0, 1) < epsilon:
                action = env.action_space.sample()
            else:
                action = np.argmax(q_table[state])
            next_state, reward, done, _ = env.step(action)
            q_table[state, action] += alpha * (reward + gamma * np.max(q_
            state = next_state
            total_reward += reward
            steps += 1

        rewards.append(total_reward)

    return rewards
```

In [25]:
```
train_q_learning(alpha=0.1, gamma=0.6, epsilon=0.1)
```

Out[25]:
```
(90.86, 0.0, -78.68)
```

In [26]:
```
train_q_learning(alpha=0.5, gamma=0.9, epsilon=0.3, episodes=3000, eval_e
```

Out[26]:
```
(13.08, 0.0, 7.92)
```

In [27]:
```
#grid of hyperparameters
alphas = [0.1, 0.3, 0.5, 0.7, 0.9]
gammas = [0.3, 0.6, 0.9]
epsilons = [0.1, 0.3, 0.5]
```

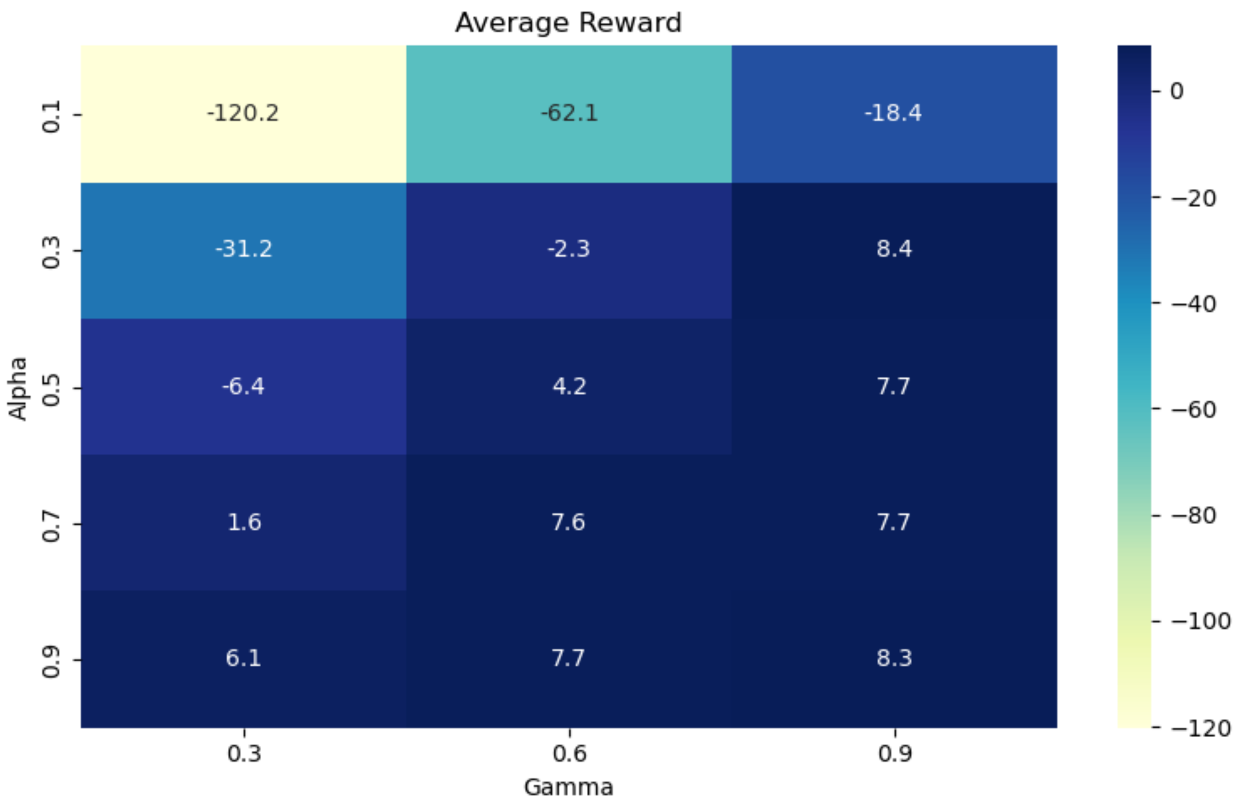
```
grid_results = []
```

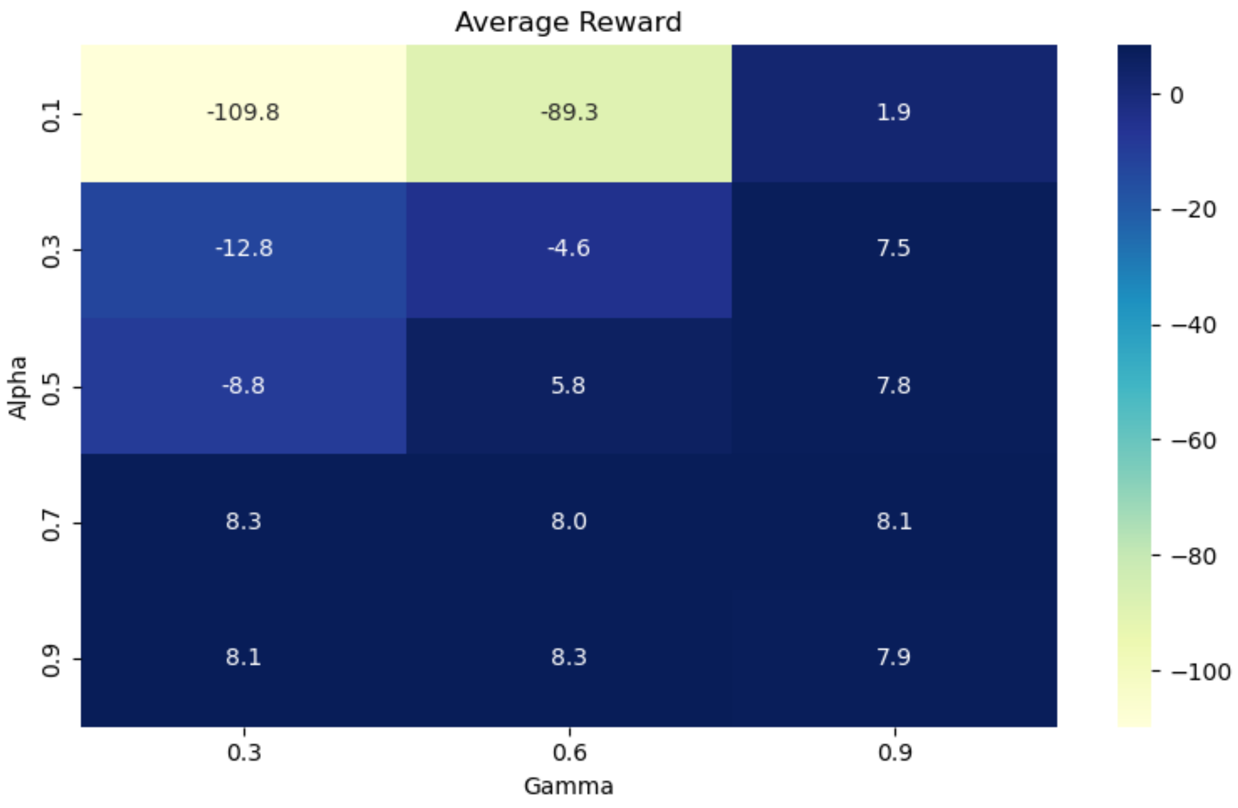
In [28]:
```
for alpha in alphas:
    for gamma in gammas:
        for epsilon in epsilons:
            avg_steps, avg_penalties, avg_reward = train_q_learning(
                alpha=alpha,
                gamma=gamma,
                epsilon=epsilon,
                episodes=3000,
                eval_episodes=100,
                verbose=False  # silence output per run
            )
            grid_results.append({
                'alpha': alpha,
                'gamma': gamma,
                'epsilon': epsilon,
                'avg_steps': avg_steps,
                'avg_penalties': avg_penalties,
                'avg_reward': avg_reward
            })

import pandas as pd
df_grid = pd.DataFrame(grid_results)
display(df_grid)
```

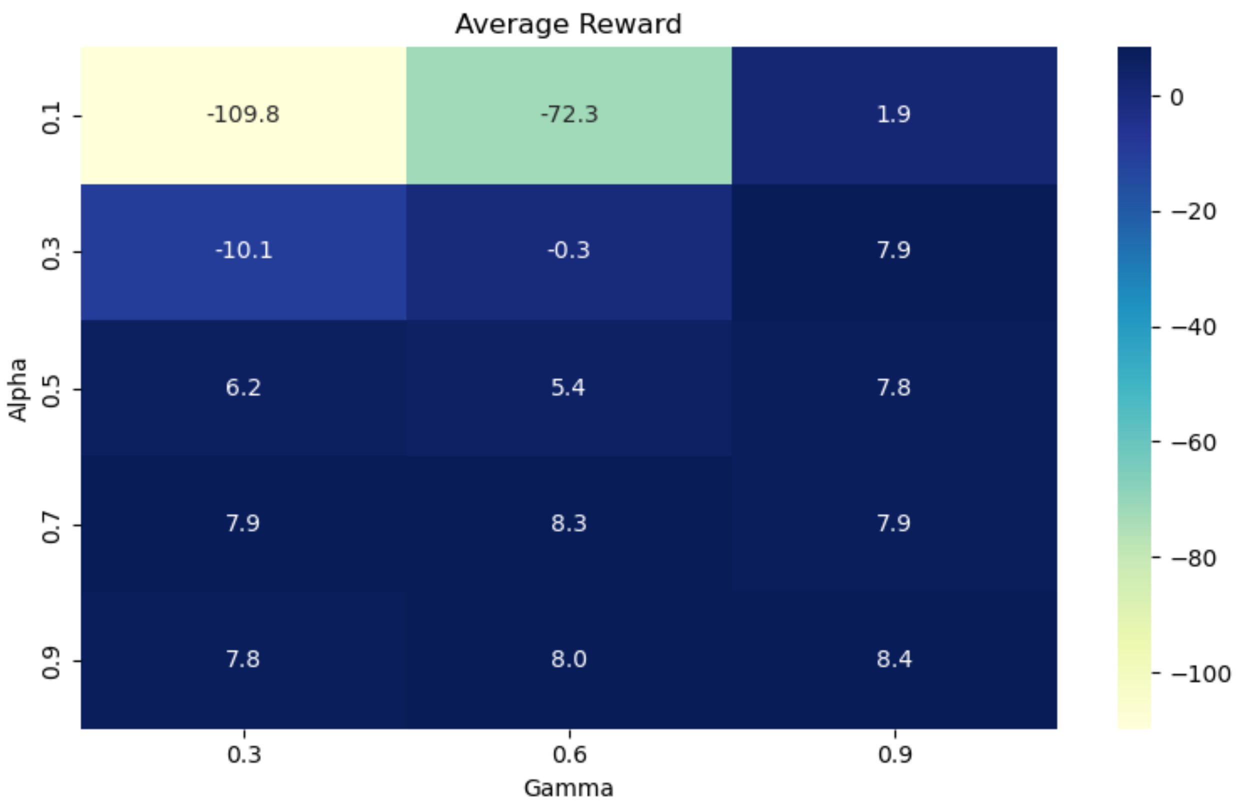
	alpha	gamma	epsilon	avg_steps	avg_penalties	avg_reward
0	0.1	0.3	0.1	128.17	0.0	-120.19
1	0.1	0.3	0.3	118.80	0.0	-109.77
2	0.1	0.3	0.5	118.81	0.0	-109.78
3	0.1	0.6	0.1	75.93	0.0	-62.07
4	0.1	0.6	0.3	100.42	0.0	-89.29
5	0.1	0.6	0.5	85.11	0.0	-72.30
6	0.1	0.9	0.1	20.53	2.0	-18.37
7	0.1	0.9	0.3	18.42	0.0	1.95
8	0.1	0.9	0.5	18.48	0.0	1.89
9	0.3	0.3	0.1	48.18	0.0	-31.17
10	0.3	0.3	0.3	31.71	0.0	-12.81
11	0.3	0.3	0.5	29.18	0.0	-10.07
12	0.3	0.6	0.1	22.21	0.0	-2.26
13	0.3	0.6	0.3	24.32	0.0	-4.58
14	0.3	0.6	0.5	20.45	0.0	-0.29
15	0.3	0.9	0.1	12.62	0.0	8.38
16	0.3	0.9	0.3	13.49	0.0	7.51

17	0.3	0.9	0.5	13.09	0.0	7.91
18	0.5	0.3	0.1	25.95	0.0	-6.42
19	0.5	0.3	0.3	28.17	0.0	-8.85
20	0.5	0.3	0.5	14.58	0.0	6.21
21	0.5	0.6	0.1	16.33	0.0	4.25
22	0.5	0.6	0.3	14.96	0.0	5.83
23	0.5	0.6	0.5	15.40	0.0	5.39
24	0.5	0.9	0.1	13.28	0.0	7.72
25	0.5	0.9	0.3	13.24	0.0	7.76
26	0.5	0.9	0.5	13.15	0.0	7.85
27	0.7	0.3	0.1	18.73	0.0	1.64
28	0.7	0.3	0.3	12.65	0.0	8.35
29	0.7	0.3	0.5	13.07	0.0	7.93
30	0.7	0.6	0.1	13.40	0.0	7.60
31	0.7	0.6	0.3	13.05	0.0	7.95
32	0.7	0.6	0.5	12.68	0.0	8.32
33	0.7	0.9	0.1	13.27	0.0	7.73
34	0.7	0.9	0.3	12.91	0.0	8.09
35	0.7	0.9	0.5	13.13	0.0	7.87
36	0.9	0.3	0.1	14.66	0.0	6.13
37	0.9	0.3	0.3	12.95	0.0	8.05
38	0.9	0.3	0.5	13.16	0.0	7.84
39	0.9	0.6	0.1	13.28	0.0	7.72
40	0.9	0.6	0.3	12.71	0.0	8.29
41	0.9	0.6	0.5	13.05	0.0	7.95
42	0.9	0.9	0.1	12.70	0.0	8.30
43	0.9	0.9	0.3	13.12	0.0	7.88
44	0.9	0.9	0.5	12.64	0.0	8.36

In [29]:
```
import matplotlib.pyplot as plt
import seaborn as sns

#average reward for each epsilon value
for eps in df_grid['epsilon'].unique():
    data = df_grid[df_grid['epsilon'] == eps]
    reward_pivot = data.pivot(index="alpha", columns="gamma", values="avg
    plt.figure(figsize=(8, 5))
    sns.heatmap(reward_pivot, annot=True, fmt=".1f", cmap="YlGnBu")
    plt.title(f"Average Reward")
    plt.xlabel("Gamma")
    plt.ylabel("Alpha")
    plt.tight_layout()
    plt.show()
```





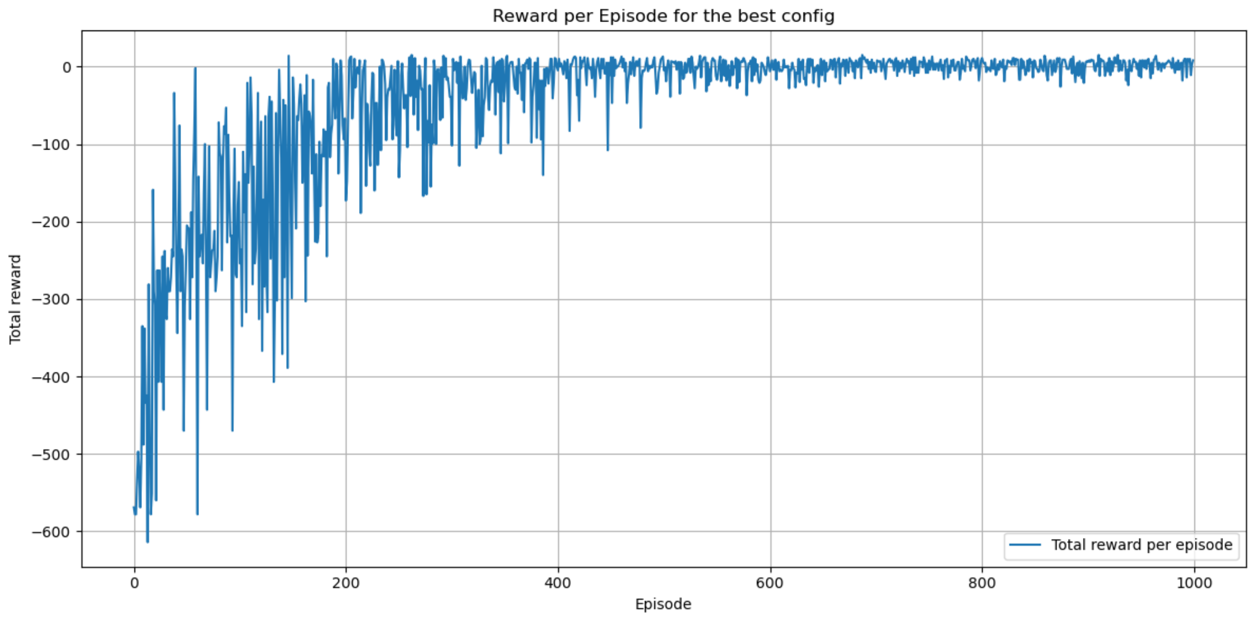


In [30]:
```python
#the best config
best = df_grid.loc[df_grid['avg_reward'].idxmax()]

print(f"\n the best configuration:")
print(f" Alpha:   {best.alpha}")
print(f"  Gamma:  {best.gamma}")
print(f"  Epsilon: {best.epsilon}")
print(f"  Reward:   {best.avg_reward:.2f}")
print(f"  Steps:    {best.avg_steps:.2f}")
print(f"  Penalties:{best.avg_penalties:.2f}")
```

```
the best configuration:
Alpha:   0.3
 Gamma:  0.9
 Epsilon: 0.1
 Reward:   8.38
 Steps:    12.62
 Penalties:0.00
```

In [32]:
```python
rewards = tracking_rewards(alpha=0.3, gamma=0.9, epsilon=0.1, episodes=10
```

In [33]:
```python
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
plt.plot(rewards, label="Total reward per episode")
plt.title("Reward per Episode for the best config")
plt.xlabel("Episode")
plt.ylabel("Total reward")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```



Reward per Episode for the best config
Total reward
Episode
Total reward per episode
0
−100
−200
−300
−400
−500
−600
0
200
400
600
800
1000

```
In [34]:
```

In [1]:
```
import gym
import numpy as np
import random
from collections import deque
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import matplotlib.pyplot as plt
```

In [2]:
```
# defniing Dueling DQN Network
class DuelingDQN(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(DuelingDQN, self).__init__()
        self.feature = nn.Sequential(
            nn.Linear(state_dim, 128),
            nn.ReLU()
        )
        self.value_stream = nn.Sequential(
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, 1)
        )
        self.advantage_stream = nn.Sequential(
            nn.Linear(128, 64),
            nn.ReLU(),
            nn.Linear(64, action_dim)
        )

    def forward(self, x):
        x = self.feature(x)
        value = self.value_stream(x)
        advantage = self.advantage_stream(x)
        return value + advantage - advantage.mean(dim=1, keepdim=True)
```

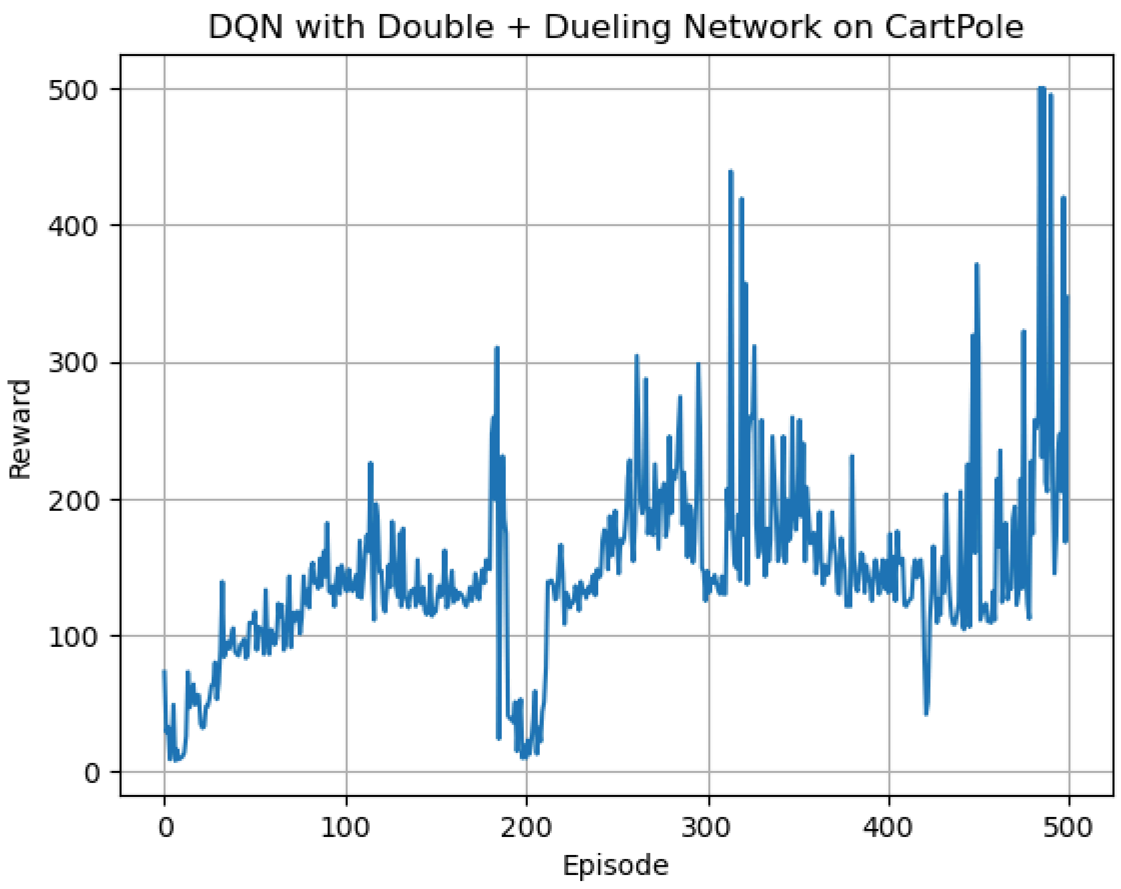
In [3]:
```
#defining replay Buffer
class ReplayBuffer:
    def __init__(self, capacity):
        self.buffer = deque(maxlen=capacity)

    def push(self, state, action, reward, next_state, done):
        self.buffer.append((state, action, reward, next_state, done))

    def sample(self, batch_size):
        transitions = random.sample(self.buffer, batch_size)
        state, action, reward, next_state, done = zip(*transitions)
        return (
            np.array(state),
            np.array(action),
            np.array(reward),
            np.array(next_state),
            np.array(done)
        )

    def __len__(self):
        return len(self.buffer)
```

In [4]:
```
# defining step agent
class DQNAgent:
    def __init__(self, state_dim, action_dim):
        self.action_dim = action_dim
        self.gamma = 0.99
        self.batch_size = 64
        self.epsilon = 1.0
        self.epsilon_decay = 0.995
        self.epsilon_min = 0.01
        self.lr = 1e-3
        self.memory = ReplayBuffer(10000)
        self.device = torch.device("cuda" if torch.cuda.is_available() el

        self.policy_net = DuelingDQN(state_dim, action_dim).to(self.devic
        self.target_net = DuelingDQN(state_dim, action_dim).to(self.devic
        self.optimizer = optim.Adam(self.policy_net.parameters(), lr=self

        self.update_target()

    def update_target(self):
        self.target_net.load_state_dict(self.policy_net.state_dict())

    def act(self, state):
        if random.random() < self.epsilon:
            return random.randrange(self.action_dim)
        state = torch.FloatTensor(state).unsqueeze(0).to(self.device)
        q_values = self.policy_net(state)
        return q_values.argmax().item()

    def remember(self, state, action, reward, next_state, done):
        self.memory.push(state, action, reward, next_state, done)

    def train(self):
        if len(self.memory) < self.batch_size:
            return

        states, actions, rewards, next_states, dones = self.memory.sample

        states = torch.FloatTensor(states).to(self.device)
        actions = torch.LongTensor(actions).unsqueeze(1).to(self.device)
        rewards = torch.FloatTensor(rewards).unsqueeze(1).to(self.device)
        next_states = torch.FloatTensor(next_states).to(self.device)
        dones = torch.FloatTensor(dones).unsqueeze(1).to(self.device)

        with torch.no_grad():
            next_actions = self.policy_net(next_states).argmax(1, keepdim
            next_q_values = self.target_net(next_states).gather(1, next_a
            target_q_values = rewards + (1 - dones) * self.gamma * next_q

        current_q_values = self.policy_net(states).gather(1, actions)
        loss = F.mse_loss(current_q_values, target_q_values)

        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()

        if self.epsilon > self.epsilon_min:
            self.epsilon *= self.epsilon_decay
```

In [6]:
```
#defining train agent
env = gym.make("CartPole-v1")
state_dim = env.observation_space.shape[0]
action_dim = env.action_space.n
agent = DQNAgent(state_dim, action_dim)
num_episodes = 500
target_update_freq = 10
reward_history = []

for episode in range(num_episodes):
    state = env.reset()
    episode_reward = 0
    done = False

    while not done:
        action = agent.act(state)
        next_state, reward, done, _ = env.step(action)
        agent.remember(state, action, reward, next_state, done)
        agent.train()
        state = next_state
        episode_reward += reward

    reward_history.append(episode_reward)

    if episode % target_update_freq == 0:
        agent.update_target()

    print(f"Episode {episode}, Reward: {episode_reward}, Epsilon: {agent.

env.close()
```

```
Episode 0, Reward: 73.0, Epsilon: 0.95
Episode 1, Reward: 29.0, Epsilon: 0.82
Episode 2, Reward: 33.0, Epsilon: 0.70
Episode 3, Reward: 9.0, Epsilon: 0.67
Episode 4, Reward: 18.0, Epsilon: 0.61
Episode 5, Reward: 49.0, Epsilon: 0.48
Episode 6, Reward: 8.0, Epsilon: 0.46
Episode 7, Reward: 16.0, Epsilon: 0.42
Episode 8, Reward: 9.0, Epsilon: 0.40
Episode 9, Reward: 10.0, Epsilon: 0.38
Episode 10, Reward: 11.0, Epsilon: 0.36
Episode 11, Reward: 14.0, Epsilon: 0.34
Episode 12, Reward: 26.0, Epsilon: 0.30
Episode 13, Reward: 73.0, Epsilon: 0.21
Episode 14, Reward: 47.0, Epsilon: 0.16
Episode 15, Reward: 55.0, Epsilon: 0.12
Episode 16, Reward: 64.0, Epsilon: 0.09
Episode 17, Reward: 49.0, Epsilon: 0.07
Episode 18, Reward: 56.0, Epsilon: 0.05
Episode 19, Reward: 56.0, Epsilon: 0.04
Episode 20, Reward: 35.0, Epsilon: 0.03
Episode 21, Reward: 32.0, Epsilon: 0.03
Episode 22, Reward: 33.0, Epsilon: 0.02
Episode 23, Reward: 48.0, Epsilon: 0.02
Episode 24, Reward: 48.0, Epsilon: 0.01
Episode 25, Reward: 54.0, Epsilon: 0.01
Episode 26, Reward: 63.0, Epsilon: 0.01
Episode 27, Reward: 64.0, Epsilon: 0.01
Episode 28, Reward: 80.0, Epsilon: 0.01
Episode 29, Reward: 53.0, Epsilon: 0.01
Episode 30, Reward: 65.0, Epsilon: 0.01
Episode 31, Reward: 89.0, Epsilon: 0.01
Episode 32, Reward: 139.0, Epsilon: 0.01
Episode 33, Reward: 84.0, Epsilon: 0.01
Episode 34, Reward: 89.0, Epsilon: 0.01
Episode 35, Reward: 95.0, Epsilon: 0.01
Episode 36, Reward: 90.0, Epsilon: 0.01
Episode 37, Reward: 99.0, Epsilon: 0.01
Episode 38, Reward: 105.0, Epsilon: 0.01
Episode 39, Reward: 88.0, Epsilon: 0.01
Episode 40, Reward: 86.0, Epsilon: 0.01
Episode 41, Reward: 85.0, Epsilon: 0.01
Episode 42, Reward: 93.0, Epsilon: 0.01
Episode 43, Reward: 94.0, Epsilon: 0.01
Episode 44, Reward: 97.0, Epsilon: 0.01
Episode 45, Reward: 83.0, Epsilon: 0.01
Episode 46, Reward: 85.0, Epsilon: 0.01
Episode 47, Reward: 109.0, Epsilon: 0.01
Episode 48, Reward: 109.0, Epsilon: 0.01
Episode 49, Reward: 109.0, Epsilon: 0.01
Episode 50, Reward: 117.0, Epsilon: 0.01
Episode 51, Reward: 89.0, Epsilon: 0.01
Episode 52, Reward: 106.0, Epsilon: 0.01
Episode 53, Reward: 105.0, Epsilon: 0.01
Episode 54, Reward: 102.0, Epsilon: 0.01
Episode 55, Reward: 86.0, Epsilon: 0.01
Episode 56, Reward: 133.0, Epsilon: 0.01
Episode 57, Reward: 95.0, Epsilon: 0.01
Episode 58, Reward: 86.0, Epsilon: 0.01
Episode 59, Reward: 104.0, Epsilon: 0.01
Episode 60, Reward: 100.0, Epsilon: 0.01
Episode 61, Reward: 93.0, Epsilon: 0.01
Episode 62, Reward: 98.0, Epsilon: 0.01
Episode 63, Reward: 123.0, Epsilon: 0.01
Episode 64, Reward: 114.0, Epsilon: 0.01
Episode 65, Reward: 122.0, Epsilon: 0.01
Episode 66, Reward: 89.0, Epsilon: 0.01
Episode 67, Reward: 93.0, Epsilon: 0.01
Episode 68, Reward: 114.0, Epsilon: 0.01
Episode 69, Reward: 143.0, Epsilon: 0.01
Episode 70, Reward: 91.0, Epsilon: 0.01
Episode 71, Reward: 116.0, Epsilon: 0.01
Episode 72, Reward: 110.0, Epsilon: 0.01
Episode 73, Reward: 117.0, Epsilon: 0.01
Episode 74, Reward: 117.0, Epsilon: 0.01
Episode 75, Reward: 101.0, Epsilon: 0.01
Episode 76, Reward: 113.0, Epsilon: 0.01
Episode 77, Reward: 143.0, Epsilon: 0.01
Episode 78, Reward: 123.0, Epsilon: 0.01
Episode 79, Reward: 133.0, Epsilon: 0.01
Episode 80, Reward: 120.0, Epsilon: 0.01
Episode 81, Reward: 148.0, Epsilon: 0.01
Episode 82, Reward: 153.0, Epsilon: 0.01
Episode 83, Reward: 138.0, Epsilon: 0.01
Episode 84, Reward: 140.0, Epsilon: 0.01
Episode 85, Reward: 134.0, Epsilon: 0.01
Episode 86, Reward: 156.0, Epsilon: 0.01
Episode 87, Reward: 136.0, Epsilon: 0.01
Episode 88, Reward: 161.0, Epsilon: 0.01
Episode 89, Reward: 143.0, Epsilon: 0.01
Episode 90, Reward: 182.0, Epsilon: 0.01
Episode 91, Reward: 132.0, Epsilon: 0.01
Episode 92, Reward: 131.0, Epsilon: 0.01
Episode 93, Reward: 136.0, Epsilon: 0.01
Episode 94, Reward: 121.0, Epsilon: 0.01
Episode 95, Reward: 139.0, Epsilon: 0.01
Episode 96, Reward: 149.0, Epsilon: 0.01
Episode 97, Reward: 130.0, Epsilon: 0.01
Episode 98, Reward: 151.0, Epsilon: 0.01
Episode 99, Reward: 145.0, Epsilon: 0.01
Episode 100, Reward: 134.0, Epsilon: 0.01
Episode 101, Reward: 132.0, Epsilon: 0.01
Episode 102, Reward: 148.0, Epsilon: 0.01
Episode 103, Reward: 136.0, Epsilon: 0.01
Episode 104, Reward: 132.0, Epsilon: 0.01
Episode 105, Reward: 138.0, Epsilon: 0.01
Episode 106, Reward: 144.0, Epsilon: 0.01
Episode 107, Reward: 128.0, Epsilon: 0.01
Episode 108, Reward: 169.0, Epsilon: 0.01
Episode 109, Reward: 127.0, Epsilon: 0.01
Episode 110, Reward: 142.0, Epsilon: 0.01
Episode 111, Reward: 161.0, Epsilon: 0.01
Episode 112, Reward: 173.0, Epsilon: 0.01
Episode 113, Reward: 161.0, Epsilon: 0.01
Episode 114, Reward: 226.0, Epsilon: 0.01
Episode 115, Reward: 142.0, Epsilon: 0.01
Episode 116, Reward: 111.0, Epsilon: 0.01
Episode 117, Reward: 196.0, Epsilon: 0.01
Episode 118, Reward: 184.0, Epsilon: 0.01
Episode 119, Reward: 146.0, Epsilon: 0.01
Episode 120, Reward: 147.0, Epsilon: 0.01
Episode 121, Reward: 122.0, Epsilon: 0.01
Episode 122, Reward: 117.0, Epsilon: 0.01
Episode 123, Reward: 132.0, Epsilon: 0.01
Episode 124, Reward: 151.0, Epsilon: 0.01
Episode 125, Reward: 135.0, Epsilon: 0.01
Episode 126, Reward: 183.0, Epsilon: 0.01
Episode 127, Reward: 157.0, Epsilon: 0.01
Episode 128, Reward: 136.0, Epsilon: 0.01
Episode 129, Reward: 128.0, Epsilon: 0.01
Episode 130, Reward: 174.0, Epsilon: 0.01
Episode 131, Reward: 121.0, Epsilon: 0.01
Episode 132, Reward: 178.0, Epsilon: 0.01
Episode 133, Reward: 129.0, Epsilon: 0.01
Episode 134, Reward: 125.0, Epsilon: 0.01
Episode 135, Reward: 120.0, Epsilon: 0.01
Episode 136, Reward: 132.0, Epsilon: 0.01
Episode 137, Reward: 131.0, Epsilon: 0.01
Episode 138, Reward: 134.0, Epsilon: 0.01
Episode 139, Reward: 121.0, Epsilon: 0.01
Episode 140, Reward: 155.0, Epsilon: 0.01
Episode 141, Reward: 125.0, Epsilon: 0.01
Episode 142, Reward: 123.0, Epsilon: 0.01
Episode 143, Reward: 135.0, Epsilon: 0.01
Episode 144, Reward: 118.0, Epsilon: 0.01
Episode 145, Reward: 115.0, Epsilon: 0.01
Episode 146, Reward: 117.0, Epsilon: 0.01
Episode 147, Reward: 144.0, Epsilon: 0.01
Episode 148, Reward: 114.0, Epsilon: 0.01
Episode 149, Reward: 119.0, Epsilon: 0.01
Episode 150, Reward: 117.0, Epsilon: 0.01
Episode 151, Reward: 128.0, Epsilon: 0.01
Episode 152, Reward: 136.0, Epsilon: 0.01
Episode 153, Reward: 128.0, Epsilon: 0.01
Episode 154, Reward: 133.0, Epsilon: 0.01
Episode 155, Reward: 162.0, Epsilon: 0.01
Episode 156, Reward: 120.0, Epsilon: 0.01
Episode 157, Reward: 122.0, Epsilon: 0.01
Episode 158, Reward: 137.0, Epsilon: 0.01
Episode 159, Reward: 147.0, Epsilon: 0.01
Episode 160, Reward: 124.0, Epsilon: 0.01
Episode 161, Reward: 133.0, Epsilon: 0.01
Episode 162, Reward: 126.0, Epsilon: 0.01
Episode 163, Reward: 131.0, Epsilon: 0.01
Episode 164, Reward: 130.0, Epsilon: 0.01
Episode 165, Reward: 128.0, Epsilon: 0.01
Episode 166, Reward: 123.0, Epsilon: 0.01
Episode 167, Reward: 121.0, Epsilon: 0.01
Episode 168, Reward: 127.0, Epsilon: 0.01
Episode 169, Reward: 135.0, Epsilon: 0.01
Episode 170, Reward: 125.0, Epsilon: 0.01
Episode 171, Reward: 131.0, Epsilon: 0.01
Episode 172, Reward: 145.0, Epsilon: 0.01
Episode 173, Reward: 130.0, Epsilon: 0.01
Episode 174, Reward: 126.0, Epsilon: 0.01
Episode 175, Reward: 141.0, Epsilon: 0.01
Episode 176, Reward: 148.0, Epsilon: 0.01
Episode 177, Reward: 138.0, Epsilon: 0.01
Episode 178, Reward: 155.0, Epsilon: 0.01
Episode 179, Reward: 150.0, Epsilon: 0.01
Episode 180, Reward: 148.0, Epsilon: 0.01
Episode 181, Reward: 247.0, Epsilon: 0.01
Episode 182, Reward: 259.0, Epsilon: 0.01
Episode 183, Reward: 199.0, Epsilon: 0.01
Episode 184, Reward: 310.0, Epsilon: 0.01
Episode 185, Reward: 24.0, Epsilon: 0.01
Episode 186, Reward: 156.0, Epsilon: 0.01
Episode 187, Reward: 231.0, Epsilon: 0.01
Episode 188, Reward: 185.0, Epsilon: 0.01
Episode 189, Reward: 174.0, Epsilon: 0.01
Episode 190, Reward: 41.0, Epsilon: 0.01
Episode 191, Reward: 39.0, Epsilon: 0.01
Episode 192, Reward: 39.0, Epsilon: 0.01
Episode 193, Reward: 36.0, Epsilon: 0.01
Episode 194, Reward: 51.0, Epsilon: 0.01
Episode 195, Reward: 15.0, Epsilon: 0.01
Episode 196, Reward: 39.0, Epsilon: 0.01
Episode 197, Reward: 53.0, Epsilon: 0.01
Episode 198, Reward: 10.0, Epsilon: 0.01
Episode 199, Reward: 19.0, Epsilon: 0.01
Episode 200, Reward: 10.0, Epsilon: 0.01
Episode 201, Reward: 23.0, Epsilon: 0.01
Episode 202, Reward: 13.0, Epsilon: 0.01
Episode 203, Reward: 23.0, Epsilon: 0.01
Episode 204, Reward: 38.0, Epsilon: 0.01
Episode 205, Reward: 59.0, Epsilon: 0.01
Episode 206, Reward: 13.0, Epsilon: 0.01
Episode 207, Reward: 32.0, Epsilon: 0.01
Episode 208, Reward: 22.0, Epsilon: 0.01
Episode 209, Reward: 44.0, Epsilon: 0.01
Episode 210, Reward: 51.0, Epsilon: 0.01
Episode 211, Reward: 76.0, Epsilon: 0.01
Episode 212, Reward: 139.0, Epsilon: 0.01
Episode 213, Reward: 138.0, Epsilon: 0.01
Episode 214, Reward: 140.0, Epsilon: 0.01
Episode 215, Reward: 137.0, Epsilon: 0.01
Episode 216, Reward: 126.0, Epsilon: 0.01
Episode 217, Reward: 127.0, Epsilon: 0.01
Episode 218, Reward: 148.0, Epsilon: 0.01
Episode 219, Reward: 166.0, Epsilon: 0.01
Episode 220, Reward: 145.0, Epsilon: 0.01
Episode 221, Reward: 108.0, Epsilon: 0.01
Episode 222, Reward: 131.0, Epsilon: 0.01
Episode 223, Reward: 128.0, Epsilon: 0.01
Episode 224, Reward: 120.0, Epsilon: 0.01
Episode 225, Reward: 124.0, Epsilon: 0.01
Episode 226, Reward: 124.0, Epsilon: 0.01
Episode 227, Reward: 136.0, Epsilon: 0.01
Episode 228, Reward: 131.0, Epsilon: 0.01
Episode 229, Reward: 118.0, Epsilon: 0.01
Episode 230, Reward: 139.0, Epsilon: 0.01
Episode 231, Reward: 131.0, Epsilon: 0.01
Episode 232, Reward: 132.0, Epsilon: 0.01
Episode 233, Reward: 127.0, Epsilon: 0.01
Episode 234, Reward: 135.0, Epsilon: 0.01
Episode 235, Reward: 131.0, Epsilon: 0.01
Episode 236, Reward: 139.0, Epsilon: 0.01
Episode 237, Reward: 144.0, Epsilon: 0.01
Episode 238, Reward: 129.0, Epsilon: 0.01
Episode 239, Reward: 148.0, Epsilon: 0.01
Episode 240, Reward: 142.0, Epsilon: 0.01
Episode 241, Reward: 144.0, Epsilon: 0.01
Episode 242, Reward: 169.0, Epsilon: 0.01
Episode 243, Reward: 177.0, Epsilon: 0.01
Episode 244, Reward: 175.0, Epsilon: 0.01
Episode 245, Reward: 148.0, Epsilon: 0.01
Episode 246, Reward: 187.0, Epsilon: 0.01
Episode 247, Reward: 158.0, Epsilon: 0.01
Episode 248, Reward: 172.0, Epsilon: 0.01
Episode 249, Reward: 191.0, Epsilon: 0.01
Episode 250, Reward: 165.0, Epsilon: 0.01
Episode 251, Reward: 145.0, Epsilon: 0.01
Episode 252, Reward: 170.0, Epsilon: 0.01
Episode 253, Reward: 167.0, Epsilon: 0.01
Episode 254, Reward: 171.0, Epsilon: 0.01
Episode 255, Reward: 186.0, Epsilon: 0.01
Episode 256, Reward: 215.0, Epsilon: 0.01
Episode 257, Reward: 228.0, Epsilon: 0.01
Episode 258, Reward: 172.0, Epsilon: 0.01
Episode 259, Reward: 154.0, Epsilon: 0.01
Episode 260, Reward: 182.0, Epsilon: 0.01
Episode 261, Reward: 304.0, Epsilon: 0.01
Episode 262, Reward: 251.0, Epsilon: 0.01
Episode 263, Reward: 197.0, Epsilon: 0.01
Episode 264, Reward: 189.0, Epsilon: 0.01
Episode 265, Reward: 196.0, Epsilon: 0.01
Episode 266, Reward: 287.0, Epsilon: 0.01
Episode 267, Reward: 174.0, Epsilon: 0.01
Episode 268, Reward: 192.0, Epsilon: 0.01
Episode 269, Reward: 177.0, Epsilon: 0.01
Episode 270, Reward: 173.0, Epsilon: 0.01
Episode 271, Reward: 225.0, Epsilon: 0.01
Episode 272, Reward: 194.0, Epsilon: 0.01
Episode 273, Reward: 163.0, Epsilon: 0.01
Episode 274, Reward: 206.0, Epsilon: 0.01
Episode 275, Reward: 198.0, Epsilon: 0.01
Episode 276, Reward: 211.0, Epsilon: 0.01
Episode 277, Reward: 172.0, Epsilon: 0.01
Episode 278, Reward: 180.0, Epsilon: 0.01
Episode 279, Reward: 245.0, Epsilon: 0.01
Episode 280, Reward: 189.0, Epsilon: 0.01
Episode 281, Reward: 220.0, Epsilon: 0.01
Episode 282, Reward: 214.0, Epsilon: 0.01
Episode 283, Reward: 222.0, Epsilon: 0.01
Episode 284, Reward: 250.0, Epsilon: 0.01
Episode 285, Reward: 274.0, Epsilon: 0.01
Episode 286, Reward: 181.0, Epsilon: 0.01
Episode 287, Reward: 219.0, Epsilon: 0.01
Episode 288, Reward: 195.0, Epsilon: 0.01
Episode 289, Reward: 157.0, Epsilon: 0.01
Episode 290, Reward: 195.0, Epsilon: 0.01
Episode 291, Reward: 192.0, Epsilon: 0.01
Episode 292, Reward: 153.0, Epsilon: 0.01
Episode 293, Reward: 168.0, Epsilon: 0.01
Episode 294, Reward: 201.0, Epsilon: 0.01
Episode 295, Reward: 298.0, Epsilon: 0.01
Episode 296, Reward: 252.0, Epsilon: 0.01
Episode 297, Reward: 149.0, Epsilon: 0.01
Episode 298, Reward: 146.0, Epsilon: 0.01
Episode 299, Reward: 125.0, Epsilon: 0.01
Episode 300, Reward: 147.0, Epsilon: 0.01
Episode 301, Reward: 131.0, Epsilon: 0.01
Episode 302, Reward: 141.0, Epsilon: 0.01
Episode 303, Reward: 138.0, Epsilon: 0.01
Episode 304, Reward: 144.0, Epsilon: 0.01
Episode 305, Reward: 138.0, Epsilon: 0.01
Episode 306, Reward: 133.0, Epsilon: 0.01
Episode 307, Reward: 130.0, Epsilon: 0.01
Episode 308, Reward: 143.0, Epsilon: 0.01
Episode 309, Reward: 130.0, Epsilon: 0.01
Episode 310, Reward: 130.0, Epsilon: 0.01
Episode 311, Reward: 207.0, Epsilon: 0.01
Episode 312, Reward: 178.0, Epsilon: 0.01
Episode 313, Reward: 439.0, Epsilon: 0.01
Episode 314, Reward: 179.0, Epsilon: 0.01
Episode 315, Reward: 152.0, Epsilon: 0.01
Episode 316, Reward: 149.0, Epsilon: 0.01
Episode 317, Reward: 188.0, Epsilon: 0.01
Episode 318, Reward: 140.0, Epsilon: 0.01
Episode 319, Reward: 419.0, Epsilon: 0.01
Episode 320, Reward: 173.0, Epsilon: 0.01
Episode 321, Reward: 357.0, Epsilon: 0.01
Episode 322, Reward: 137.0, Epsilon: 0.01
Episode 323, Reward: 247.0, Epsilon: 0.01
Episode 324, Reward: 260.0, Epsilon: 0.01
Episode 325, Reward: 259.0, Epsilon: 0.01
Episode 326, Reward: 311.0, Epsilon: 0.01
Episode 327, Reward: 181.0, Epsilon: 0.01
Episode 328, Reward: 157.0, Epsilon: 0.01
Episode 329, Reward: 161.0, Epsilon: 0.01
Episode 330, Reward: 257.0, Epsilon: 0.01
Episode 331, Reward: 179.0, Epsilon: 0.01
Episode 332, Reward: 143.0, Epsilon: 0.01
Episode 333, Reward: 178.0, Epsilon: 0.01
Episode 334, Reward: 154.0, Epsilon: 0.01
Episode 335, Reward: 170.0, Epsilon: 0.01
Episode 336, Reward: 245.0, Epsilon: 0.01
Episode 337, Reward: 216.0, Epsilon: 0.01
Episode 338, Reward: 189.0, Epsilon: 0.01
Episode 339, Reward: 154.0, Epsilon: 0.01
Episode 340, Reward: 158.0, Epsilon: 0.01
Episode 341, Reward: 193.0, Epsilon: 0.01
Episode 342, Reward: 245.0, Epsilon: 0.01
Episode 343, Reward: 153.0, Epsilon: 0.01
Episode 344, Reward: 199.0, Epsilon: 0.01
Episode 345, Reward: 169.0, Epsilon: 0.01
Episode 346, Reward: 172.0, Epsilon: 0.01
Episode 347, Reward: 259.0, Epsilon: 0.01
Episode 348, Reward: 190.0, Epsilon: 0.01
Episode 349, Reward: 177.0, Epsilon: 0.01
Episode 350, Reward: 196.0, Epsilon: 0.01
Episode 351, Reward: 257.0, Epsilon: 0.01
Episode 352, Reward: 187.0, Epsilon: 0.01
Episode 353, Reward: 240.0, Epsilon: 0.01
Episode 354, Reward: 154.0, Epsilon: 0.01
Episode 355, Reward: 208.0, Epsilon: 0.01
Episode 356, Reward: 188.0, Epsilon: 0.01
Episode 357, Reward: 167.0, Epsilon: 0.01
Episode 358, Reward: 172.0, Epsilon: 0.01
Episode 359, Reward: 175.0, Epsilon: 0.01
Episode 360, Reward: 145.0, Epsilon: 0.01
Episode 361, Reward: 163.0, Epsilon: 0.01
Episode 362, Reward: 190.0, Epsilon: 0.01
Episode 363, Reward: 153.0, Epsilon: 0.01
Episode 364, Reward: 137.0, Epsilon: 0.01
Episode 365, Reward: 151.0, Epsilon: 0.01
Episode 366, Reward: 144.0, Epsilon: 0.01
Episode 367, Reward: 145.0, Epsilon: 0.01
Episode 368, Reward: 163.0, Epsilon: 0.01
Episode 369, Reward: 190.0, Epsilon: 0.01
Episode 370, Reward: 167.0, Epsilon: 0.01
Episode 371, Reward: 158.0, Epsilon: 0.01
Episode 372, Reward: 131.0, Epsilon: 0.01
Episode 373, Reward: 130.0, Epsilon: 0.01
Episode 374, Reward: 171.0, Epsilon: 0.01
Episode 375, Reward: 158.0, Epsilon: 0.01
Episode 376, Reward: 148.0, Epsilon: 0.01
Episode 377, Reward: 121.0, Epsilon: 0.01
Episode 378, Reward: 128.0, Epsilon: 0.01
Episode 379, Reward: 121.0, Epsilon: 0.01
Episode 380, Reward: 231.0, Epsilon: 0.01
Episode 381, Reward: 159.0, Epsilon: 0.01
Episode 382, Reward: 136.0, Epsilon: 0.01
Episode 383, Reward: 132.0, Epsilon: 0.01
Episode 384, Reward: 146.0, Epsilon: 0.01
Episode 385, Reward: 160.0, Epsilon: 0.01
Episode 386, Reward: 158.0, Epsilon: 0.01
Episode 387, Reward: 131.0, Epsilon: 0.01
Episode 388, Reward: 151.0, Epsilon: 0.01
Episode 389, Reward: 143.0, Epsilon: 0.01
Episode 390, Reward: 138.0, Epsilon: 0.01
Episode 391, Reward: 125.0, Epsilon: 0.01
Episode 392, Reward: 147.0, Epsilon: 0.01
Episode 393, Reward: 155.0, Epsilon: 0.01
Episode 394, Reward: 143.0, Epsilon: 0.01
Episode 395, Reward: 130.0, Epsilon: 0.01
Episode 396, Reward: 144.0, Epsilon: 0.01
Episode 397, Reward: 155.0, Epsilon: 0.01
Episode 398, Reward: 134.0, Epsilon: 0.01
Episode 399, Reward: 154.0, Epsilon: 0.01
Episode 400, Reward: 131.0, Epsilon: 0.01
Episode 401, Reward: 174.0, Epsilon: 0.01
Episode 402, Reward: 132.0, Epsilon: 0.01
Episode 403, Reward: 157.0, Epsilon: 0.01
Episode 404, Reward: 125.0, Epsilon: 0.01
Episode 405, Reward: 176.0, Epsilon: 0.01
Episode 406, Reward: 156.0, Epsilon: 0.01
Episode 407, Reward: 152.0, Epsilon: 0.01
Episode 408, Reward: 156.0, Epsilon: 0.01
Episode 409, Reward: 122.0, Epsilon: 0.01
Episode 410, Reward: 121.0, Epsilon: 0.01
Episode 411, Reward: 124.0, Epsilon: 0.01
Episode 412, Reward: 126.0, Epsilon: 0.01
Episode 413, Reward: 127.0, Epsilon: 0.01
Episode 414, Reward: 149.0, Epsilon: 0.01
Episode 415, Reward: 155.0, Epsilon: 0.01
Episode 416, Reward: 142.0, Epsilon: 0.01
Episode 417, Reward: 148.0, Epsilon: 0.01
Episode 418, Reward: 155.0, Epsilon: 0.01
Episode 419, Reward: 140.0, Epsilon: 0.01
Episode 420, Reward: 84.0, Epsilon: 0.01
Episode 421, Reward: 42.0, Epsilon: 0.01
Episode 422, Reward: 51.0, Epsilon: 0.01
Episode 423, Reward: 110.0, Epsilon: 0.01
Episode 424, Reward: 133.0, Epsilon: 0.01
Episode 425, Reward: 165.0, Epsilon: 0.01
Episode 426, Reward: 136.0, Epsilon: 0.01
Episode 427, Reward: 109.0, Epsilon: 0.01
Episode 428, Reward: 126.0, Epsilon: 0.01
Episode 429, Reward: 115.0, Epsilon: 0.01
Episode 430, Reward: 157.0, Epsilon: 0.01
Episode 431, Reward: 131.0, Epsilon: 0.01
Episode 432, Reward: 203.0, Epsilon: 0.01
Episode 433, Reward: 153.0, Epsilon: 0.01
Episode 434, Reward: 129.0, Epsilon: 0.01
Episode 435, Reward: 114.0, Epsilon: 0.01
Episode 436, Reward: 108.0, Epsilon: 0.01
Episode 437, Reward: 108.0, Epsilon: 0.01
Episode 438, Reward: 115.0, Epsilon: 0.01
Episode 439, Reward: 123.0, Epsilon: 0.01
Episode 440, Reward: 205.0, Epsilon: 0.01
Episode 441, Reward: 107.0, Epsilon: 0.01
Episode 442, Reward: 104.0, Epsilon: 0.01
Episode 443, Reward: 156.0, Epsilon: 0.01
Episode 444, Reward: 225.0, Epsilon: 0.01
Episode 445, Reward: 106.0, Epsilon: 0.01
Episode 446, Reward: 223.0, Epsilon: 0.01
Episode 447, Reward: 319.0, Epsilon: 0.01
Episode 448, Reward: 160.0, Epsilon: 0.01
Episode 449, Reward: 371.0, Epsilon: 0.01
Episode 450, Reward: 310.0, Epsilon: 0.01
Episode 451, Reward: 111.0, Epsilon: 0.01
Episode 452, Reward: 116.0, Epsilon: 0.01
Episode 453, Reward: 121.0, Epsilon: 0.01
Episode 454, Reward: 123.0, Epsilon: 0.01
Episode 455, Reward: 110.0, Epsilon: 0.01
Episode 456, Reward: 110.0, Epsilon: 0.01
Episode 457, Reward: 109.0, Epsilon: 0.01
Episode 458, Reward: 132.0, Epsilon: 0.01
Episode 459, Reward: 111.0, Epsilon: 0.01
Episode 460, Reward: 214.0, Epsilon: 0.01
Episode 461, Reward: 165.0, Epsilon: 0.01
Episode 462, Reward: 235.0, Epsilon: 0.01
Episode 463, Reward: 124.0, Epsilon: 0.01
Episode 464, Reward: 158.0, Epsilon: 0.01
Episode 465, Reward: 182.0, Epsilon: 0.01
Episode 466, Reward: 126.0, Epsilon: 0.01
Episode 467, Reward: 136.0, Epsilon: 0.01
Episode 468, Reward: 134.0, Epsilon: 0.01
Episode 469, Reward: 185.0, Epsilon: 0.01
Episode 470, Reward: 194.0, Epsilon: 0.01
Episode 471, Reward: 122.0, Epsilon: 0.01
Episode 472, Reward: 130.0, Epsilon: 0.01
Episode 473, Reward: 214.0, Epsilon: 0.01
Episode 474, Reward: 134.0, Epsilon: 0.01
Episode 475, Reward: 322.0, Epsilon: 0.01
Episode 476, Reward: 151.0, Epsilon: 0.01
Episode 477, Reward: 123.0, Epsilon: 0.01
Episode 478, Reward: 112.0, Epsilon: 0.01
Episode 479, Reward: 227.0, Epsilon: 0.01
Episode 480, Reward: 174.0, Epsilon: 0.01
Episode 481, Reward: 257.0, Epsilon: 0.01
Episode 482, Reward: 251.0, Epsilon: 0.01
Episode 483, Reward: 261.0, Epsilon: 0.01
Episode 484, Reward: 500.0, Epsilon: 0.01
Episode 485, Reward: 230.0, Epsilon: 0.01
Episode 486, Reward: 500.0, Epsilon: 0.01
Episode 487, Reward: 212.0, Epsilon: 0.01
Episode 488, Reward: 205.0, Epsilon: 0.01
Episode 489, Reward: 260.0, Epsilon: 0.01
Episode 490, Reward: 495.0, Epsilon: 0.01
Episode 491, Reward: 215.0, Epsilon: 0.01
Episode 492, Reward: 145.0, Epsilon: 0.01
Episode 493, Reward: 173.0, Epsilon: 0.01
Episode 494, Reward: 235.0, Epsilon: 0.01
Episode 495, Reward: 247.0, Epsilon: 0.01
Episode 496, Reward: 205.0, Epsilon: 0.01
Episode 497, Reward: 420.0, Epsilon: 0.01
Episode 498, Reward: 168.0, Epsilon: 0.01
Episode 499, Reward: 347.0, Epsilon: 0.01
```

In [7]:
```
#plotting
plt.plot(reward_history)
plt.xlabel("Episode")
plt.ylabel("Reward")
plt.title("DQN with Double and Dueling Network on CartPole")
plt.grid()
plt.show()
```



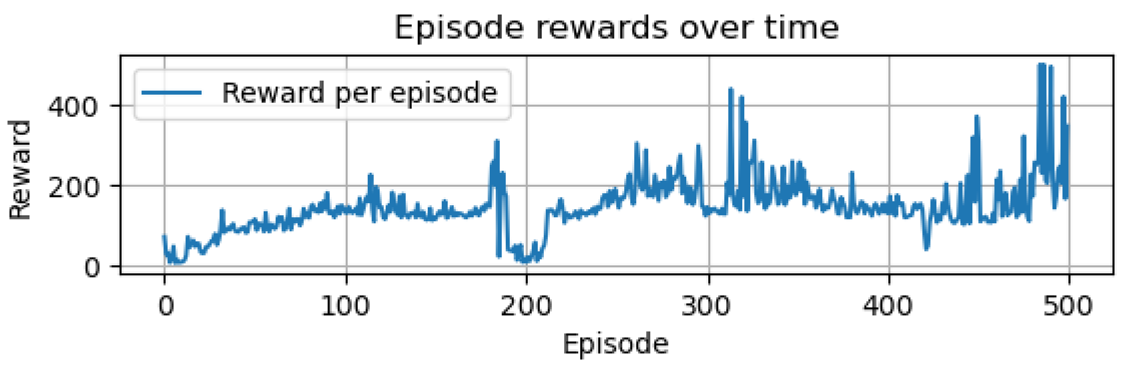
In [8]:
```
import numpy as np
import matplotlib.pyplot as plt

rewards = reward_history
```

```
window = 20

moving_avg = np.convolve(rewards, np.ones(window) / window, mode='valid')

reward_variance = [
    np.var(rewards[max(0, i - window):i + 1])
    for i in range(len(rewards))
]

plt.figure(figsize=(14, 10))
```

Out[8]:
```
<Figure size 1400x1000 with 0 Axes>
```
```
<Figure size 1400x1000 with 0 Axes>
```

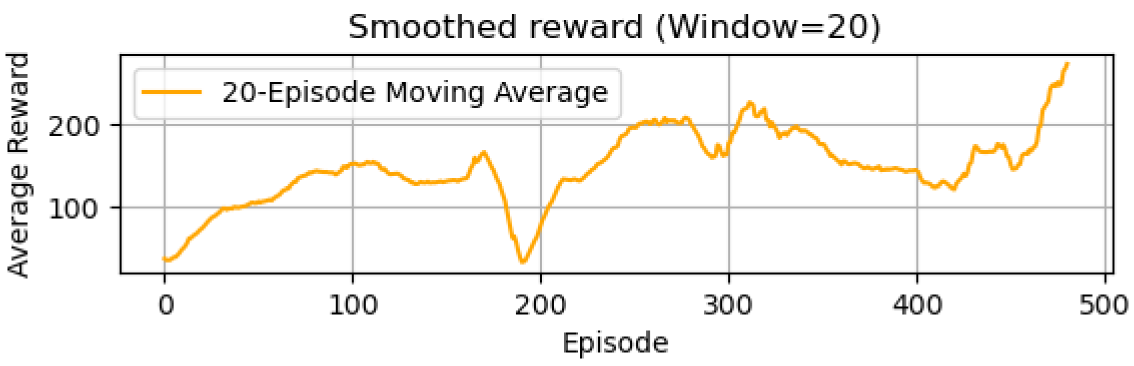
In [12]:
```
# Raw rewards
plt.subplot(3, 1, 1)
plt.plot(rewards, label='Reward per episode')
plt.title("Episode rewards over time")
plt.xlabel("Episode")
plt.ylabel("Reward")
plt.grid(True)
plt.legend()
```

Out[12]:
```
<matplotlib.legend.Legend at 0x1666fd760>
```

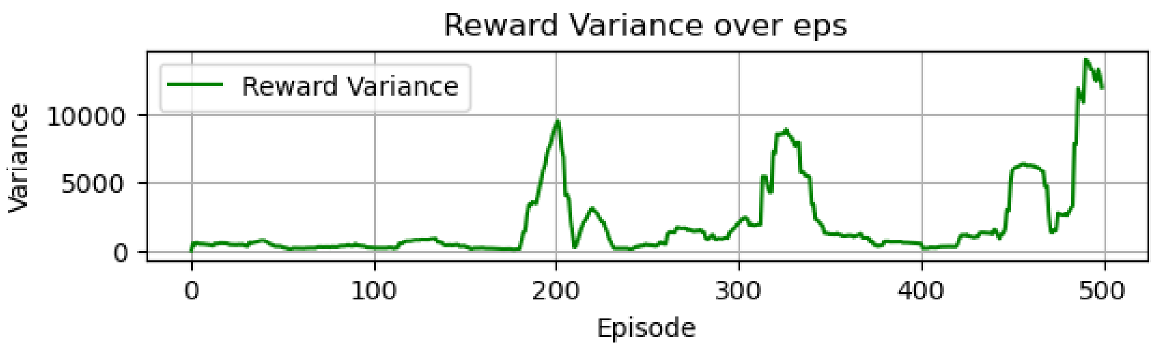


In [13]:
```
#Moving average
plt.subplot(3, 1, 2)
plt.plot(moving_avg, color='orange', label=f"{window}-Episode Moving Aver
plt.title(f"Smoothed reward (Window={window})")
plt.xlabel("Episode")
plt.ylabel("Average Reward")
plt.grid(True)
plt.legend()
```

Out[13]:
```
<matplotlib.legend.Legend at 0x1666ad310>
```



```
In [14]: # reward variance
         plt.subplot(3, 1, 3)
         plt.plot(reward_variance, color='green', label="Reward Variance")
         plt.title("Reward Variance over eps")
         plt.xlabel("Episode")
         plt.ylabel("Variance")
         plt.grid(True)
         plt.legend()

         plt.tight_layout()
         plt.show()
```



```
In [ ]:
```

In [13]:
```python
from stable_baselines3 import PPO
import gym
import numpy as np
import matplotlib.pyplot as plt
```

In [14]:
```python
#enviroment
env = gym.make('CartPole-v1')
```

In [15]:
```python
model = PPO(
    policy='MlpPolicy',
    env=env,
    verbose=1,

)
```

```
Using cpu device
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
```

In [16]:
```python
#training
model.learn(
    total_timesteps=100_000,

)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 20.2     |
|    ep_rew_mean     | 20.2     |
| time/              |          |
|    fps             | 7309     |
|    iterations      | 1        |
|    time_elapsed    | 0        |
|    total_timesteps | 2048     |
---------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 28.3        |
|    ep_rew_mean          | 28.3        |
| time/                   |             |
|    fps                  | 5296        |
|    iterations           | 2           |
|    time_elapsed         | 0           |
|    total_timesteps      | 4096        |
| train/                  |             |
|    approx_kl            | 0.010084883 |
|    clip_fraction        | 0.133       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.685      |
|    explained_variance   | 0.000473    |
|    learning_rate        | 0.0003      |
|    loss                 | 5.23        |
|    n_updates            | 10          |
|    policy_gradient_loss | -0.0218     |
|    value_loss           | 45.9        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 36.8        |
|    ep_rew_mean          | 36.8        |
| time/                   |             |
|    fps                  | 4848        |
|    iterations           | 3           |
|    time_elapsed         | 1           |
|    total_timesteps      | 6144        |
| train/                  |             |
|    approx_kl            | 0.009573863 |
|    clip_fraction        | 0.0563      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.666      |
|    explained_variance   | 0.0859      |
|    learning_rate        | 0.0003      |
|    loss                 | 13.7        |
|    n_updates            | 20          |
|    policy_gradient_loss | -0.0156     |
|    value_loss           | 40.2        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 49.2        |
|    ep_rew_mean          | 49.2        |
| time/                   |             |
|    fps                  | 4650        |
|    iterations           | 4           |
|    time_elapsed         | 1           |
|    total_timesteps      | 8192        |
| train/                  |             |
|    approx_kl            | 0.008038716 |
|    clip_fraction        | 0.0853      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.635      |
|    explained_variance   | 0.219       |
|    learning_rate        | 0.0003      |
|    loss                 | 18.1        |
|    n_updates            | 30          |
|    policy_gradient_loss | -0.0188     |
|    value_loss           | 57.9        |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 61.5         |
|    ep_rew_mean          | 61.5         |
| time/                   |              |
|    fps                  | 4510         |
|    iterations           | 5            |
|    time_elapsed         | 2            |
|    total_timesteps      | 10240        |
| train/                  |              |
|    approx_kl            | 0.0060370853 |
|    clip_fraction        | 0.0404       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.618       |
|    explained_variance   | 0.137        |
|    learning_rate        | 0.0003       |
|    loss                 | 20.4         |
|    n_updates            | 40           |
|    policy_gradient_loss | -0.0121      |
|    value_loss           | 70           |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 77.7         |
|    ep_rew_mean          | 77.7         |
| time/                   |              |
|    fps                  | 4413         |
|    iterations           | 6            |
|    time_elapsed         | 2            |
|    total_timesteps      | 12288        |
| train/                  |              |
|    approx_kl            | 0.0111379335 |
|    clip_fraction        | 0.134        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.589       |
|    explained_variance   | 0.45         |
|    learning_rate        | 0.0003       |
|    loss                 | 31.4         |
|    n_updates            | 50           |
|    policy_gradient_loss | -0.0187      |
|    value_loss           | 62.5         |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 94.5        |
|    ep_rew_mean          | 94.5        |
| time/                   |             |
|    fps                  | 4350        |
|    iterations           | 7           |
|    time_elapsed         | 3           |
|    total_timesteps      | 14336       |
| train/                  |             |
|    approx_kl            | 0.006713805 |
|    clip_fraction        | 0.0531      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.569      |
|    explained_variance   | 0.521       |
|    learning_rate        | 0.0003      |
|    loss                 | 6.57        |
|    n_updates            | 60          |
|    policy_gradient_loss | -0.00977    |
|    value_loss           | 60.8        |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 113         |
|    ep_rew_mean          | 113         |
| time/                   |             |
|    fps                  | 4293        |
|    iterations           | 8           |
|    time_elapsed         | 3           |
|    total_timesteps      | 16384       |
| train/                  |             |
|    approx_kl            | 0.006808429 |
|    clip_fraction        | 0.067       |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.573      |
|    explained_variance   | 0.615       |
|    learning_rate        | 0.0003      |
|    loss                 | 18          |
|    n_updates            | 70          |
|    policy_gradient_loss | -0.0117     |
|    value_loss           | 51.3        |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 130          |
|    ep_rew_mean          | 130          |
| time/                   |              |
|    fps                  | 4266         |
|    iterations           | 9            |
|    time_elapsed         | 4            |
|    total_timesteps      | 18432        |
| train/                  |              |
|    approx_kl            | 0.0065910453 |
|    clip_fraction        | 0.0457       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.564       |
|    explained_variance   | 0.695        |
|    learning_rate        | 0.0003       |
|    loss                 | 18.9         |
|    n_updates            | 80           |
|    policy_gradient_loss | -0.00795     |
|    value_loss           | 43.2         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 147          |
|    ep_rew_mean          | 147          |
| time/                   |              |
|    fps                  | 4252         |
|    iterations           | 10           |
|    time_elapsed         | 4            |
|    total_timesteps      | 20480        |
| train/                  |              |
|    approx_kl            | 0.0073036794 |
|    clip_fraction        | 0.0856       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.578       |
|    explained_variance   | 0.769        |
|    learning_rate        | 0.0003       |
|    loss                 | 4.65         |
|    n_updates            | 90           |
|    policy_gradient_loss | -0.013       |
|    value_loss           | 32.7         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 162          |
|    ep_rew_mean          | 162          |
| time/                   |              |
|    fps                  | 4244         |
|    iterations           | 11           |
|    time_elapsed         | 5            |
|    total_timesteps      | 22528        |
| train/                  |              |
|    approx_kl            | 0.0023154113 |
|    clip_fraction        | 0.0111       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.552       |
|    explained_variance   | 0.501        |
|    learning_rate        | 0.0003       |
|    loss                 | 16.3         |
|    n_updates            | 100          |
|    policy_gradient_loss | -0.00333     |
|    value_loss           | 52.1         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 179          |
|    ep_rew_mean          | 179          |
| time/                   |              |
|    fps                  | 4232         |
|    iterations           | 12           |
|    time_elapsed         | 5            |
|    total_timesteps      | 24576        |
| train/                  |              |
|    approx_kl            | 0.0039658635 |
|    clip_fraction        | 0.0507       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.563       |
|    explained_variance   | 0.958        |
|    learning_rate        | 0.0003       |
|    loss                 | 0.661        |
|    n_updates            | 110          |
|    policy_gradient_loss | -0.00757     |
|    value_loss           | 9.84         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 195          |
|    ep_rew_mean          | 195          |
| time/                   |              |
|    fps                  | 4210         |
|    iterations           | 13           |
|    time_elapsed         | 6            |
|    total_timesteps      | 26624        |
| train/                  |              |
|    approx_kl            | 0.0034678157 |
|    clip_fraction        | 0.0233       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.536       |
|    explained_variance   | 0.00167      |
|    learning_rate        | 0.0003       |
|    loss                 | 4.45         |
|    n_updates            | 120          |
|    policy_gradient_loss | -0.0043      |
|    value_loss           | 66.3         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 212          |
|    ep_rew_mean          | 212          |
| time/                   |              |
|    fps                  | 4212         |
|    iterations           | 14           |
|    time_elapsed         | 6            |
|    total_timesteps      | 28672        |
| train/                  |              |
|    approx_kl            | 0.0039183805 |
|    clip_fraction        | 0.0304       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.545       |
|    explained_variance   | 0.677        |
|    learning_rate        | 0.0003       |
|    loss                 | 0.478        |
|    n_updates            | 130          |
|    policy_gradient_loss | -0.00456     |
|    value_loss           | 17.1         |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 228         |
|    ep_rew_mean          | 228         |
| time/                   |             |
|    fps                  | 4203        |
|    iterations           | 15          |
|    time_elapsed         | 7           |
|    total_timesteps      | 30720       |
| train/                  |             |
|    approx_kl            | 0.007481771 |
|    clip_fraction        | 0.0413      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.533      |
|    explained_variance   | 0.967       |
|    learning_rate        | 0.0003      |
|    loss                 | 0.375       |
|    n_updates            | 140         |
|    policy_gradient_loss | -0.00179    |
|    value_loss           | 3.3         |
-----------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 249         |
|    ep_rew_mean          | 249         |
| time/                   |             |
|    fps                  | 4197        |
|    iterations           | 16          |
|    time_elapsed         | 7           |
|    total_timesteps      | 32768       |
| train/                  |             |
|    approx_kl            | 0.004168884 |
|    clip_fraction        | 0.0502       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.53        |
|    explained_variance   | 0.972        |
|    learning_rate        | 0.0003       |
|    loss                 | 0.204        |
|    n_updates            | 150          |
|    policy_gradient_loss | -0.00797     |
|    value_loss           | 1.6          |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 266          |
|    ep_rew_mean          | 266          |
| time/                   |              |
|    fps                  | 4197         |
|    iterations           | 17           |
|    time_elapsed         | 8            |
|    total_timesteps      | 34816        |
| train/                  |              |
|    approx_kl            | 0.0017226466 |
|    clip_fraction        | 0.0178       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.53        |
|    explained_variance   | 0.843        |
|    learning_rate        | 0.0003       |
|    loss                 | 1.26         |
|    n_updates            | 160          |
|    policy_gradient_loss | 0.00036      |
|    value_loss           | 2.76         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 284          |
|    ep_rew_mean          | 284          |
| time/                   |              |
|    fps                  | 4188         |
|    iterations           | 18           |
|    time_elapsed         | 8            |
|    total_timesteps      | 36864        |
| train/                  |              |
|    approx_kl            | 0.0036730266 |
|    clip_fraction        | 0.0279       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.541       |
|    explained_variance   | -0.024       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.115        |
|    n_updates            | 170          |
|    policy_gradient_loss | -0.00184     |
|    value_loss           | 0.482        |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 301          |
|    ep_rew_mean          | 301          |
| time/                   |              |
|    fps                  | 4175         |
|    iterations           | 19           |
|    time_elapsed         | 9            |
|    total_timesteps      | 38912        |
| train/                  |              |
|    approx_kl            | 0.0013488977 |
|    clip_fraction        | 0.00186      |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.53        |
|    explained_variance   | 0.00849      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.002        |
|    n_updates            | 180          |
|    policy_gradient_loss | 0.000594     |
|    value_loss           | 0.328        |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 318         |
|    ep_rew_mean          | 318         |
| time/                   |             |
|    fps                  | 4173        |
|    iterations           | 20          |
|    time_elapsed         | 9           |
|    total_timesteps      | 40960       |
| train/                  |             |
|    approx_kl            | 0.002233721 |
|    clip_fraction        | 0.0139      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.533      |
|    explained_variance   | 0.0277      |
|    learning_rate        | 0.0003      |
|    loss                 | 0.0163      |
|    n_updates            | 190         |
|    policy_gradient_loss | -0.00126    |
|    value_loss           | 0.194       |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 334          |
|    ep_rew_mean          | 334          |
| time/                   |              |
|    fps                  | 4172         |
|    iterations           | 21           |
|    time_elapsed         | 10           |
|    total_timesteps      | 43008        |
| train/                  |              |
|    approx_kl            | 0.0023918014 |
|    clip_fraction        | 0.0134       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.512       |
|    explained_variance   | -0.019       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0138       |
|    n_updates            | 200          |
|    policy_gradient_loss | 2.14e-05     |
|    value_loss           | 0.121        |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 350          |
|    ep_rew_mean          | 350          |
| time/                   |              |
|    fps                  | 4166         |
|    iterations           | 22           |
|    time_elapsed         | 10           |
|    total_timesteps      | 45056        |
| train/                  |              |
|    approx_kl            | 0.002129831  |
|    clip_fraction        | 0.00776      |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.521       |
|    explained_variance   | 0.0615       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0129       |
|    n_updates            | 210          |
|    policy_gradient_loss | -0.000853    |
|    value_loss           | 0.081        |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 367          |
|    ep_rew_mean          | 367          |
| time/                   |              |
|    fps                  | 4166         |
|    iterations           | 23           |
|    time_elapsed         | 11           |
|    total_timesteps      | 47104        |
| train/                  |              |
|    approx_kl            | 0.007730594  |
|    clip_fraction        | 0.0656       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.494       |
|    explained_variance   | 0.048        |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0243       |
|    n_updates            | 220          |
|    policy_gradient_loss | -0.0048      |
|    value_loss           | 0.0526       |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 380          |
|    ep_rew_mean          | 380          |
| time/                   |              |
|    fps                  | 4168         |
|    iterations           | 24           |
|    time_elapsed         | 11           |
|    total_timesteps      | 49152        |
| train/                  |              |
|    approx_kl            | 0.003001992  |
|    clip_fraction        | 0.03         |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.5         |
|    explained_variance   | -0.0383      |
|    learning_rate        | 0.0003       |
|    loss                 | -0.00891     |
|    n_updates            | 230          |
|    policy_gradient_loss | -0.00208     |
|    value_loss           | 0.0322       |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 392          |
|    ep_rew_mean          | 392          |
| time/                   |              |
|    fps                  | 4168         |
|    iterations           | 25           |
|    time_elapsed         | 12           |
|    total_timesteps      | 51200        |
| train/                  |              |
|    approx_kl            | 0.0022486816 |
|    clip_fraction        | 0.029        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.502       |
|    explained_variance   | -0.0892      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00343      |
|    n_updates            | 240          |
|    policy_gradient_loss | -0.00146     |
|    value_loss           | 0.02         |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 410          |
|    ep_rew_mean          | 410          |
| time/                   |              |
|    fps                  | 4164         |
|    iterations           | 26           |
|    time_elapsed         | 12           |
|    total_timesteps      | 53248        |
| train/                  |              |
|    approx_kl            | 0.0019321006 |
|    clip_fraction        | 0.00771      |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.493       |
|    explained_variance   | -0.0246      |
|    learning_rate        | 0.0003       |
|    loss                 | -0.0178      |
|    n_updates            | 250          |
|    policy_gradient_loss | -0.00028     |
|    value_loss           | 0.0129       |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 423          |
|    ep_rew_mean          | 423          |
| time/                   |              |
|    fps                  | 4163         |
|    iterations           | 27           |
|    time_elapsed         | 13           |
|    total_timesteps      | 55296        |
| train/                  |              |
|    approx_kl            | 0.0066883126 |
|    clip_fraction        | 0.0636       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.48        |
|    explained_variance   | 0.0117       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.0124      |
|    n_updates            | 260          |
|    policy_gradient_loss | -0.00227     |
|    value_loss           | 0.00886      |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 434          |
|    ep_rew_mean          | 434          |
| time/                   |              |
|    fps                  | 4162         |
|    iterations           | 28           |
|    time_elapsed         | 13           |
|    total_timesteps      | 57344        |
| train/                  |              |
|    approx_kl            | 0.0022244358 |
|    clip_fraction        | 0.02         |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.472       |
|    explained_variance   | -0.0409      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00435      |
|    n_updates            | 270          |
|    policy_gradient_loss | -0.00117     |
|    value_loss           | 0.00557      |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 443         |
|    ep_rew_mean          | 443         |
| time/                   |             |
|    fps                  | 4162        |
|    iterations           | 29          |
|    time_elapsed         | 14          |
|    total_timesteps      | 59392       |
| train/                  |             |
|    approx_kl            | 0.006202965 |
|    clip_fraction        | 0.0495      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.468      |
|    explained_variance   | -0.00115    |
|    learning_rate        | 0.0003      |
|    loss                 | 0.00205     |
|    n_updates            | 280         |
|    policy_gradient_loss | -0.00177    |
|    value_loss           | 0.00383     |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 452          |
|    ep_rew_mean          | 452          |
| time/                   |              |
|    fps                  | 4163         |
|    iterations           | 30           |
|    time_elapsed         | 14           |
|    total_timesteps      | 61440        |
| train/                  |              |
|    approx_kl            | 0.0025365464 |
|    clip_fraction        | 0.024        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.469       |
|    explained_variance   | 0.127        |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00404      |
|    n_updates            | 290          |
|    policy_gradient_loss | -0.000495    |
|    value_loss           | 0.00244      |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 463          |
|    ep_rew_mean          | 463          |
| time/                   |              |
|    fps                  | 4164         |
|    iterations           | 31           |
|    time_elapsed         | 15           |
|    total_timesteps      | 63488        |
| train/                  |              |
|    approx_kl            | 0.0036244143 |
|    clip_fraction        | 0.0289       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.46        |
|    explained_variance   | 0.0509       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.027       |
|    n_updates            | 300          |
|    policy_gradient_loss | -0.00141     |
|    value_loss           | 0.00163      |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 469          |
|    ep_rew_mean          | 469          |
| time/                   |              |
|    fps                  | 4165         |
|    iterations           | 32           |
|    time_elapsed         | 15           |
|    total_timesteps      | 65536        |
| train/                  |              |
|    approx_kl            | 0.0075699445 |
|    clip_fraction        | 0.0403       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.47        |
|    explained_variance   | 0.0378       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.0017      |
|    n_updates            | 310          |
|    policy_gradient_loss | -0.00177     |
|    value_loss           | 0.00111      |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 479          |
|    ep_rew_mean          | 479          |
| time/                   |              |
|    fps                  | 4166         |
|    iterations           | 33           |
|    time_elapsed         | 16           |
|    total_timesteps      | 67584        |
| train/                  |              |
|    approx_kl            | 0.0022079456 |
|    clip_fraction        | 0.0132       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.491       |
|    explained_variance   | -0.099       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.00181     |
|    n_updates            | 320          |
|    policy_gradient_loss | 0.000232     |
|    value_loss           | 0.000792     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 483          |
|    ep_rew_mean          | 483          |
| time/                   |              |
|    fps                  | 4167         |
|    iterations           | 34           |
|    time_elapsed         | 16           |
|    total_timesteps      | 69632        |
| train/                  |              |
|    approx_kl            | 0.0035761064 |
|    clip_fraction        | 0.0188       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.485       |
|    explained_variance   | -0.0582      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00513      |
|    n_updates            | 330          |
|    policy_gradient_loss | -4.28e-05    |
|    value_loss           | 0.000548     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 488          |
|    ep_rew_mean          | 488          |
| time/                   |              |
|    fps                  | 4168         |
|    iterations           | 35           |
|    time_elapsed         | 17           |
|    total_timesteps      | 71680        |
| train/                  |              |
|    approx_kl            | 0.0058300495 |
|    clip_fraction        | 0.0609       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.469       |
|    explained_variance   | -0.0779      |
|    learning_rate        | 0.0003       |
|    loss                 | -0.00304     |
|    n_updates            | 340          |
|    policy_gradient_loss | -0.00285     |
|    value_loss           | 0.000391     |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 491         |
|    ep_rew_mean          | 491         |
| time/                   |             |
|    fps                  | 4170        |
|    iterations           | 36          |
|    time_elapsed         | 17          |
|    total_timesteps      | 73728       |
| train/                  |             |
|    approx_kl            | 0.003982329 |
|    clip_fraction        | 0.0282      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.481      |
|    explained_variance   | 0.0281      |
|    learning_rate        | 0.0003      |
|    loss                 | -0.000474   |
|    n_updates            | 350         |
|    policy_gradient_loss | -0.000718   |
|    value_loss           | 0.000273    |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 496          |
|    ep_rew_mean          | 496          |
| time/                   |              |
|    fps                  | 4171         |
|    iterations           | 37           |
|    time_elapsed         | 18           |
|    total_timesteps      | 75776        |
| train/                  |              |
|    approx_kl            | 0.0053857984 |
|    clip_fraction        | 0.0582       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.472       |
|    explained_variance   | 0.00826      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.000567     |
|    n_updates            | 360          |
|    policy_gradient_loss | -0.00395     |
|    value_loss           | 0.000194     |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 499         |
|    ep_rew_mean          | 499         |
| time/                   |             |
|    fps                  | 4174        |
|    iterations           | 38          |
|    time_elapsed         | 18          |
|    total_timesteps      | 77824       |
| train/                  |             |
|    approx_kl            | 0.007361267 |
|    clip_fraction        | 0.0954      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.458      |
|    explained_variance   | 0.0361      |
|    learning_rate        | 0.0003      |
|    loss                 | -0.00938    |
|    n_updates            | 370         |
|    policy_gradient_loss | -0.0051     |
|    value_loss           | 0.000142    |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 499          |
|    ep_rew_mean          | 499          |
| time/                   |              |
|    fps                  | 4174         |
|    iterations           | 39           |
|    time_elapsed         | 19           |
|    total_timesteps      | 79872        |
| train/                  |              |
|    approx_kl            | 0.0031702667 |
|    clip_fraction        | 0.0171       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.446       |
|    explained_variance   | -0.254       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00296      |
|    n_updates            | 380          |
|    policy_gradient_loss | -0.00179     |
|    value_loss           | 0.000109     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4173         |
|    iterations           | 40           |
|    time_elapsed         | 19           |
|    total_timesteps      | 81920        |
| train/                  |              |
|    approx_kl            | 0.0012967444 |
|    clip_fraction        | 0.0138       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.433       |
|    explained_variance   | -0.0899      |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0058       |
|    n_updates            | 390          |
|    policy_gradient_loss | -0.0012      |
|    value_loss           | 7.14e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4173         |
|    iterations           | 41           |
|    time_elapsed         | 20           |
|    total_timesteps      | 83968        |
| train/                  |              |
|    approx_kl            | 0.0039352803 |
|    clip_fraction        | 0.0365       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.408       |
|    explained_variance   | -0.118       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.00578     |
|    n_updates            | 400          |
|    policy_gradient_loss | -0.00297     |
|    value_loss           | 6.02e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4174         |
|    iterations           | 42           |
|    time_elapsed         | 20           |
|    total_timesteps      | 86016        |
| train/                  |              |
|    approx_kl            | 0.0044813827 |
|    clip_fraction        | 0.042        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.398       |
|    explained_variance   | -0.104       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.0196      |
|    n_updates            | 410          |
|    policy_gradient_loss | -0.00404     |
|    value_loss           | 3.67e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4175         |
|    iterations           | 43           |
|    time_elapsed         | 21           |
|    total_timesteps      | 88064        |
| train/                  |              |
|    approx_kl            | 0.0049104686 |
|    clip_fraction        | 0.044        |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.406       |
|    explained_variance   | -0.092       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0043       |
|    n_updates            | 420          |
|    policy_gradient_loss | -0.00145     |
|    value_loss           | 2.45e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4175         |
|    iterations           | 44           |
|    time_elapsed         | 21           |
|    total_timesteps      | 90112        |
| train/                  |              |
|    approx_kl            | 0.0014221703 |
|    clip_fraction        | 0.00962      |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.405       |
|    explained_variance   | -0.283       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.0143       |
|    n_updates            | 430          |
|    policy_gradient_loss | -0.00119     |
|    value_loss           | 2.12e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4176         |
|    iterations           | 45           |
|    time_elapsed         | 22           |
|    total_timesteps      | 92160        |
| train/                  |              |
|    approx_kl            | 0.0075518903 |
|    clip_fraction        | 0.0676       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.373       |
|    explained_variance   | -0.28        |
|    learning_rate        | 0.0003       |
|    loss                 | -0.000583    |
|    n_updates            | 440          |
|    policy_gradient_loss | -0.00556     |
|    value_loss           | 1.54e-05     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4177         |
|    iterations           | 46           |
|    time_elapsed         | 22           |
|    total_timesteps      | 94208        |
| train/                  |              |
|    approx_kl            | 0.0037326524 |
|    clip_fraction        | 0.0277       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.393       |
|    explained_variance   | -0.257       |
|    learning_rate        | 0.0003       |
|    loss                 | -0.00892     |
|    n_updates            | 450          |
|    policy_gradient_loss | -0.00249     |
|    value_loss           | 1.17e-05     |
------------------------------------------
-----------------------------------------
| rollout/                |             |
|    ep_len_mean          | 500         |
|    ep_rew_mean          | 500         |
| time/                   |             |
|    fps                  | 4177        |
|    iterations           | 47          |
|    time_elapsed         | 23          |
|    total_timesteps      | 96256       |
| train/                  |             |
|    approx_kl            | 0.007448323 |
|    clip_fraction        | 0.0695      |
|    clip_range           | 0.2         |
|    entropy_loss         | -0.386      |
|    explained_variance   | -0.511      |
|    learning_rate        | 0.0003      |
|    loss                 | -0.000411   |
|    n_updates            | 460         |
|    policy_gradient_loss | -0.00209    |
|    value_loss           | 8.11e-06    |
-----------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4177         |
|    iterations           | 48           |
|    time_elapsed         | 23           |
|    total_timesteps      | 98304        |
| train/                  |              |
|    approx_kl            | 0.0007008846 |
|    clip_fraction        | 0.00703      |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.383       |
|    explained_variance   | -0.464       |
|    learning_rate        | 0.0003       |
|    loss                 | 0.00388      |
|    n_updates            | 470          |
|    policy_gradient_loss | 9.96e-05     |
|    value_loss           | 8.61e-06     |
------------------------------------------
------------------------------------------
| rollout/                |              |
|    ep_len_mean          | 500          |
|    ep_rew_mean          | 500          |
| time/                   |              |
|    fps                  | 4177         |
|    iterations           | 49           |
|    time_elapsed         | 24           |
|    total_timesteps      | 100352       |
| train/                  |              |
|    approx_kl            | 0.0041607255 |
|    clip_fraction        | 0.0556       |
|    clip_range           | 0.2          |
|    entropy_loss         | -0.378       |
|    explained_variance   | -0.86        |
|    learning_rate        | 0.0003       |
|    loss                 | -0.0197      |
|    n_updates            | 480          |
|    policy_gradient_loss | -0.00277     |
|    value_loss           | 5.4e-06      |
------------------------------------------
```

Out[16]:
```
<stable_baselines3.ppo.ppo.PPO at 0x2990e0cb0>
```

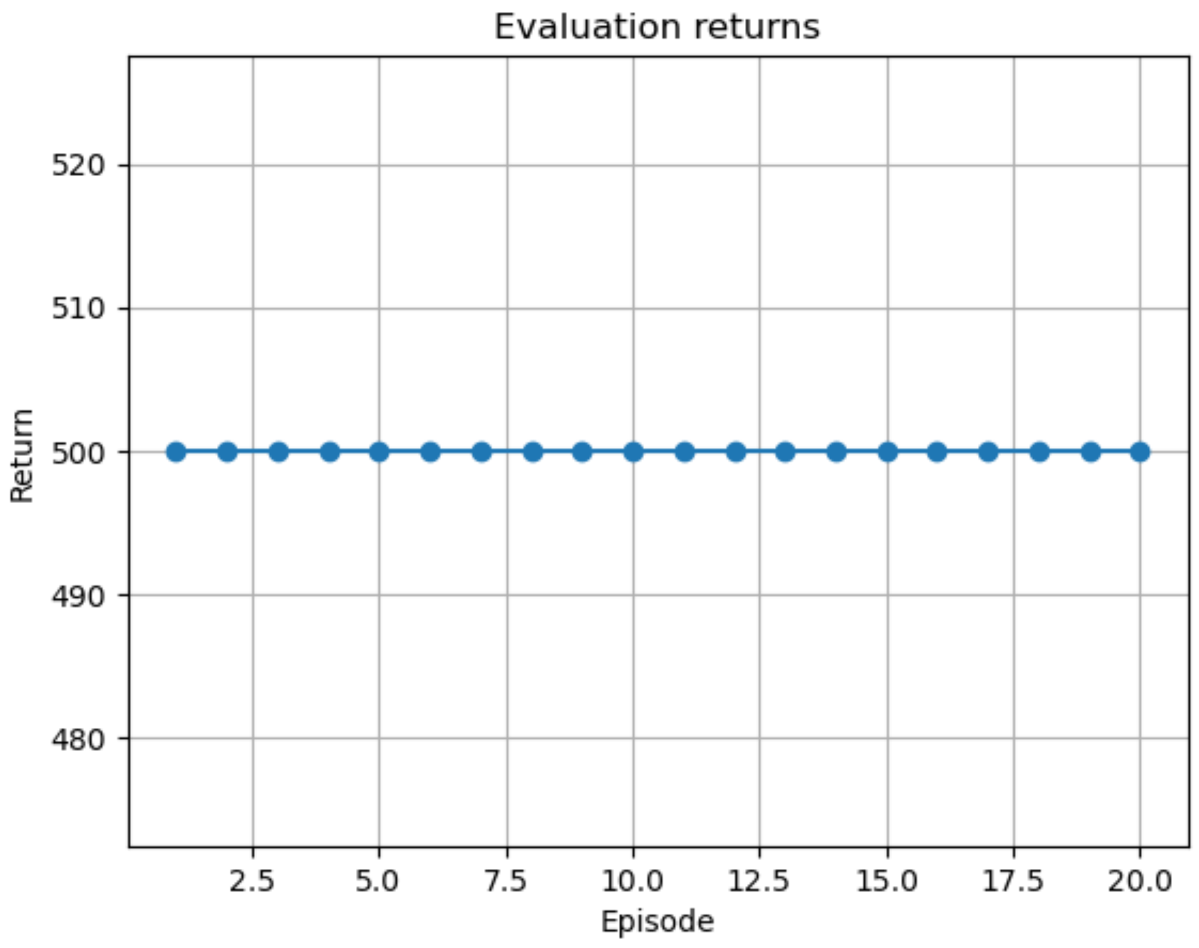
In [21]:
```python
# evaluating
def evaluate(model, env, episodes=20):
    returns = []
    for ep in range(1, episodes + 1):
        obs = env.reset()
        done = False
        total_reward = 0.0
        while not done:
            action, _ = model.predict(obs)
            obs, reward, done, _ = env.step(action)
            total_reward += reward
        returns.append(total_reward)
        print(f"Episode {ep}: Return = {total_reward}")
    print(f"Mean Return over {episodes} episodes: {np.mean(returns):.2f}"
    return returns
```

In [26]:
```python
returns = evaluate(model, env, episodes=20)
```

```
Episode 1: Return = 500.0
Episode 2: Return = 500.0
Episode 3: Return = 500.0
Episode 4: Return = 500.0
Episode 5: Return = 500.0
Episode 6: Return = 500.0
Episode 7: Return = 500.0
Episode 8: Return = 500.0
Episode 9: Return = 500.0
Episode 10: Return = 500.0
Episode 11: Return = 500.0
Episode 12: Return = 500.0
Episode 13: Return = 500.0
Episode 14: Return = 500.0
Episode 15: Return = 500.0
Episode 16: Return = 500.0
Episode 17: Return = 500.0
Episode 18: Return = 500.0
Episode 19: Return = 500.0
Episode 20: Return = 500.0
Mean Return over 20 episodes: 500.00
```

In [23]:
```python
def plot_returns(returns):
    episodes = list(range(1, len(returns) + 1))
    plt.figure()
    plt.plot(episodes, returns, marker='o', linestyle='-')
    plt.xlabel('Episode')
    plt.ylabel('Return')
    plt.title('Evaluation returns')
    plt.grid(True)
    plt.show()
```

In [25]:
```python
plot_returns(returns)
```


Evaluation returns
Return
Episode
520
510
500
490
480
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

```
In [ ]:
```

```
!pip install --upgrade pip setuptools wheel
!pip install gym==0.26.2
!pip install ale-py==0.7.4
!pip install autorom[accept-rom-license] --no-cache-dir
!pip install "ray[rllib]" matplotlib
```

```
Requirement already satisfied: pip in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (24.3.1)
Collecting pip
  Downloading pip-25.1.1-py3-none-any.whl.metadata (3.6 kB)
Requirement already satisfied: setuptools in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (75.1.0)
Collecting setuptools
  Using cached setuptools-80.4.0-py3-none-any.whl.metadata (6.5 kB)
Collecting wheel
  Downloading wheel-0.45.1-py3-none-any.whl.metadata (2.3 kB)
Downloading pip-25.1.1-py3-none-any.whl (1.8 MB)
   ---------------------------------------- 0.0/1.8 MB ? eta -:--:--
   ---------------------------------------- 1.8/1.8 MB 14.3 MB/s eta
0:00:00
Using cached setuptools-80.4.0-py3-none-any.whl (1.2 MB)
Downloading wheel-0.45.1-py3-none-any.whl (72 kB)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
ERROR: To modify pip, please run the following command:
C:\Users\Sara Iqbal\AppData\Local\Programs\Python\Python312\python.exe
-m pip install --upgrade pip setuptools wheel

Requirement already satisfied: gym==0.26.2 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (0.26.2)
Requirement already satisfied: numpy>=1.18.0 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
gym==0.26.2) (2.2.5)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
gym==0.26.2) (3.1.0)
Requirement already satisfied: gym_notices>=0.0.4 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
gym==0.26.2) (0.0.8)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip
ERROR: Could not find a version that satisfies the requirement ale-
py==0.7.4 (from versions: 0.9.0, 0.9.1, 0.10.0, 0.10.1, 0.10.2,
0.11.0)

[notice] A new release of pip is available: 24.3.1 -> 25.1.1
```

```
[notice] To update, run: python.exe -m pip install --upgrade pip
ERROR: No matching distribution found for ale-py==0.7.4

Collecting autorom[accept-rom-license]
  Downloading AutoROM-0.6.1-py3-none-any.whl.metadata (2.4 kB)
Requirement already satisfied: click in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from
autorom[accept-rom-license]) (8.1.7)
Requirement already satisfied: requests in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
autorom[accept-rom-license]) (2.32.3)
Collecting AutoROM.accept-rom-license (from autorom[accept-rom-
license])
  Downloading AutoROM.accept-rom-license-0.6.1.tar.gz (434 kB)
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: colorama in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from click-
>autorom[accept-rom-license]) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
sara iqbal\appdata\local\programs\python\python312\lib\site-packages
(from requests->autorom[accept-rom-license]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
requests->autorom[accept-rom-license]) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
requests->autorom[accept-rom-license]) (1.26.20)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
requests->autorom[accept-rom-license]) (2024.8.30)
Downloading AutoROM-0.6.1-py3-none-any.whl (9.4 kB)
Building wheels for collected packages: AutoROM.accept-rom-license
  Building wheel for AutoROM.accept-rom-license (pyproject.toml):
started
  Building wheel for AutoROM.accept-rom-license (pyproject.toml):
finished with status 'done'
  Created wheel for AutoROM.accept-rom-license:
filename=autorom_accept_rom_license-0.6.1-py3-none-any.whl size=446730
sha256=44b0ba0bce80c285b3bca06967f3b632e4ffcf0d8a6b66ca0389ded8b4de4ea
2
  Stored in directory: C:\Users\Sara Iqbal\AppData\Local\Temp\pip-
ephem-wheel-cache-5um4e5tc\wheels\99\f1\ff\
c6966c034a8259164bdc9deb4d1ea839f119474638100e6645
Successfully built AutoROM.accept-rom-license
```

```
Installing collected packages: AutoROM.accept-rom-license, autorom
Successfully installed AutoROM.accept-rom-license-0.6.1 autorom-0.6.1


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

Requirement already satisfied: matplotlib in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (3.9.2)
Requirement already satisfied: ray[rllib] in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (2.46.0)
Requirement already satisfied: click>=7.0 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (8.1.7)
Requirement already satisfied: filelock in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (3.17.0)
Requirement already satisfied: jsonschema in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (4.23.0)
Requirement already satisfied: msgpack<2.0.0,>=1.0.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (1.1.0)
Requirement already satisfied: packaging in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (24.1)
Requirement already satisfied: protobuf!=3.19.5,>=3.15.3 in c:\users\
sara iqbal\appdata\local\programs\python\python312\lib\site-packages
(from ray[rllib]) (6.30.2)
Requirement already satisfied: pyyaml in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(6.0.2)
Requirement already satisfied: requests in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (2.32.3)
Requirement already satisfied: pandas in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(2.2.3)
Requirement already satisfied: tensorboardX>=1.9 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (2.6.2.2)
Requirement already satisfied: pyarrow>=9.0.0 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (20.0.0)
Requirement already satisfied: fsspec in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(2024.12.0)
Requirement already satisfied: dm_tree in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(0.1.9)
```

```
Requirement already satisfied: gymnasium==1.0.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (1.0.0)
Requirement already satisfied: lz4 in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(4.4.4)
Requirement already satisfied: ormsgpack==1.7.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
ray[rllib]) (1.7.0)
Requirement already satisfied: scipy in c:\users\sara iqbal\appdata\
local\programs\python\python312\lib\site-packages (from ray[rllib])
(1.14.1)
Requirement already satisfied: numpy>=1.21.0 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
gymnasium==1.0.0->ray[rllib]) (2.2.5)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
gymnasium==1.0.0->ray[rllib]) (3.1.0)
Requirement already satisfied: typing-extensions>=4.3.0 in c:\users\
sara iqbal\appdata\local\programs\python\python312\lib\site-packages
(from gymnasium==1.0.0->ray[rllib]) (4.12.2)
Requirement already satisfied: farama-notifications>=0.0.1 in c:\
users\sara iqbal\appdata\local\programs\python\python312\lib\site-
packages (from gymnasium==1.0.0->ray[rllib]) (0.0.4)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (1.4.7)
Requirement already satisfied: pillow>=8 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: colorama in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
click>=7.0->ray[rllib]) (0.4.6)
Requirement already satisfied: six>=1.5 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
python-dateutil>=2.7->matplotlib) (1.16.0)
Requirement already satisfied: absl-py>=0.6.1 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
dm_tree->ray[rllib]) (2.1.0)
Requirement already satisfied: attrs>=18.2.0 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
dm_tree->ray[rllib]) (24.2.0)
Requirement already satisfied: wrapt>=1.11.2 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
dm_tree->ray[rllib]) (1.17.2)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in
c:\users\sara iqbal\appdata\local\programs\python\python312\lib\site-
packages (from jsonschema->ray[rllib]) (2024.10.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
jsonschema->ray[rllib]) (0.35.1)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
jsonschema->ray[rllib]) (0.20.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
pandas->ray[rllib]) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
pandas->ray[rllib]) (2024.2)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\
sara iqbal\appdata\local\programs\python\python312\lib\site-packages
(from requests->ray[rllib]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\sara iqbal\
appdata\local\programs\python\python312\lib\site-packages (from
requests->ray[rllib]) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
requests->ray[rllib]) (1.26.20)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\sara
iqbal\appdata\local\programs\python\python312\lib\site-packages (from
requests->ray[rllib]) (2024.8.30)


[notice] A new release of pip is available: 24.3.1 -> 25.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

# Import Required Libraries

import ray
from ray import tune, air
from ray.rllib.algorithms.ppo import PPOConfig
import matplotlib.pyplot as plt
import os
```

```python
import json
import numpy as np

# Initialize Ray and Configure PPO for Pong

ray.shutdown() # restart Ray if already running
ray.init(ignore_reinit_error=True)

config = (
PPOConfig()
.environment("PongNoFrameskip-v4")
.framework("torch")
.rollouts(num_rollout_workers=2)
.training(
gamma=0.99,
lr=5e-4,
train_batch_size=4000,
)
.resources(num_gpus=0) # set to 1 if running on GPU
)
```

```
2025-05-11 15:39:24,149    ERROR services.py:1362 -- Failed to start
the dashboard , return code 3221226505
2025-05-11 15:39:24,154    ERROR services.py:1387 -- Error should be
written to 'dashboard.log' or 'dashboard.err'. We are printing the
last 20 lines for you. See 'https://docs.ray.io/en/master/ray-
observability/user-guides/configure-logging.html#logging-directory-
structure' to find where the log file is.
2025-05-11 15:39:24,157    ERROR services.py:1431 --
The last 20 lines of C:\Users\SARAIQ~1\AppData\Local\Temp\ray\
session_2025-05-11_15-39-13_301673_34884\logs\dashboard.log (it
contains the error message from the dashboard):
Traceback (most recent call last):
  File "C:\Users\Sara Iqbal\AppData\Local\Programs\Python\Python312\
Lib\site-packages\ray\dashboard\dashboard.py", line 247, in <module>
    logging_utils.redirect_stdout_stderr_if_needed(
  File "C:\Users\Sara Iqbal\AppData\Local\Programs\Python\Python312\
Lib\site-packages\ray\_private\logging_utils.py", line 48, in
redirect_stdout_stderr_if_needed
    sys.stderr = open_log(stderr_fileno, unbuffered=True,
closefd=False)

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\Sara Iqbal\AppData\Local\Programs\Python\Python312\
Lib\site-packages\ray\_private\utils.py", line 446, in open_log
    stream = open(path, **kwargs)
             ^^^^^^^^^^^^^^^^^^^^
OSError: [WinError 6] The handle is invalid
```

```
2025-05-11 15:39:24,538    INFO worker.py:1888 -- Started a local Ray
instance.

# Train the PPO Agent (20 Iterations)

tuner = tune.Tuner(
"PPO",
param_space=config.to_dict(),
run_config=air.RunConfig(
stop={"training_iteration": 20},
checkpoint_config=air.CheckpointConfig(checkpoint_at_end=True),
verbose=1,
name="PPO_Pong_Training",
),
)
results = tuner.fit()

# Extract Training Results

result_dir = results.get_best_result().log_dir
metrics_path = os.path.join(result_dir, "result.json")

iterations = []
mean_rewards = []
reward_variance = []

with open(metrics_path, "r") as f:
for line in f:
data = json.loads(line)
if "episode_reward_mean" in data:
iterations.append(data["training_iteration"])
mean_rewards.append(data["episode_reward_mean"])
reward_variance.append(np.var(data.get("hist_stats",
{}).get("episode_reward", [0])))

#STEP 6: Plot Mean Episode Reward

plt.figure(figsize=(10, 5))
plt.plot(iterations, mean_rewards, marker='o', label="Mean Reward")
plt.xlabel("Training Iteration")
plt.ylabel("Mean Episode Reward")
plt.title("PPO on Pong – Mean Episode Reward")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

#STEP 7: Plot Reward Variance

plt.figure(figsize=(10, 5))
plt.plot(iterations, reward_variance, marker='x', color='orange',
label="Reward Variance")
plt.xlabel("Training Iteration")
plt.ylabel("Reward Variance")
plt.title("PPO on Pong — Reward Variance")
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()

#STEP 8: Clean Up
ray.shutdown()
```