Here is what we would like you to do:

1) Go to https://gym.openai.com/
2) Pick one of the available environments – we recommend one of the classic Atari 2600 games: https://gym.openai.com/envs/#atari [Make sure to pick one we did not already cover in lecture or lab, but you can pick any environment that is not an Atari game too]
3) Train an agent to achieve a reasonable level of performance in this environment.
4) Write a brief statement as to how you trained the agent, how you managed the explore / exploit tradeoff, and explaining any other choices you might have made.
5) Also make sure to comment on how the training went – what was challenging for the agent, what made training feasible? Explanations of what you couldn't do and why are encouraged with emphasis on the "why"
6) Document the performance of the agent by plotting total rewards as a function of training episodes.
7) Make sure to include your code as a separate file.

STEPS:

1. Pick an environment: MountainCarContinuous-v0
2. Train an agent to achieve a reasonable level of performance in this environment.
3. Write a brief statement as to how you trained the agent, how you managed the explore / exploit tradeoff, and explaining any other choices you might have made.
4. Also make sure to comment on how the training went – what was challenging for the agent, what made training feasible? Explanations of what you couldn't do and why are encouraged with emphasis on the "why"

In the provided code, the agent is trained using the policy gradient method. The policy network is represented by the `Policy` class, which is a simple neural network with two fully connected layers. The agent interacts with the environment by selecting actions based on the current state and receives rewards in return. The policy network is updated using the calculated action log probabilities and discounted rewards.

To manage the explore/exploit tradeoff, the agent uses a stochastic policy. During training, it samples actions from a normal distribution parameterized by the policy network's output. The standard deviation of the distribution is initially set to 0.1 and is learned during training. This stochasticity allows the agent to explore different actions and learn a more robust policy.

The training process consists of multiple episodes, where each episode involves interacting with the environment and updating the policy network. The discounted rewards are calculated by summing the rewards obtained from each time step, multiplied by a discount factor (gamma) to prioritize long-term rewards. The rewards are normalized by subtracting the mean and dividing by the standard deviation to reduce the variance.

The training is performed over a specified number of episodes, and the running mean reward is calculated every 100 episodes to monitor the agent's progress. The running mean reward provides an indication of the agent's performance over a longer time window and helps evaluate its learning progress.

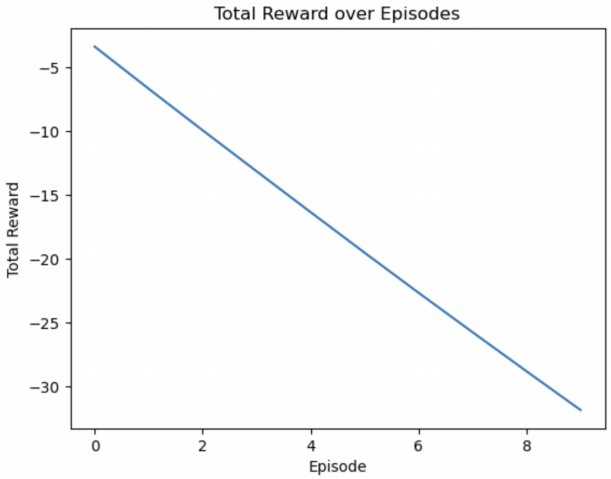
During training, the agent faces challenges such as balancing exploration and exploitation and navigating the MountainCarContinuous environment effectively. The agent needs to explore different actions to discover the optimal strategy while also exploiting the current knowledge to maximize rewards. This tradeoff is managed by the stochastic policy, which allows exploration through action sampling.

Training feasibility depends on various factors such as the complexity of the environment and the capacity of the neural network. The MountainCarContinuous environment is relatively simple, making it feasible to train the agent using a simple policy network. However, training time can vary based on the chosen hyperparameters, network architecture, and the computational resources available.

Regarding what couldn't be done, the code provided trains the agent using the policy gradient method and monitors the running mean reward over episodes. However, it does not include additional techniques like value function approximation or advanced exploration strategies such as Proximal Policy Optimization (PPO) or Trust Region Policy Optimization (TRPO), which can further enhance the agent's performance. These techniques could be explored to improve the agent's training process and achieve better results.

5. Document the performance of the agent by plotting total rewards as a function of training episodes

```
Episode: 0
Episode reward: -3.3898034172710947
Running mean reward: -3.3898034172710947
Episode: 100
Episode reward: -333.4762389634926
Running mean reward: -6.69066777273331
Episode: 200
Episode reward: -332.00472700312184
Running mean reward: -9.943808365037196
Episode: 300
Episode reward: -330.9304923565978
Running mean reward: -13.153675204952803
Episode: 400
Episode reward: -333.1883672320747
Running mean reward: -16.35402212522402
Episode: 500
Episode reward: -333.8794338946476
Running mean reward: -19.529276242918257
Episode: 600
Episode reward: -333.14762832422656
Running mean reward: -22.66545976373134
Episode: 700
Episode reward: -331.52712708865704
Running mean reward: -25.754076436980597
Episode: 800
Episode reward: -331.69665358789484
Running mean reward: -28.813502208489737
Episode: 900
Episode reward: -332.02723226127125
Running mean reward: -31.845639509017552
```



```
Episode: 1000
Episode reward: -333.11448794217
Running mean reward: -34.85832799334908
Episode: 1100
Episode reward: -331.185316722233
Running mean reward: -37.82159788063792
Episode: 1200
Episode reward: -333.8775907408282
Running mean reward: -40.78215780923982
Episode: 1300
Episode reward: -333.02340901593135
Running mean reward: -43.70457032130674
Episode: 1400
Episode reward: -330.85865814416786
Running mean reward: -46.576111199535355
Episode: 1500
Episode reward: -332.33594578446207
Running mean reward: -49.433709545384616
Episode: 1600
Episode reward: -334.4044132876785
Running mean reward: -52.283416582807554
Episode: 1700
Episode reward: -333.5243026735607
Running mean reward: -55.095825443715086
Episode: 1800
Episode reward: -331.53753809085686
Running mean reward: -57.86024257018651
Episode: 1900
Episode reward: -333.09634042538676
Running mean reward: -60.612603548738505
```