# Machine Learning -Report Assignment 2 – Deep Reinforcement Learning

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No | Team members | Student Id | Email |
| 1. | Kevin John Mathew | 201591357 | [k.j.mathew@liverpool.ac.uk](mailto:k.j.mathew@liverpool.ac.uk) |
| 2. | Thalirajesh Vignesh Arjun Raj | 201672337 | [t.vignesh-arjun-raj@liverpool.ac.uk](mailto:t.vignesh-arjun-raj@liverpool.ac.uk) |
| 3. | Pothapragada Sri Krishna Chaitanya | 201650545 | [sgspotha@liverpool.ac.uk](mailto:sgspotha@liverpool.ac.uk) |
| 4. | Shreya Krishnarth | 201651868 | [s.krishnarth@liverpool.ac.uk](mailto:s.krishnarth@liverpool.ac.uk) |
| 5. | Sanjana Nitin Tarekar | 201671238 | [s.tarekar@liverpool.ac.uk](mailto:s.tarekar@liverpool.ac.uk) |

**Our Solution**

OpenAI Gym is a powerful tool for researchers and practitioners in the field of reinforcement learning. Its extensive library of environments, flexible interface, and useful tools make it an invaluable resource for developing and testing new machine learning algorithms, training agents on a variety of tasks, and advancing our understanding of how to build intelligent, adaptive agents. [1]

We attempt to train a Deep Q-Network (DQN) agent to simulate the OpenAI Gym’s CarRacing-v2 environment . The DQN agent adapts a Convolutional Neural Network (CNN) to learn a Q-value function that estimates the expected future reward for each action in a given state.

The agent then uses an epsilon-greedy policy to choose the actions based on the Q-values. The training process involves replaying the experiences stored in a memory buffer and then using the Bellman equation to update the Q-values. The approach we aimed includes some utility functions for processing images and generating state frames, as well as code for setting up a virtual display to render the game environment.

We import the necessary packages and installs some additional dependencies via pip and apt-get. We define a CarRacingDQNAgent class that contain the DQN agent's properties and methods. The agent is initialized with different hyperparameters, such as the action space, frame stack size, memory size, discount rate, and learning rate.

The build\_model method creates a CNN with several layers that takes in a stack of grayscale frames and outputs a Q-value for each action in the action space. The replay method samples experiences from the memory buffer, calculates the target Q-values, and trains the model using mini-batch gradient descent. The ‘act method’ selects an action to take based on the current state and the agent's exploration/exploitation strategy. The ‘memorize method’ stores the agent's experience in the memory buffer. The update\_target\_model method updates the weights of the target network, which is used to compute the target Q-values during training.

The main function sets up the CarRacing-v2 environment via the function gym.make() which will initialize the agent, and trains it using a loop that runs for a fixed number of episodes and get the accuracies of the reward attained. The loop repeatedly samples a state from the environment, uses the agent to select an action, applies the action to the environment, stores the resulting experience in the memory buffer, and updates the agent's Q-values. The loop also saves the model weights and renders the game environment at regular intervals. The script outputs the average reward per episode and the total training time.

**Exploration and Exploitation**

Exploration and Exploitation are two fundamental areas/concept in Deep Reinforcement learning. Here an agent learns to make appropriate decisions after interacting with the environment. In reinforcement learning, agents typically have partial knowledge about the states, actions, rewards, and resulting state. This partial knowledge can result in a dilemma for the agent because it may not know the optimal action to take in a given situation. [2]

For example, if the agent is playing a game and is not aware of all the possible moves that can be made, it may not know which move the best is to make in a particular situation. Additionally, if the agent is uncertain about the rewards associated with different actions, it may not be able to accurately estimate the expected reward for each action, which can further complicate the decision-making process.

In deep reinforcement learning, finding the optimal balance between exploration and exploitation is crucial for achieving the best results.

Exploration: Exploration is a crucial component of deep reinforcement learning (RL), as it allows an agent to learn more about the environment and potentially discover better policies. During exploration, an agent tries out different actions to see what rewards it can receive, without relying solely on its current policy. However, it is important to balance exploration and exploitation, as too much exploration can result in the agent not exploiting its current knowledge and not achieving optimal performance.

One common exploration strategy is epsilon-greedy, which involves choosing a random action with probability epsilon and choosing the action with the highest expected reward with probability (1-epsilon). Another strategy is Upper Confidence Bound (UCB), which balances the agent's current knowledge about the environment with the potential rewards of taking a particular action. A third strategy is Thompson sampling, which involves sampling from the posterior distribution of the expected rewards for each action.

However, these strategies may not always be sufficient for deep RL training, and researchers have proposed additional methods for better exploration. One such strategy is the use of an entropy loss term, which encourages the agent to take actions that are uncertain or have high variability in their expected rewards. By doing so, the agent can learn more about the environment and potentially discover better policies.

Another strategy is noise-based exploration, which involves adding random noise to the agent's actions or observations. This can encourage the agent to explore regions of the state space that it may not have otherwise explored, leading to better learning and more robust policies. [3]

Overall, exploration is an essential component of deep RL, and there are various strategies that can be used to ensure that the agent learns the most about the environment while achieving optimal performance. By continually exploring and refining its policies, an RL agent can achieve impressive results in a wide range of tasks, from playing games to navigating complex environments.

Exploitation: Exploitation refers to using the knowledge gained through exploration to take actions that maximize expected reward. By exploiting the best action at each step, an agent can improve its performance and achieve its goals more efficiently. [4]

However, relying solely on exploitation can lead to the agent getting stuck in a suboptimal policy and failing to discover better policies. Therefore, exploration is necessary to ensure that the agent continues to learn and improve over time.

One common strategy for balancing exploration and exploitation is to use a method called epsilon-greedy. This strategy involves choosing the action with the highest expected reward with probability (1-epsilon) and choosing a random action with probability epsilon. By introducing some amount of randomness into the agent's decision-making process, it can continue to explore new actions and potentially discover better policies.

Another strategy is to use upper confidence bound (UCB) methods, which balance the agent's current knowledge about the environment with the potential rewards of taking a particular action. This method involves selecting the action that maximizes a confidence bound on the expected reward, which encourages the agent to explore actions that have uncertain or variable expected rewards. [5]

Ultimately, the optimal strategy for balancing exploration and exploitation will depend on the specifics of the task and the agent's current knowledge. However, it is clear that both exploration and exploitation are necessary for achieving the best results in deep RL. By using our current knowledge to exploit the actions that seem to produce the most rewards while still exploring new actions, we can develop an effective overall strategy that balances exploration and exploitation and maximizes our performance in the task at hand.

Various algorithms and techniques have been developed to help agents overcome the dilemma of partial knowledge, such as Monte Carlo methods, Q-learning, and deep reinforcement learning. These approaches allow agents to gradually learn and improve their decision-making abilities over time, even when faced with partial knowledge about the environment. [6]

# References

|  |  |
| --- | --- |
| [1] | B. Sonawane, “Getting started with OpenAI Gym,” Medium, May 2018. [Online]. Available: https://towardsdatascience.com/getting-started-with-openai-gym-d2ac911f5cbc. |
| [2] | A. Yang, “Medium- What is Exploration vs. Exploitation in Reinforcement Learning?,” Jul 25. [Online]. Available: https://angelina-yang.medium.com/what-is-exploration-vs-exploitation-in-reinforcement-learning-a3b96dcc9503. |
| [3] | L. Weng, “Exploration Strategies in Deep Reinforcement Learning,” 7 June 2020. [Online]. Available: https://lilianweng.github.io/posts/2020-06-07-exploration-drl/. |
| [4] | Manning, “concept of exploitation in category deep learning,” [Online]. Available: https://livebook.manning.com/concept/deep-learning/exploitation. |
| [5] | JavaTpoint, “Exploitation and Exploration in Machine Learning,” [Online]. Available: https://www.javatpoint.com/exploitation-and-exploration-in-machine-learning. |
| [6] | P. Wei, “Exploration-Exploitation Strategies in Deep QNetworks Applied to Route-Finding Problems,” 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1684/1/012073/pdf. |