CS 614 – Applications of ML

Project – Gaming (Mountain Car – Deep Q-Learning)

Submitted by: Manil Shrestha

<u>Project: Agent Plays Mountain Car with Deep Q-Learning</u>

1. Pitch:

State the circumstances an organization or sets of users would be willing to fund the proposed ML application based on their perceived value. Value could be financial, or any other type of outcome organizations or users need to obtain (e.g., institutional image, customer satisfaction, increased quality, or efficiency).

Organizations or users, particularly those in the gaming industry seeking innovative ways to enhance player experiences, may be drawn to fund a Deep Q-learning based gaming project due to its advanced AI capabilities and potential for heightened user engagement. The enhanced dynamism and immersion offered by such an application can directly boost revenue generation, making the investment an attractive one. Additionally, the prospect of long-term savings in development costs, due to reduced need for manual coding of complex gaming scenarios offered by a well-structured ML application, further strengthens the appeal for potential funding.

2. Data source:

Indicate with a link where you obtained the data. If you generated the data yourself, please provide a link to the code of the used approach.

I used the gaming environment provided by OpenAI's Gym library.
https://gymnasium.farama.org/environments/classic_control/mountain_car/

Note:
For this project, I utilized the approach detailed in PyTorch's Deep Q-Learning tutorial, accessible at https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html. Reward mechanisms, neural network architecture, and hyperparameters were best modified to suit the specifics of the 'Mountain Car' gaming environment.

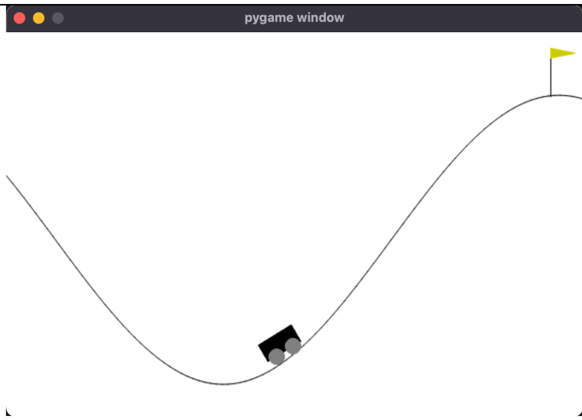
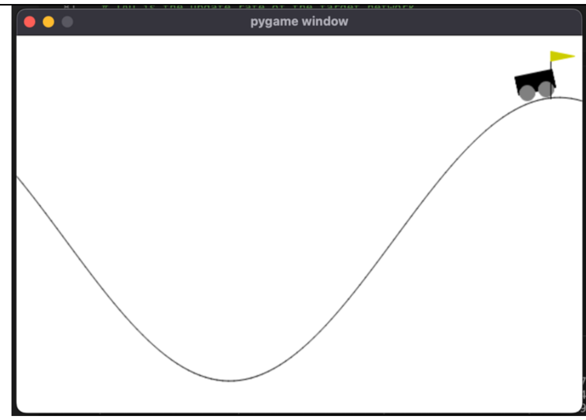
3. Model and data justification:

Justify why you chose a specific model to learn from the selected data. If you learned that a given more would be suitable for a type of data from a publication, please provide a link to the source. Note you still have to justify it with your own words (as always, limit to three sentences).

I chose Deep Q-Learning as the model for this project because of its proven efficiency in dealing with high-dimensional and complex environments, such as those found in many gaming situations. The deep learning component allows the model to extract useful representations from raw data, efficiently handling the intricacies of these problems. Furthermore, this model is highly versatile, providing a balance between exploration and exploitation, which is crucial in reinforcement learning scenarios.

4. Commented examples:

Indicate the input where trained model is applied, the output and whether it is as expected or any observations you may have.

 *Figure 1. Prior to the training process, the agent was unable to determine the necessary actions to reach the top of the environment.*	 *Figure 2 After training, the agent could successfully determine the actions that led to goal state.*

In the game, the goal is to direct the car to the peak on the right, an action that initially requires moving to the left to gain downhill momentum. Prior to the implementation of Deep Q-Learning, the agent was stuck at the bottom, but post-training, the agent successfully learned to initially move backwards, harnessing the gathered speed to reach the finish line.

5. Testing:

Provide a confusion matrix with one or more metrics and comment the results.

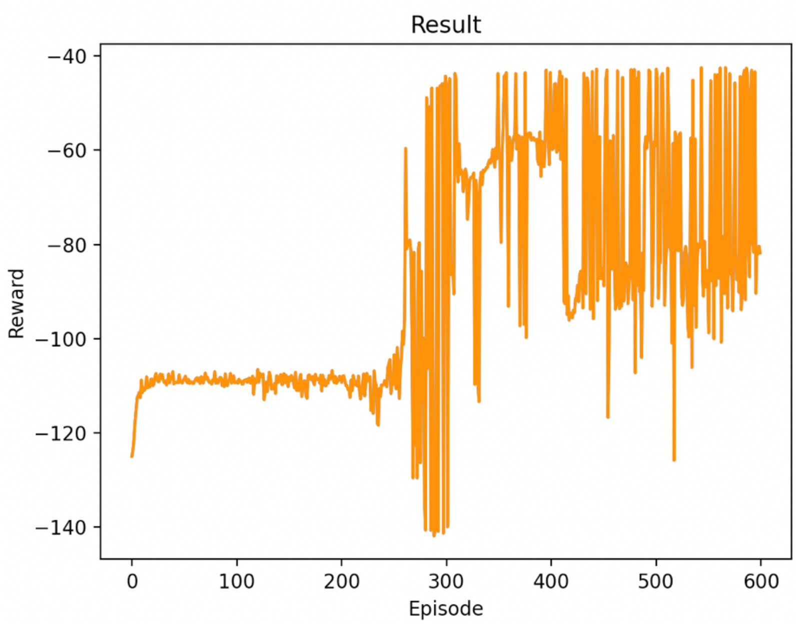


Figure 3 Reward value by training episodes

When I adjusted the hyperparameters to prioritize exploiting the Q-table over exploration, the reward plateaued, and the car was unable to reach the goal state. This is because while the immediate best action seems to be moving to the right, the strategy yielding long-term rewards involves moving left initially, a realization that only emerges with more exploration.

As you can see in the graph above, towards the end of training, the reward value exhibited significant fluctuation as the model was more focused on exploration rather than exploiting its acquired knowledge and ultimately learning to reach the goal state.

6. Code and instructions to run it:

Provide a link to the code and any required instructions to run it. Please include some testing examples so we can quickly experience what you experienced with the model.

Here is the link to python code which has been uploaded in the github:
https://github.com/ManilShrestha/AppliedMLProjects/blob/main/mountain-car-pytorch.py

You can recreate the training and experiments by following the steps:

1. Make sure all the dependencies are installed in your machine, you would need to install all the packages mentioned in the import section:
    a. Pygame
    b. Openai Gym
    c. PyTorch
    d. Numpy
    e. Matplotlib
2. Run the code with: python3 mountain-car-pytorch.py.
    - This initializes the training and once 600 episodes are played through and the Deep Q Network model is ready, 5 episodes are displayed in the screen, displaying the model's learned behavior along with the achieved reward values.