Deep Learning Lab
Reinforcement Learning

Ayisha Ryhana Dawood
5653447

May 2, 2024

1 Introduction

The assignment[1] focuses on Reinforcement Learning to train autonomous agents in OpenAI Gym. Deep Q-Networks are trained and evaluated on two different environments: CartPole and CarRacing.

The evaluation of the CartPole agent over 15 episodes resulted in an average score of 944.3 (std 215.8). The evaluation of the CarRacing agent over 15 episodes [2] resulted in an average score of 791.8 (std 45.9).

2 CartPole

In the CartPole environment, a pole is attached to a cart, which can move along a frictionless track. The goal is to balance the pole upright by applying left or right forces to the cart.

Configuring the DQN as a simple MLP with 3 layers achieves a mean score of 944.3 (std 215.8).

```
Model: MLP
    Hyperparameters:
        input: 4
        batch_size: 64
        learning_rate: 1e4
    Architecture:
        1. Linear: 400
        2. Linear: 400
        3. Linear: out=n_classes
    Training:
gamma: 0.95
epsilon: 0.1
tau: 0.01
num_episodes:200
```

[1] GitHub

[2] The YouTube video of the test run available here

3 CarRacing

CarRacing simulates driving a car on a track, where the agent must navigate through checkpoints while avoiding obstacles and staying on the road.

3.1 Setting up the agent

Each input frame from the environment shows the tracks, the position of the car, the reward, and bar graphs indicating the current acceleration state. Each frame is converted to greyscale.

To balance exploration and exploitation, a non-deterministic approach is taken towards training 10% of the time. However, the probabilities of sampling the 5 actions - straight, left, right, accelerate, and brake are not all equal. Instead, it is set to [0.3, 0.2, 0.2, 0.2, 0.1] because the distribution of actions in the environment during an expert run is not equal. The agent is expected to accelerate or go straight, more than the other three actions.

By incorporating n previous frames, along with the current one, the agent benefits from a history that enhances its understanding of the environment's temporal evolution. This expanded view improves the understanding of the car's movement, facilitating more informed decision-making during the run.

3.2 Model

The Deep Q-Network has the following architecture:

```
Model: CNN
    Hyperparameters:
        dimensions: 6x96x96
        batch_size: 64
        learning_rate: 1e5
    Architecture:
        1. Conv2d: 32 filters,
            kernel=(3,3), stride=1, pad=1
        2. ReLU
        3. MaxPool2d: kernel=(2,2)
        4. Conv2d: 32 filters,
            kernel=(3,3), stride=1, pad=1
        5. ReLU
        6. MaxPool2d: kernel=(2,2)
        7. Flatten
        8. Linear: out=512
        9. Dropout: p=0.1
        10. Linear: out=n_classes
```

3.3 Training and evaluation

The other training hyperparameters are as follows:

```
Training:
    gamma: 0.95
    epsilon: 0.1
    tau: 0.01
    history_length: 5
    acceleration: 0.5
    maxtimesteps:
        min(((step//5)+1)*100, 2000)
    skipframes: 5
    num_episodes:200
    sampling_prob of 5 actions:
        [0.3, 0.2, 0.2, 0.2, 0.1]
Evaluation:
    history_length: 5
    acceleration: 0.2
    skipframes: 5
```

The parameter skipframes enables skipping n frames between two next state predictions. The current prediction is repeated n times before making the next. This repetition allows for the impact of the action on the environment to be established clearly and then measured. Subsequently, the agent learns to take preventive steps to prevent the reduction of the rewards.

The model was trained with different values of skipframes - 0, 3, 5. The number of training timesteps increases with the number of episodes. This is because, initially the agent spends considerable time outside the tracks. Training for higher timesteps initially is not optimal for time and cost. After around 60 episodes, the model starts to show good progress in rewards.

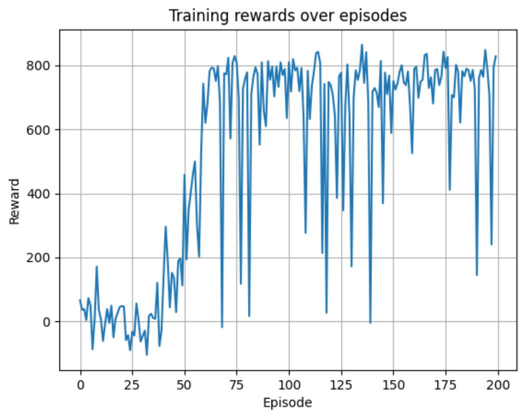

Figure 1: Training rewards over episodes for skipframes=5

Skipframes	Test score
0	393.9 (140.4)
3	660.9 (93.6)
5	791.8 (45.9)

Table 1: Testing results

The agent navigates the world by predicting one of 5 action IDs based on the input sequence of previous frames. The action IDs are mapped to the action configurations in different directions. The best mean score over 15 episodes was observed with skipframes = 5.

3.4 Configuration of the environment for test runs

It is observed that the agent performs better when the acceleration speeds are lower than the expert run as this enables the agent to move slower and make more precise decisions in subsequent frames.

4 Conclusion

Reinforcement learning has been successfully used to train an agent using a DQN to navigate a race track with mean score of 791.8 (std 45.9).