DL lab Assignnment 1

Students: Rean Fernandes

May 25, 2023

0.1 Imitation Learning

0.1.1 Data Collection

I collected training data for the imitation learning task in the carracing environment of gym over 20,000 steps. However, the data collection did not record the Brake action, likely due to a bug. To compensate, I trained the agent without the ability to brake to see how it would react to sharp turns.

0.1.2 Training Loop

The agent was trained over a sample of epochs $[10, 20, 50]$, with varying history lengths of $[1, 4, 8]$. The best performance was recorded with the following hyperparameters: {**lr:** 0.0001, **Batch size:** 256, **Epochs:** 50, **History length:** 4}

In the training loop, training data was sampled using weighted random sampling to balance the class distribution. The dataset was highly skewed towards the straight action, so weighted random sampling was used to balance the distribution of actions in the training data. During each epoch of the training loop, the agent is updated on the training data, and its performance is evaluated on the validation data. For each training step, the training accuracy and loss are calculated for the current batch of data, and the validation accuracy and loss are calculated every five steps. The best model found is saved, along with its hyperparameters, if its test reward exceeds the previous best reward. The test reward is the average reward obtained by the agent over five test episodes. Finally, the best reward obtained during training is returned. The best reward obtained during training is used to determine whether the current model should be saved or not. This approach is taken to avoid manually doing training and testing. I did not want the validation accuracy to be the sole metric to judge the agents performance, as the end goal was to train the agent to get the highest reward.

0.1.3 Network architecture

I used the CNN architecture used in Stable Baselines3. It is as follows:

- **Input layer:** 1 channel (greyscale) with size $(\text{batch size}, \text{history length}, 96, 96)$
- **Convolutional layer 1:** input features = $(\text{history length}, 96, 96)$, output features = $(32, 23, 23)$, filter size = 8×8, stride = 4
- **Convolutional layer 2:** input features = $(32, 23, 23)$, output features = $(64, 10, 10)$, filter size = 4×4, stride = 2

- **Convolutional layer 3:** input features = $(64, 10, 10)$, output features = $(64, 8, 8)$, filter size = 3×3, stride = 1
- **Activation layer:** Rectified Linear Unit (ReLU)
- **Flattening layer:**
- **Fully connected layer 1:** input features = 4096, output features = 512
- **Fully connected layer 2:** input features = 512, output features = output classes = 4 (no braking)

0.2 Reinforcement Learning CarRacing

The MLP based DQN agent was succesfully implemented for the CartPole environment. This section talks more about the CarRacing environment

0.2.1 Exploration

The initial probabilities for the exploration were
{**Straight:** 0.3, **Left:** 0.2, **Right:** 0.2, **Accelerate:** 0.3, **Brake:** 0.0}. I noticed that number of times the actions were used as training progress, converged to the following values {**Straight:** 0.125, **Left:** 0.24, **Right:** 0.16, **Accelerate:** 0.375, **Brake:** 0.1}. As for steering, the environment always starts in a position where the first turn taken is left. The probability for left was also slightly greater than that for the right, which makes sense since the car always turns to the left. Although I am unsure about whether the number of left and right turns are balanced in the environment.

0.2.2 Training

The model was trained over 1000 episodes. The following are the hyperparams:

- History length: 5
- Batch size: 32
- Number of actions: 5
- Gamma: 0.95
- Epsilon: 0.1
- Tau: 0.05
- Learning rate: 1e-5
- Number of episodes: 300
- Frames skipped : 3

0.2.3 Frame Skipping

Initially, I set the frame skip to 5 frames, but this resulted in erratic driving behaviour by the agent. The agent was not anticipating turns and overshooting this. I believe that this is due to the fact that its actions continues for too long, thus not allowing it to correct for the turns by braking in advance. Setting frame skip to 3 led to better performance by the agent as it would be prepared to act ahead.

0.2.4 Adapting Max Timesteps

Initially during training, since the agent is just exploring, there was no need to run the episodes for longer. Thus I scheduled the max timesteps to start with 250 and increase to 1000 as time progressed.

1 Results

1.1 Imitation Learning

1.1.1 Training Results

1.1.2 Data Augmentation Results

1.2 Reinforcement Learning

1.2.1 Training Results

1.2.2 Exploration Results

1.2.3 Frame Skipping Results

1.2.4 Adapting Max Timesteps Results

2 Conclusion