# Deep Learning Lab Course

Assignment 1: Imitation Learning and Reinforcement Learning

Soham Basu – 5576954 May 02, 2023

**Abstract**. The Imitation Learning model achieved a mean score of **884.14**, while the Reinforcement Learning agents achieved mean scores of **218.4** & **467.91** on CartPole and CarRacing tasks respectively.

# 1 Imitation Learning

## 1.1 Dataset

### 1.1.1 Dataset Creation

Two datasets were originally created with 7 and 18 episodes recorded, and scores of 876.08 and 800.92 respectively. Dataset 1 was the better one in terms of driving skill, but it was hugely unbalanced due to the brake action being seldom applied (80 out 15000 actions). The dataset with 18 episodes was created more carefully by conservatively making the apex of the tracks' corners and making sure a single key was pressed at a time. The 18 episodes generated a total of 40,000 samples. The dataset was split into the training and validation sets using a 9:1 ratio. This created 36000 training samples and 4000 validation samples.

### 1.1.2 Undersampling the majority class

From Fig. 1(a) it's clear that the dataset is quite imbalanced. Training the model on this imbalanced data created a huge bias towards going straight and the model didn't learn to accelerate or change directions well. Thus, 70% of the samples of the 'go straight' or action-id '0' class were discarded at random from the entire dataset. Although this would increase the variance of the model, in this particular game this isn't an issue since the goal is to essentially drive as fast as possible and string all corners as efficiently as possible. So ideally, the model only needs to learn *when* to turn and accelerate.

## 1.2 Behavioral Cloning Agent

### 1.2.1 Network Architecture

The Convolutional Neural Network shown in Fig. 2 was used for the Imitation Learning task.

### 1.2 2 Training and Hyperparameters

The model was trained for 100 epochs of 1000 iterations with batch size 128 and the Adam Optimizer with learning rate 3E-4. The model was trained using 3 history lengths [1, 3, 5] as instructed and the results are given in the Table 1.

## 1.3 Results

**Table** 1: Imitation Learning agent results for 3 history values

| History length | Mean Score | Standard Deviation |
|---|---|---|
| 1 | 884.14 | 35.37 |
| 3 | 644.28 | 269.31 |
| 5 | 281.46 | 180.63 |

## 1.4 Conclusion

The model was running quite slow since it was still overestimating the *go straight* action and as a result was often getting stuck while turning. Ideally, we would like the model to accelerate in most instances instead of just going straight. Thus, while testing, whenever the model's output was 0, the action was instead randomly sampled from 0 and 3 (*go straight* and *accelerate*) with weights of 0.4 and 0.6 respectively. More weight to *accelerate* caused the car to miss some low-speed, sharp corners and spin out of the track.

Around the first 40 states of every episode, the environment zooms into the track. The actions in these 40 steps were overwritten with *accelerate* since the model doesn't seem to have enough capacity to predict the actions at multiple scale and was starting off slow.

The validation accuracy doesn't seem to be very strongly correlated with the rewards since the model is essentially unaware of the rewards.

Training with more history states, made the model increasingly worse as it overcorrects the position on the straights after turns, and often spins out of track as a result (especially with history = 5).

# 2 Reinforcement Learning: Deep Q-Networks

## 2.1 CartPole Control Task

### 2.1.1 Neural Network Model

The Model used by the DQN agent for the CartPole task is the one already given in the repository, shown in Fig. 4. The DQN agent model for the CarRacing task used the same CNN used for Imitation Learning (Fig. 2).

### 2.1.2 Training and Hyperparameters

**Table 2**: Hyperparameter settings for DQN Agents

| Task | Max. Episodes | Replay Buffer Size | γ | T | Learning Rate | Batch Size |
|---|---|---|---|---|---|---|
| CartPole | 1000 | 10E5 | 0.95 | 0.01 | 1E-4 | 64 |
| CarRacing | 500 | 10 | 0.95 | 0.01 | 3E-4 | 64 |

The Hyperparameter settings of both the tasks are given in Table 2. For the CartPole task the DQN agent was trained for a maximum of 1000 timesteps per episode with a fixed ε value of 0.1. For the CarRacing task, the ε value and maximum iterations were kept constant at 1 and 300 respectively, until the 300th training episode. The ε value was then decayed by 0.995 until the end. The maximum iterations were increased by 1.2 of the training episode number. Both the agent used the Adam Optimizer.

### 2.1.3 Learning degradation and Catastrophic forgetting

When the agent was trained on the CartPole task with the Replay Buffer deleting the first frame after every frame addition, learning would 'degrade' and the model actually gets worse after around 300 episodes. This could be because of catastrophic forgetting. To combat that, instead of deleting the first frame in the buffer, the first 10% of the total frames in buffer were preserved and the deletion was done from the next frame onwards, so that the agent retains the knowledge of failure modes. It was observed that the agent indeed gets higher rewards after using this strategy. For the CarRacing task, no such strategy was used.

### 2.1.4 Results

**Table 3**: Reinforcement Learning Results

| Task | Mean Score | Standard Deviation |
|---|---|---|
| CartPole | 218.4 | 142.26 |
| CarRacing | 467.91 | 187.51 |

### 2.1.5 Conclusion

The CartPole control is quite an easy to solve task and as shown in Fig. 5, the DQN agent solves the problem quite early during training.

The CarRacing control is quite a hard and time-consuming task and after 500 training episodes the average evaluation reward of 467.91 is still quite bad compared to the result obtained by the Behavioral Cloning agent. It seemed a bit tricky to hit the right values with the probability bias in the ε-greedy exploration. The agent ended up with an overestimation bias towards *right* prediction (especially compared to *left*) and often drove just parallel to the right of the track, without correcting itself.

## Figures:

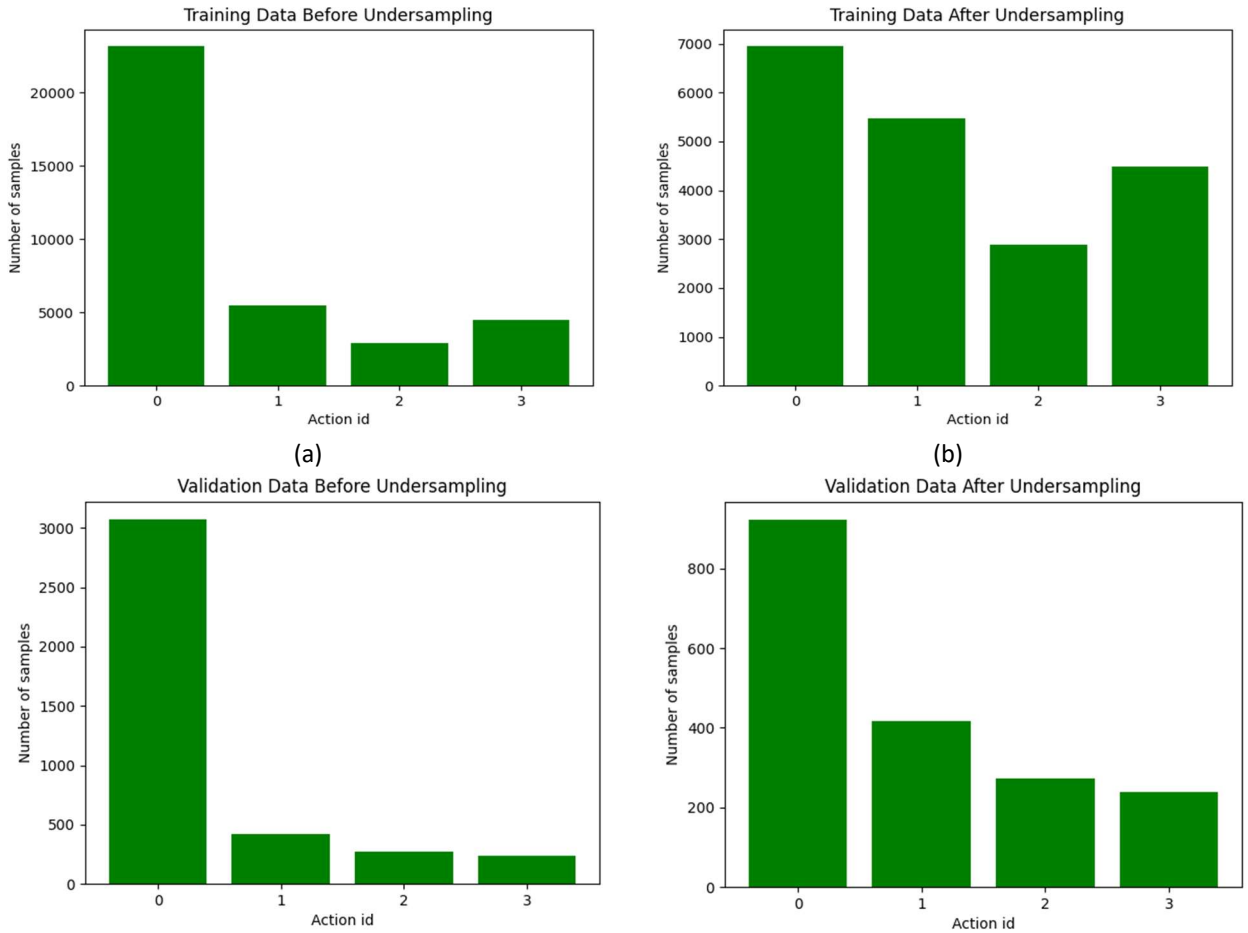



**Figure 1**: Training and Validation Data Before and After Majority class undersampling

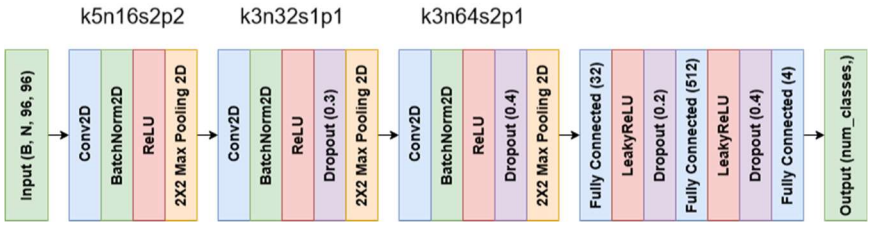



**Figure 2**: Convolutional Neural Network Model (k=kernel size, n=no. of output channel, s=strides, p=padding)

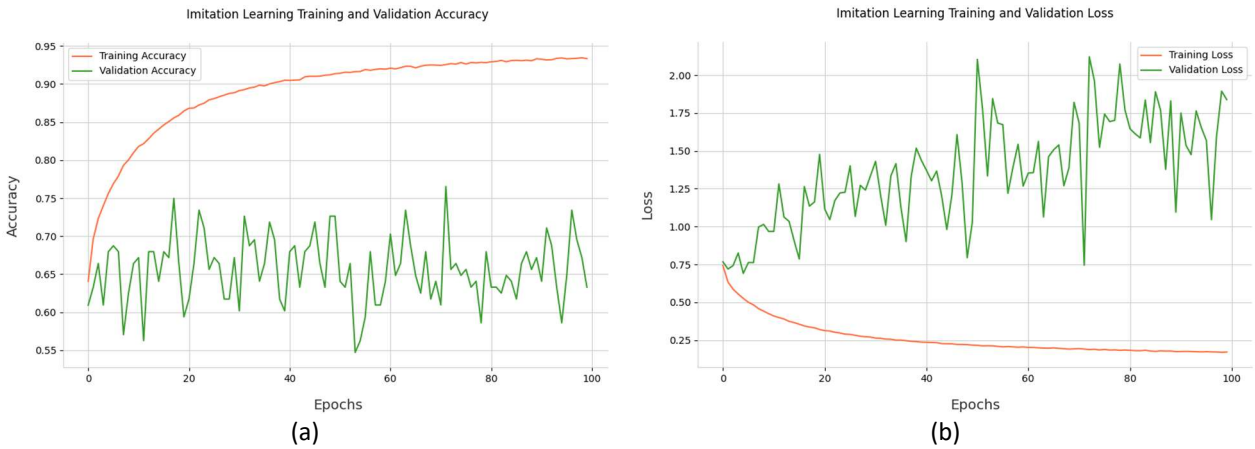



**Figure 3**: Training and Validation Results for Imitation Learning Task

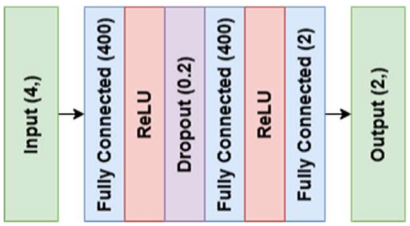
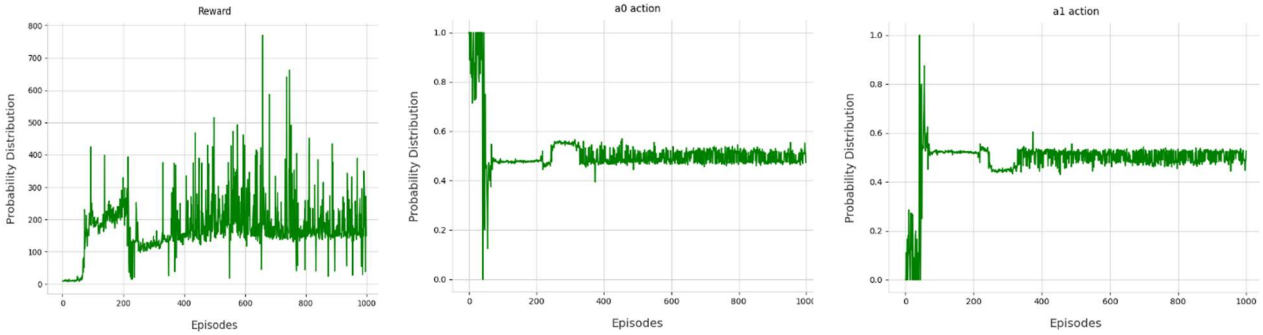
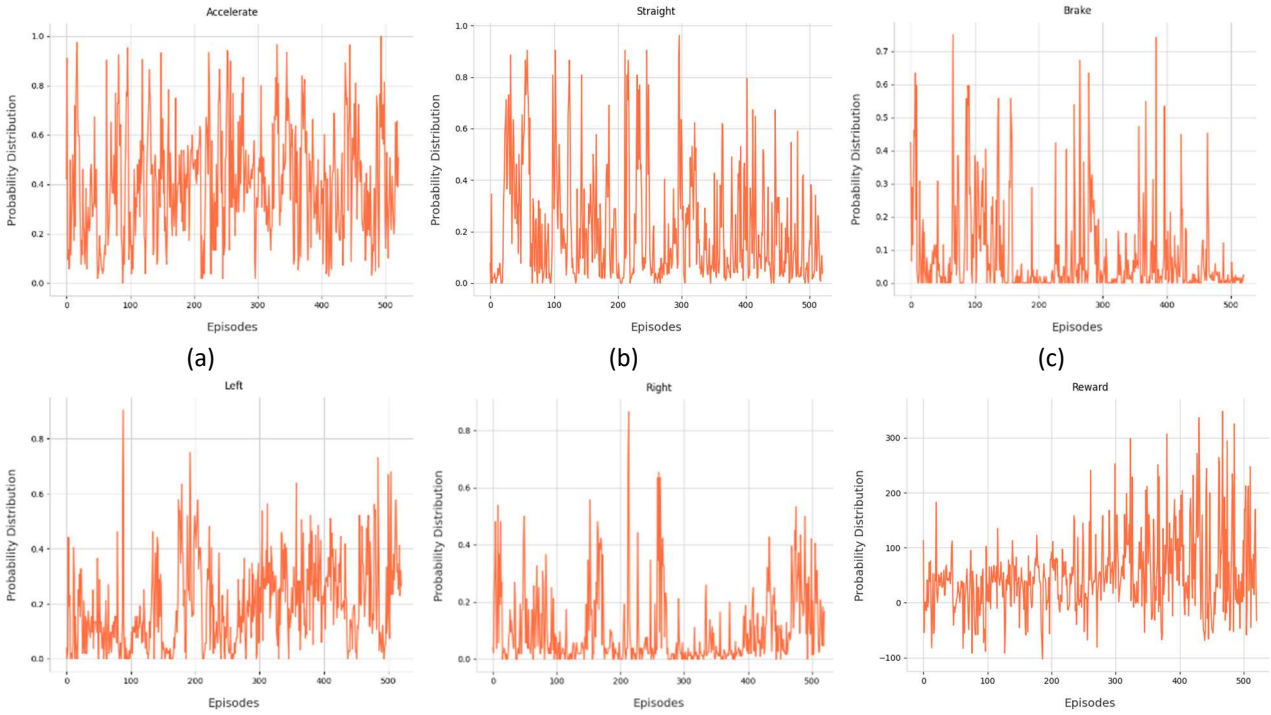



**Figure 4**: Model used as Q-network for the CartPole task




(a) (b) (c)

**Figure 5**: DQN Training for CartPole Task




(a) (b) (c)

(d) (e) (f)

**Figure 6**: CarRacing DQN Training