Deep Learning Lab Course

Imitation Learning and Reinforcement Learning

Sejal Mutakekar - 5775691

May 02, 2024

Contents

1 Abstract

Using Imitation learning and Reinforcement learning, behavioral cloning and DQN agent are implemented. The code has been executed in the Linux OS, by creating the virtual environment in conda.
Results (Means Scores) are as follows :

1. Imitation Learning - **543.69**
2. Reinforcement Learning -
   (a) Cartpole - **382.06**
   (b) CarRacing - **468.58**

2 Imitation Learning

2.1 Dataset

2.1.1 Dataset Used

Dataset contains manually generated data (Generated by playing a car racing game). Integer values are mapped to the meaningful action taken by pressing a direction key. Data contains various episodes having respective scores. For example : 6 episodes give a score of 767, 13 episodes give a score of 772.
Details -

- Total Data (samples) : 30,000
- During the training process, the data has been split into : Training dataset and Validation Dataset.
- Proportion of split : 9:1
- Samples in proportion : Training dataset is trained with a total of 27000 data samples, whereas validation dataset gets 3000 samples.

Since the car travels straight the maximum amount of time, the '0' action_id is the maximum, whereas the brake button is rarely pressed, hence action_id : 4 is hardly visible in the dataset, and hence arises the need for undersampling.

2.1.2 Data Preprocessing :

- **Undersampling** : After having the look at the **Figure.**1 denoting the distribution of the data, it is clear that the data is highly biased and may produce biased results (car may end up driving only straight). To deal with this, a few samples (around 60%) are removed randomly. This definately generates a variance, but may not necessarily impact the result, since ultimate goal is to take appropriate path (straight, left, right) and accelerate.
- **RGB2Gray** : RGB to Gray scale conversion has been done for dimensionalty reduction.
- **Discretization** : Discretization is done for the simplification of data.

2.2 Behavioral Cloning

2.2.1 Network Architecture

CNN (Convolutional Neural Network) as shown in **Figure.** 2 is used for Imitation Learning tasks.

2.2.2 Training and Hyperparameter settings

The model has been trained for 30 epochs, with 1000 mini batches (iterations), batch size of 100, optimizer as Adam and learning rate of 3e-4. The model was trained using various history lengths with varying performance as given in the table below.

Results :

History Length	Mean Score	Standard Deviation
1	543.69	110.68
4	389.30	151.29
8	304.69	161.67
11	278.14	165.95

Table 1 : **Results : Imitation Learning**

Graphs showing the Accuracy and Loss for Training and Validation can be seen in the **Figure.** 3

2.2.3 Inference

As mentioned earlier, since data was biased on movement of a car in one forward direction, the model was running slow and was getting stuck when it came to taking a turn. Since, the goal is also to accelerate the model, getting stuck was a hurdle. Hence, during testing, when the model gave an output '0', the action would be sampled randomly between 0 and 3. This sometimes caused a car to go off-track by missing a few turns.
Training the model by increasing the number of epochs and adjusting the hyperparameters improved the accuracy. Adding history length helped the model perform better, but adding too much of history length led it to overfit the data, which is evident from the table (for history length 11).

3 Reinforcement Learning

3.1 Network Architecture

The concept of DQN is used for Cartpole and CarRacing tasks. For a CartPole task, a simple network with Fully Connected layers and activation functions has been used **Figure.** 2. For a CarRacing task, CNN model has been used with various hyperparameters **Figure.** 2. The model used for CarRacing is in correspondence with the model used for Imitation Learning.

3.2 Training and Hyperparameter Setting

The table below (Table 2) denotes the Hyperparameters used for training both the models. Optimizer used for the above tasks is Adam Optimizer. Additionally, the Maximum timestep of 1000 and epsilon value of 0.01 is used. However, for CarRacing, the epsilon value is kept constant for initial 300 episodes and after that it goes on reducing by 0.995.

	Epochs	**Learning Rate**	**Batch Size**	**Replay Buffer**	γ	**T**
CartPole	1000	1e-3	64	10E5	0.95	0.01
CarRacing	1000	3e-4	64	10	0.95	0.01

Table 2 : **Hyperparameters : Reinforcement Learning**

3.3 Observation and Actions taken

In CartPole, the model does not tend to perform well and the performance goes on degrading due to Replay Buffer that deletes the first frame on addition of each frame. To overcome this degradation of performance, the deletion is stopped for the first few initial frames and then after 10% of total frames are completed, it starts deleting. Following this pattern of preserving the initial frames, the model is noticed to perform better and reward count gets higher.

3.4 Results

Statistics	**CartPole**	**CarRacing**
Mean Score	382.06	468.58
Standard Deviation	111.38	128.57

Table 3 : **Results : Reinforcement Learning**

Graphs showing the Results for CartPole and CarRacing can be seen in the **Figure.** 4 and **Figure.** 5 respectively.

3.5 Inference

After figuring out the Learning performance issue in CartPole and fixing it up with preserving initial frames technique, the task for CartPole was simplified.
However, in comparison to this, CarRacing is an intensively time consuming and 'hard to make the model learn' process. It takes longer time for a car to learn driving on tracks with proper turns and acceleration and hence there is a need of appropriate hyperparameters and episode count to accurately train the model. Sometimes, it goes off track and takes unnecessary turns, which in turn impacts the final performance of the car.

4 Online Resources

Graphs depicted above have been generated using tensorboard.
Additionally for a few of them data has been downloaded from tensorboard in csv files and graph is generated using matplotlib.

Code and CSV files for the same can be found under .Plots folder.
Online resource like 'draw.io' has been used for generating diagrams for Network Architectures.
Overleaf LaTex has been used for writing a Report.

5 Overall Conclusion

After comparing the results for both Imitation and Reinfocement Learning, following can be concluded -
The performance for the Imitation Learning is better in comparison with the CarRacing in Reinforcement Learning. Ideally, the performance for the Reinforcement Learning should be better, since we donot have the unbiased data, whereas in Imitation Learning the data is biased and hence we imply that performace for Reinforcement Learning should be better. But, due to computational constraints and limitations, it is hard to satisfy the ideology. Accurately changing the hyperparameters and training the model for more episodes will definitely help improve the performance.

6 Video

Click here for Videos

7 Appendix

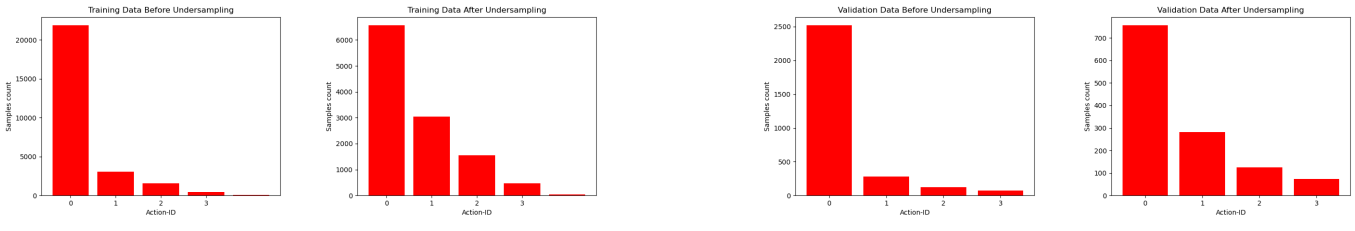

Figure 1: Undersampling

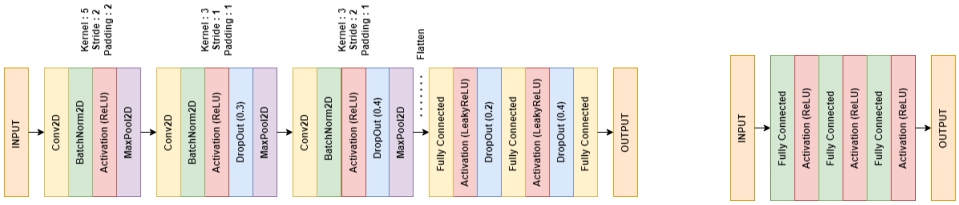

Figure 2: Network Architectures - a) CNN b) Model for CartPole

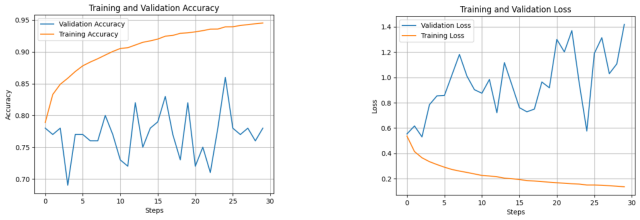

Figure 3: Results : Imitation Learning

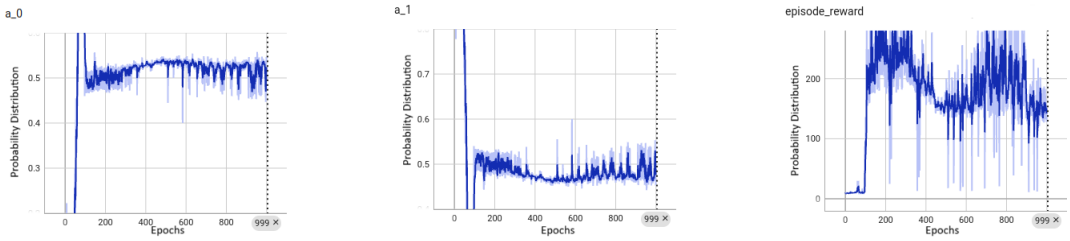

Figure 4: Results : CartPole

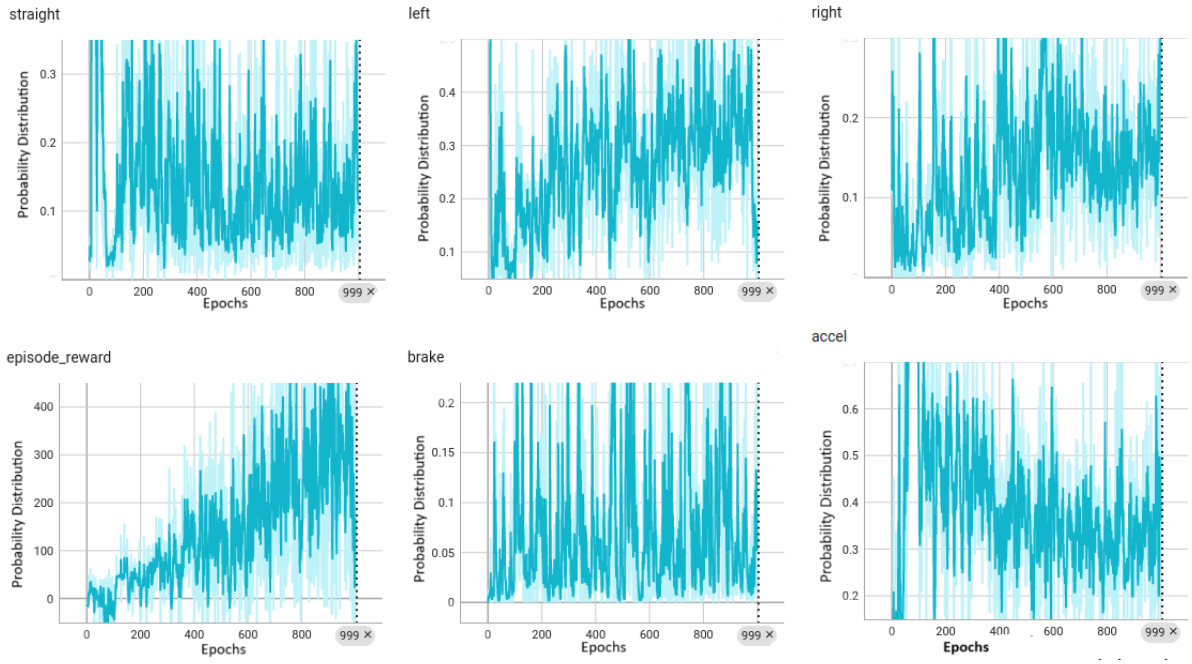

Figure 5: Results : CarRacing