Chapter 1

Imitation Learning

1.1 Architecture

The CNN network used initaial a Conv2d layer with inChannels = N (where N was the stack size) and out channel of 64 with 3x3 kernel. The output was then given to a batchnorm. It then used a Conv2d layer with inChannels = 64 and out channel of 32 with 3x3 kernel. The output was then given to a batchnorm. It was then passed through a max pool layer of kernel 4x4. It was then flattened and given to a linear layer with output nodes of 128. The input channel being the array of flattened layer. A dropout of 0.5 was performed on it. It was then passed through a CNN layer of 64 input channels and 32 out channels and to a drop out. It was passed to the final linear layer. After every layer including the conv2d layer a ReLu activation was performed. The linear layer finally mapped to the number of action. Argmax was take to find the best action.

1.2 Hyperparameters

The loss function used was CrossEntropyLoss. The learning rate was $1e^{-4}$. The optimizer used was Adam. Number of minibatches here was considered as epochs. The training was done for 100 epochs. Weighted sampling was done to weight the unbalanced dataset. Frame stacking was done in oreder to predict a resultant action in a given set of frames. A batch size of 64 was used.

1.3 Performance Plots

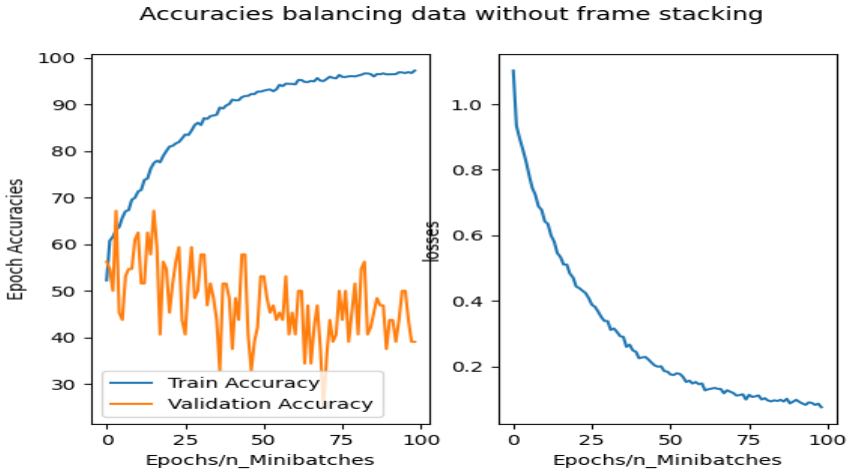

Figure 1.1: Performance plot with only balancing data

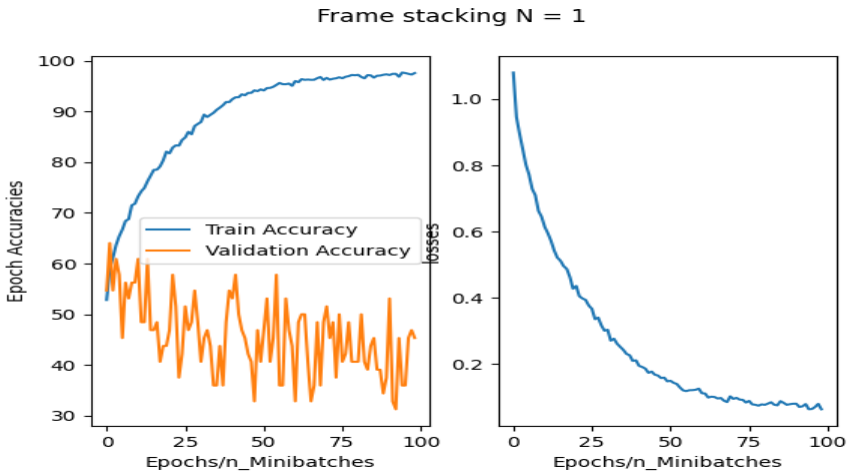

Figure 1.2: Performance plot for frame stacking N = 1

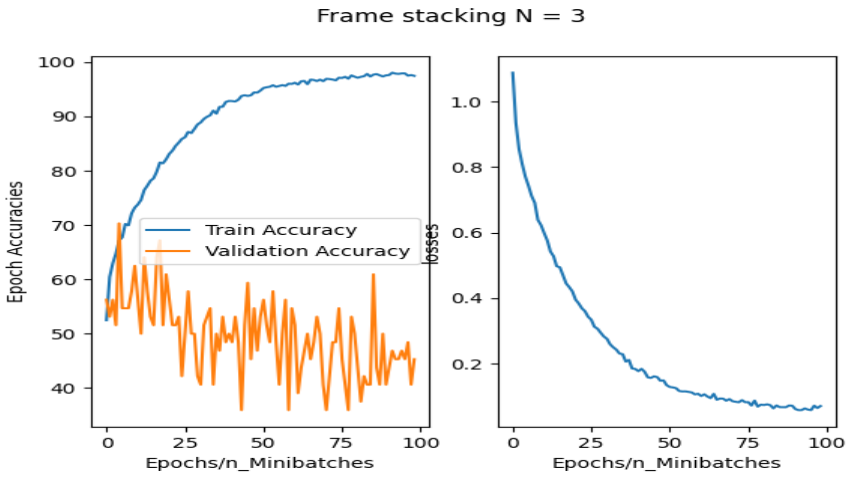

Figure 1.3: Performance plot for frame stacking N = 3

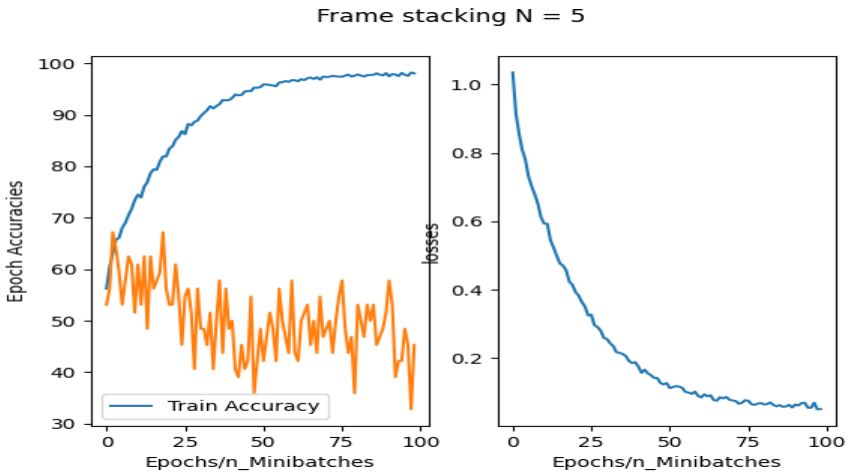

Figure 1.4: Performance plot for frame stacking N = 5

1.4 Approach

The sampling of minibatches were also done by using Dataloader and TensorDataset. The weighted sampling was accomplished by using the WeightedRandomSampler from torchvision package. To calculate the class weights a custom function called sampling was made. The Frame stacking was also done by using a custom function.

AccB	Acc1	Acc3	Acc5
-23.17	-43.05	64.69	201.98
50.92	-69.04	36.13	258.15
-25.63	-40.72	-13.59	367.25
18.60	-32.90	22.20	269.71
-33.21	-51.55	79.05	94.84
-28.42	-69.54	44.16	319.69
-33.21	-56.30	-22.43	359.96
-33.00	-55.42	-16.46	81.38
-28.18	-73.28	-13.29	229.37
-29.63	-63.86	-12.82	306.37
-21.76	-72.18	-18.18	249.01
-33.21	-53.72	-21.33	226.63
-19.94	-59.65	18.24	57.13
-23.17	-68.18	49.53	311.64
-25.63	-75.36	-10.54	248.49
Mean	Mean	Mean	Mean
-16.34	-58.98	12.35	238.77
std	std	std	std
-25.63	12.48	33.53	92.81

Table 1.1: Table of test reward values.

1.5 Observation and Conclusion

All the performance plots a good training accuracy. The loss to is depicting reduction. Although the Validation accuarcy is unstable and very low. This suggests that the model has overfitted by a large margin. In order to overcome the low validation accuracy a different CNN was tried but it was not successful. The imitation learning model here has overfit. The results of the different variation test runs are as follows:

The difficulty here was to get a good validation accuracy as it fluctuated greatly. Even in the end the agent with frame stacking of 5 performed the best. Frame stacking of 1 agent barely moved. Eventhough I tried changing the CNN network it barely worked. Even changing the train validation split size barely mattered. Hence the agents have a low reward.

Chapter 2

Reinforcement Learning

2.1 Hyperparameters

The loss function used was CrossEntropyLoss. The learning rate was $1e^{-4}$. The optimizer used was Adam. A total of 400 epochs were used. Weighted sampling was done to weight the unbalanced dataset. Frame stacking was done in oreder to predict a resultant action in a given set of frames. Adam optimizer was used with a learning rate of $1e^{-4}$. The CNN architecture was the same as that of the Imitation Learning agent. A batch size of 64 was used while sampling from the replay buffer.

2.2 Performance graphs

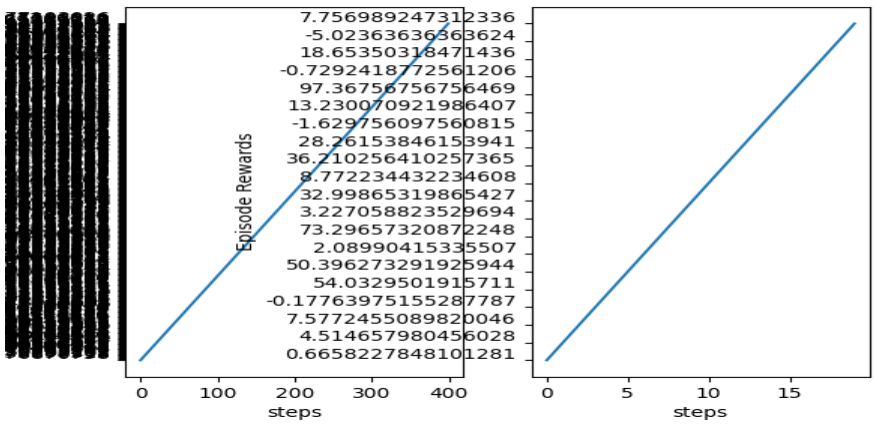

Figure 2.1: Episodes Rewards

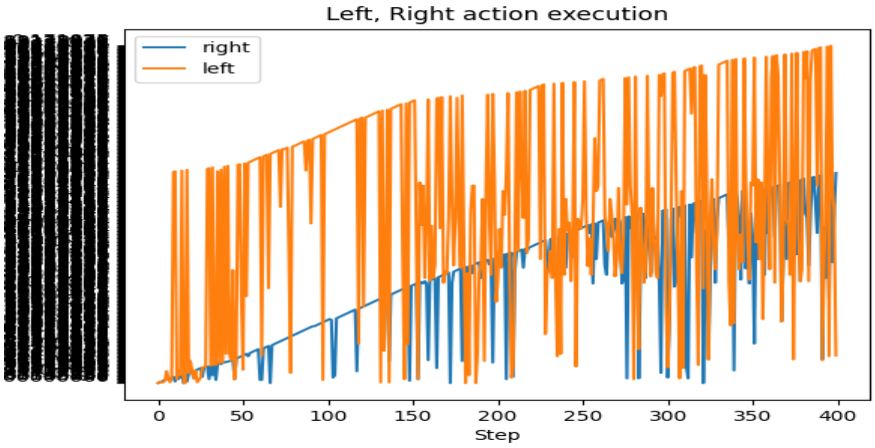

Figure 2.2: Left and Right Actions

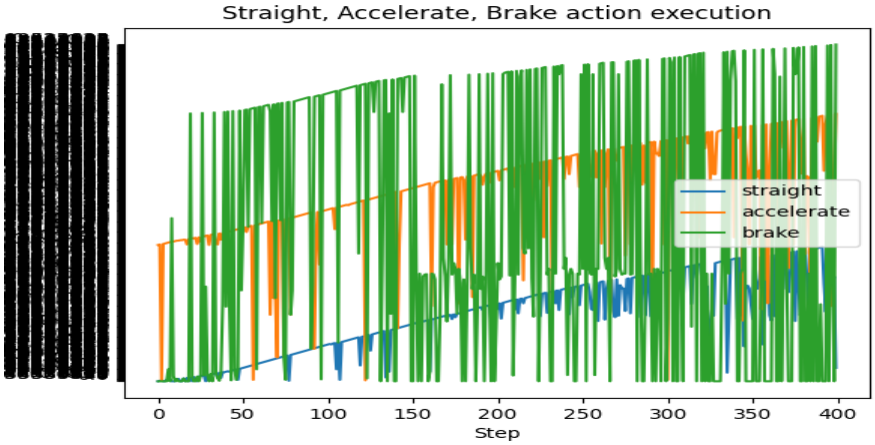

Figure 2.3: Accelerate, Brake, Straight Actions

The episode rewards are given below

Test Episode Rewards: [283.9606299212602, 13.734883720930938, 159.11406844106594, 131.8787878787888, 146.99352750809174, 24.411764705883428, 202.5766423357677, 210.25874125874185, 277.9700996677749, 225.9659442724466, 224.8139534883733, 209.25974025974136, 4.171428571429317, 51.73381294964057, 283.7540983606563], "mean": 163.37320822270618, "std": 95.38961839485157]

As I tried to obtain the graphs of the training process I was not able to get them on the tensorboard. I checked the code for indentation errors

but found none. I also added markers to the position when the tensorboard write should take place but had no success. Hence I tried to store all the action and reward values in a .txt file and then tried to plot but it did not result in a neat plot. I have also attached the text files along with the report.

2.3 Observation and Conclusions

Several methods were tried to improve the reward. One of them as given was skip frame. The weighting of actions was also performed. Accelerate was given the highest. Although when observing the rendering process action LEFT was being executed the most at the initial stages eventhough it had a low weight. Linear variance with time steps along with conditional statement were used to control the max time steps. Intially the max step count was kept low and then gradually increased. Even though it was done the max rewards were about 500 and very greatly fluctuating.

2.4 Cartpole

The performance plots for cartpole are given as belows. The max episodes rewards were being obtained at 180. But the mean episode rewards were not proceeding beyond 45. Changes were made to the replay buffer as to only to sample if there are batch size number of samples present in the buffer. Also a function was used to convert a variable to a tensor. There were few difficulties in plotting the graphs from tensorboard as a result the plot is not up to the mark. I tried to rectify the errors by plotting the graphs by placing them in a pandas dataframe and then converting it to an numpy array by reducing the decimals

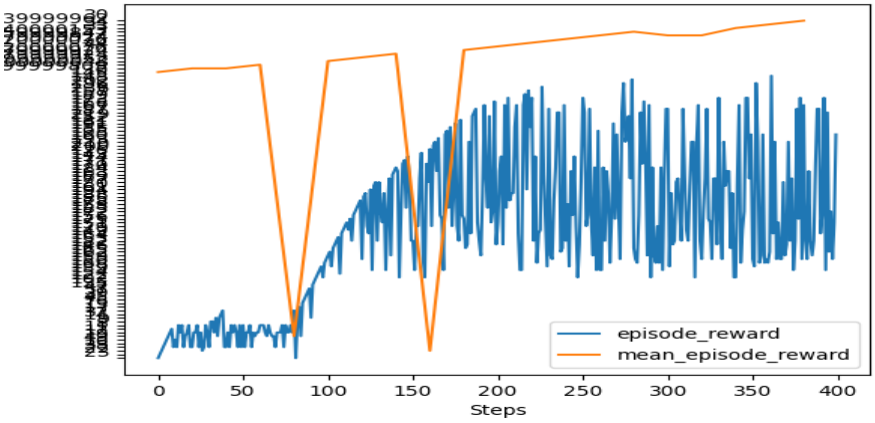

Figure 2.4: Episode Rewards

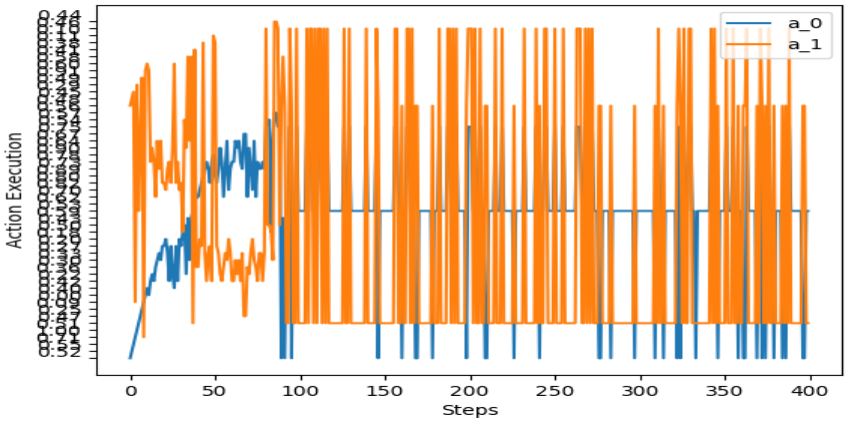

Figure 2.5: Actions