Name Aadesh Varude
RL PRoject 3: DQN Breakout game

The DQN and the duelling DQN and the double DQN have the standard network and layer structure as defined in the paper and lectures and PyTorch implementation.
Below is the network defined.
Here the only change is each layer is multiplied by 2 rather than the original layers.

```
nn.Conv2d(in_channels, 32, kernel_size=8, stride=4),
          nn.ReLU(),
          nn.Conv2d(32, 64, kernel_size=4, stride=2),
          nn.ReLU(),
          nn.Conv2d(64, 64, kernel_size=3, stride=1),
          nn.ReLU(),
nn.Linear(self.input_sz, 512),
          nn.ReLU(),
          nn.Linear(512, self.num_actions))
```

In rest implementations of Dueling DQN and Double DQN are small modifications in the train and model file.
It didn't work as per expectation hence only results are included and not the code.
In project 3 I have implemented DeepQlearning and duelling Q learning and double DQN.

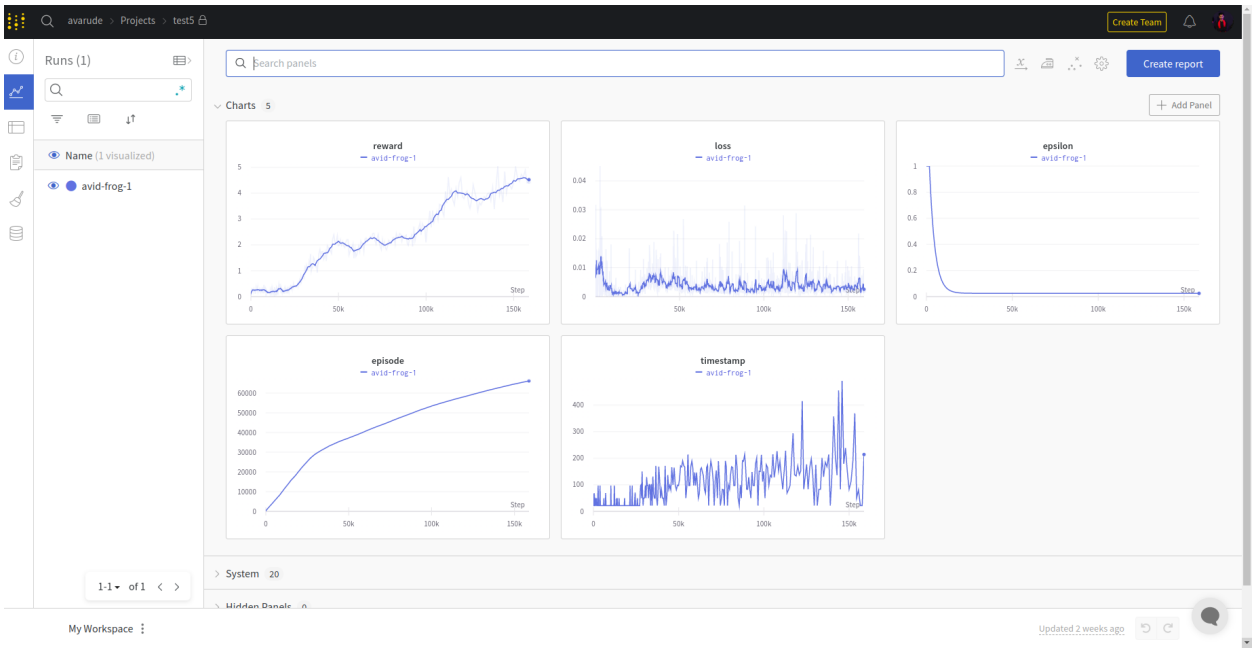
Initially for my DeepQlearning Model:
Attempt1:
I tried various parameters and epsilon decay methods but my output was below 1 and the model was training properly with optimizing the loss but couldn’t get the desired reward.
Following is the graph of the test:

Here the reward is stagnant at 0.5 and the loss is lower too.

Attempt 2:
Later I tuned the hyperparameters to obtain the reward rising the training reward rose up to 5 and then it started terminating.

```
EPISODES = 200000
LEARNING_RATE = 1.5e-4  # alpha
GAMMA = 0.99
BATCH_SIZE = 32
BUFFER_SIZE = 10000
EPSILON = 1.0
EPSILON_END = 0.025
FINAL_EXPL_FRAME = 1000000
TARGET_UPDATE_FREQUENCY = 1000
SAVE_MODEL_AFTER = 5000
DECAY_EPSILON_AFTER = 3000
```

This was my tuned parameters,
Later I made some changes in the way my optimize model was working.
But the result were still 5 reward points as showm in below:



Attempt 3 :
On the 3rd attempt, I used the sweep test which was performed using wandb wher i put in all the various parameters for which i wanted my model to train and give me good results .
`command:`

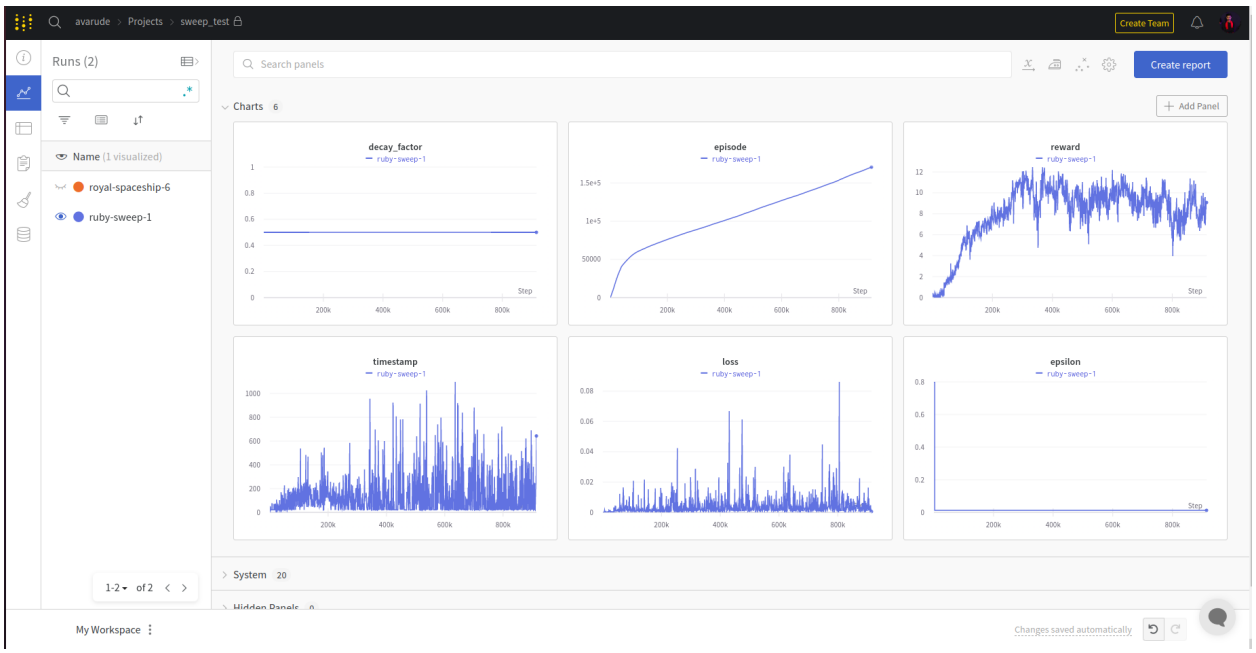
```
  - python
  - ${program}
  - --train_dqn
  - ${args}
method: random
metric:
  goal: maximize
  name: reward_buffer
parameters:
  BUFFER_SIZE:
    values:
      - 1000
      - 100
      - 500
  DECAY_EPSILON_AFTER:
    values:
      - 3000
      - 5000
      - 1000
  EPSILON:
    values:
      - 1
      - 0.6
      - 0.8
  EPSILON_DECAY:
    values:
      - 1
      - 0.8
      - 0.5
  EPSILON_END:
    values:
      - 0.025
      - 0.002
      - 0.25
  TARGET_UPDATE_FREQUENCY:
    values:
      - 1000
      - 5000
      - 500

program: main.py
```

After the sweep test the tuned parameters were:

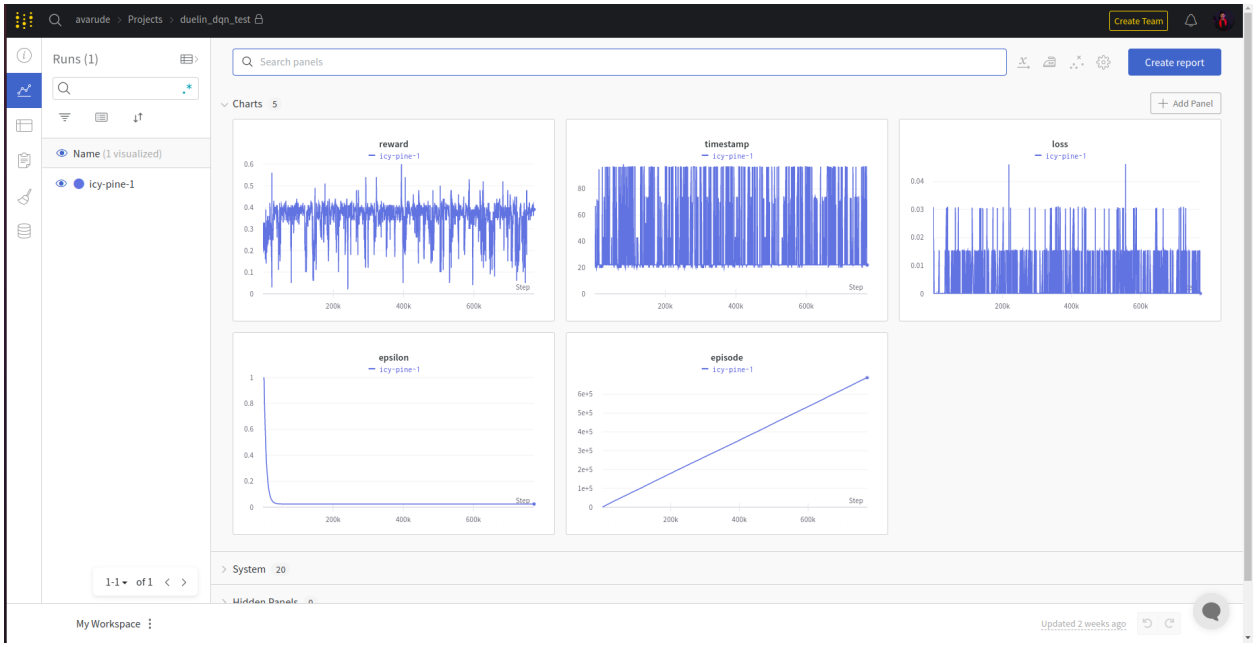
Name (1 visualized)	es	User	Tags	Created	Runtime	Sweep	BUFFER_SI	DECAY_EPS	EPSILON	EPSILON_D	EPSILON_E	TARGET_UF	decay_factc	episode	epsilon	loss	reward	timestamp
ruby-sweep-1	notes	avarud		1w ago	18h 10s	4np2xk64	500	3000	0.8	0.5	0.025	1000	0.5	170700	0.0125	0.002166	9.08	643

Results:



Attempt 4:

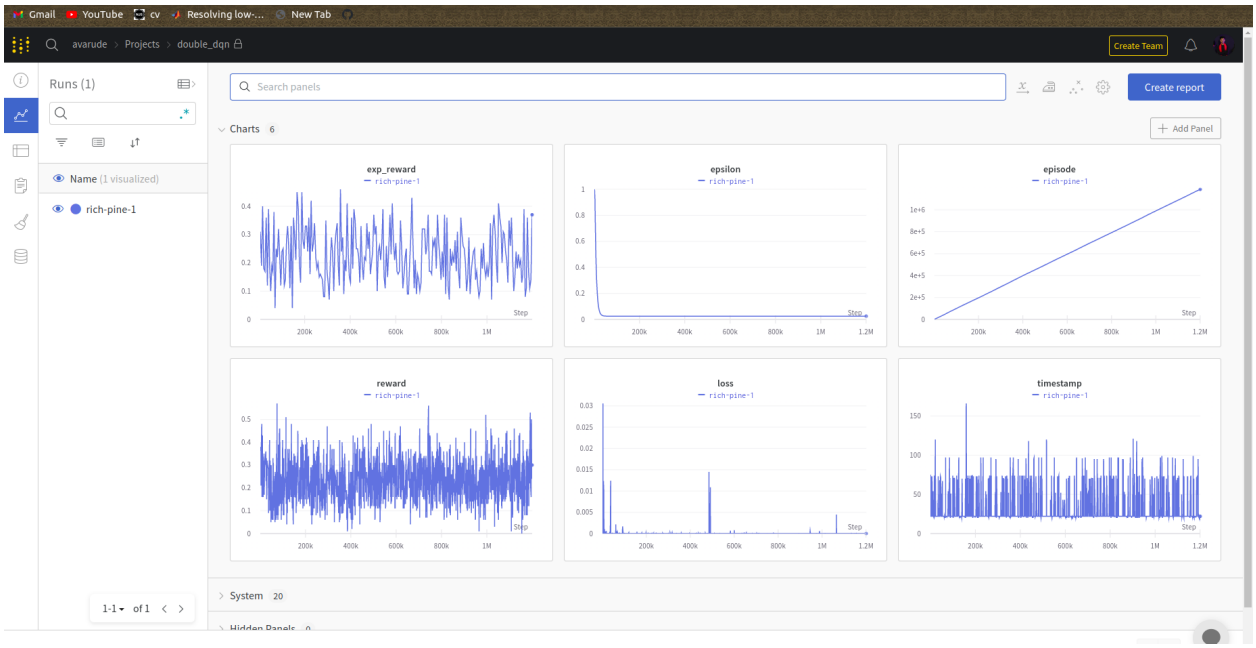
I further implemented the dueling DQN network and the results are as follows:



Here still, the rewards didn't spike more than 0.5 hence.

Attempt 5:

Further, I implemented a double DQN, still, the results were unsatisfactory:



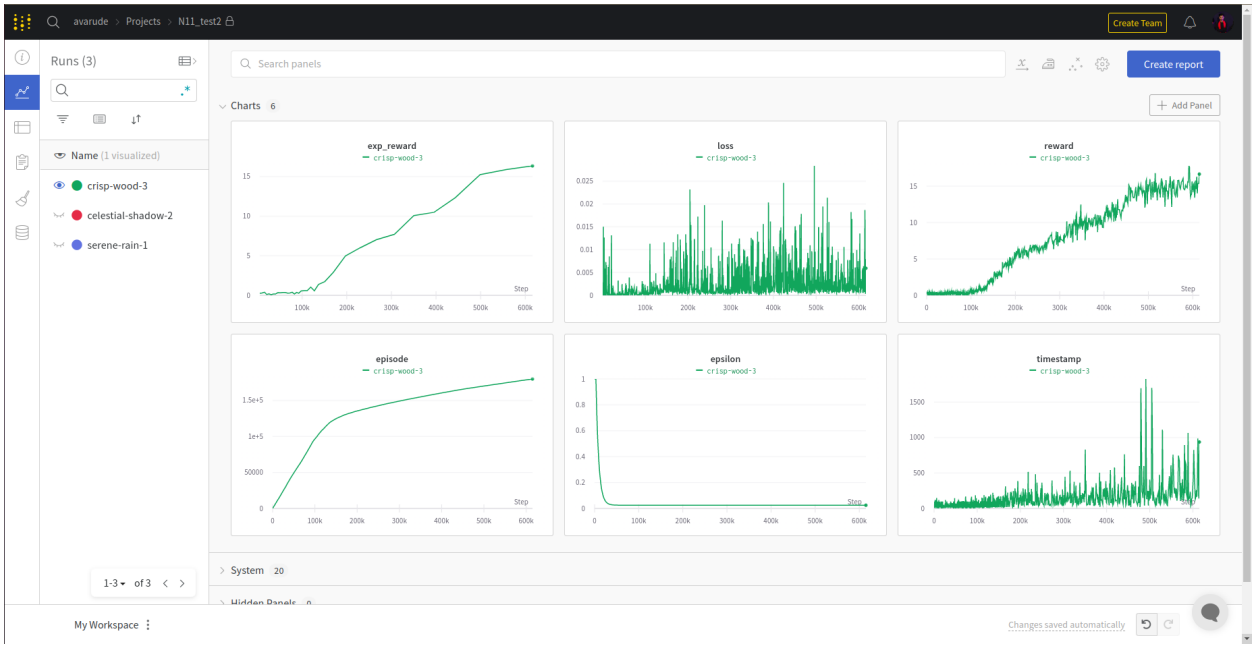
Attempt 6:
In my final attempt, I used the simple DQN and the parameter that was tuned are as follows:

```
torch.manual_seed(595)
np.random.seed(595)
random.seed(595)
#Params from the original paper
CONSTANT = 200000
GAMMA = 0.99
BATCH_SIZE = 32
BUFFER_SIZE = 10000

#epsilon pararmeters for the decay
EPSILON = 1
EPSILON_END = 0.025
DECAY_EPSILON_AFTER = 3000
#updating the model params
TARGET_UPDATE_FREQUENCY = 5000
SAVE_MODEL_AFTER = 5000
# EPSILON_DECAY=0.5
#learning rate
LEARNING_RATE = 1.5e-4
```

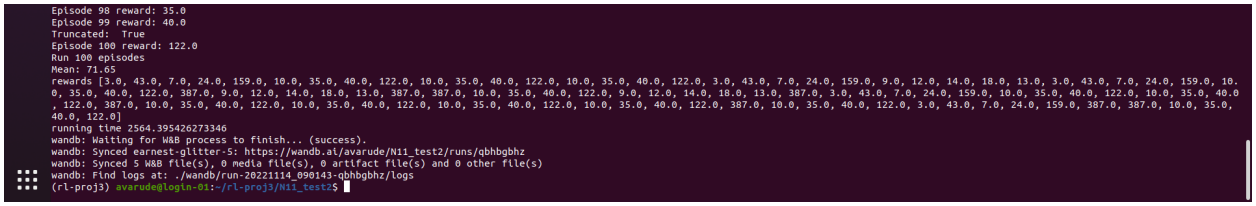
Also, I made changes to the epsilon decay method and the rate, further ii used some optimizing conditions to make it better.

Result:



In this graph, I have recorded 15 points which is the training reward.

The testing reward obtained for the same model file for 100 episodes is 71.65.



References :

https://github.com/higgsfield/RL-Adventure

https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

https://github.com/radhasaraf/ds595-rl/tree/main/Project3-DeepQlearning

https://medium.com/nerd-for-tech/reinforcement-learning-deep-q-learning-with-atari-games-63f5242440b1