RL Project 3 Report

Uthiralakshmi Sivaraman

133545470

Experiments I performed for Project3:

Architecture Model

I tried various versions of Deep Q Learning for training an agent to play Atari game from Gym environment, which are as follows.

1. Vanilla DQN (Deep Q Learning)
2. Dueling DQN
3. Double DQN
4. Dueling Double DQN

Architecture:

Model architectures are taken from Deep Mind.

DQN:

The input to the neural network consists of an 84 * 84 * 4 image produced by the preprocessing map w.

The first hidden layer convolves 32 filters of 8 * 8 with stride 4 with the input image and applies a rectifier nonlinearity

The second hidden layer convolves 64 filters of 4 * 4 with stride 2, again followed by a rectifier nonlinearity.

This is followed by a third convolutional layer that convolves 64 filters of 3 * 3 with Stride 1 followed by a rectifier.

The final hidden layer is fully connected and consists of 512 rectifier units.

The output layer is a fully connected linear layer with a single output for each valid action

```python
self.num_actions = num_actions
self.conv_layes = nn.Sequential(
    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(),
)
out_h = out_w = self._conv2d_size_out(
    self._conv2d_size_out(self._conv2d_size_out(84, 8, 4), 4, 2),
    3,
    1
)

self.in_features =int(out_h*out_w*64)
```

Duelling DQN :

Advantage Stream:

Value Stream:

```python
self.value_stream = nn.Sequential(
    nn.Linear(self.in_features, 512),
    nn.ReLU(),
    nn.Linear(512, 1)
)

self.advantage_stream = nn.Sequential(
    nn.Linear(self.in_features, 512),
    nn.ReLU(),
    nn.Linear(512, self.num_actions)
)
```

```python
values = self.value_stream(conv_output)
advantages = self.advantage_stream(conv_output)
x = values + (advantages - advantages.mean())
```

Hyperparameter Tuning:

I tried to tune various hyperparameters

1. Epsilon Start- the value of start of epsilon
2. Epsilon end- the value of end of epsilon after all episodes
3. Epsilon decay – the ratio of epsilon decay
4. Number of steps after which Epsilon was decayed
5. Target update Frequency- number after target Q network is updated
6. Learning rate
7. Total number of episodes

Fixed Parameters: Loss Function: Huber Loss Function, Optimizer: ADAM, Buffer Size (Replay buffer – sample)- 5000

The tabular column shows various models and hyperparameters I have trained.

SNO	Model	Epsilon Start	Epsilon End	Epsilon decay	Decay Epsilon after	Number of Episodes	Learning Rate
1	Vanilla DQN	0.01	0.002	500	500	1 million	1.5e^-4
2	Vanilla DQN	0.02	0.005	500	500	1 million	1.5e^-4
3	Vanilla DQN +Dueling	0.01	0.002	500	500	1 million	1.5e^-4
4	Vanilla DQN + Dueling	0.02	0.005	500	500	1 million	1.5e^-4
5	Vanilla DQN + Dueling	0.02	0.005	500	500	2 million	1.5e^-4
7	Vanilla Double DQN	0.02	0.005	500	500	2 million	1.5e^-4
8	Vanilla Double DQN	0.01	0.002	500	500	1 million	1.5e^-4
9	Vanilla Double DQN with Dueling	0.02	0.005	500	500	1.7 million	1.5e^-4
10	Vanilla Double DQN with Dueling	0.02	0.005	500	500	2 million	1.7e^-4

Trial 1: Vanilla Double DQN with Dueling

Test Mean Reward for 100 Episodes:

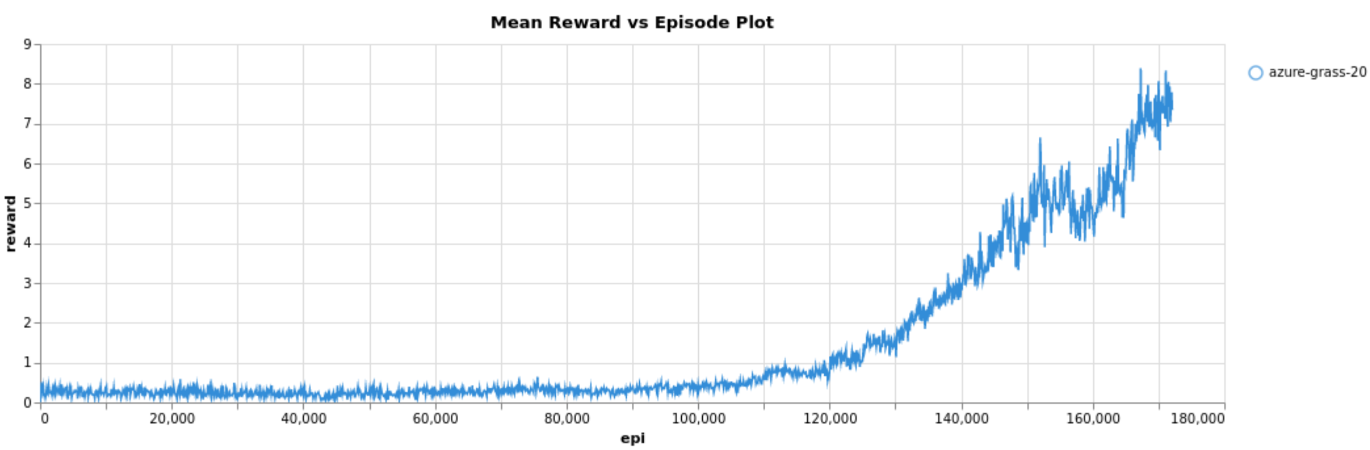
```
0: Testing ...Action : 1
0: Episode 100 reward: 11.0
0: Run 100 episodes
0: Mean: 18.33
0: rewards [6.0, 14.0, 19.0, 17.0, 172.0, 8.0, 15.0, 23.0, 138.0, 0.0, 12.0, 30.0, 22.0, 19.0, 25.0, 12.0, 7.0, 17.0, 15.0, 30.0, 6.0, 19.0, 14.0, 14.0
0: running time 143.44244575500488
```

Mean: 18.33

Trained model file: vanilla_dueldqn_model.pth

Code name: agent_dqn.py

Mean Reward vs Episode plot



Link: https://wandb.ai/usivaraman/usivaraman-train_sample/runs/fqzown1o?workspace=user-usivaraman

Hyperparameters:

Total Number of Episodes	1. 7 Million
Epsilon Start	0.02
Epsilon End	0.005
Epsilon decay after episode number	500
Learning Rate	1.5 e^-4
Target Update frequency	5000
Epsilon decay	1000
Epsilon Decay type:	Geometric Epsilon decay for each step

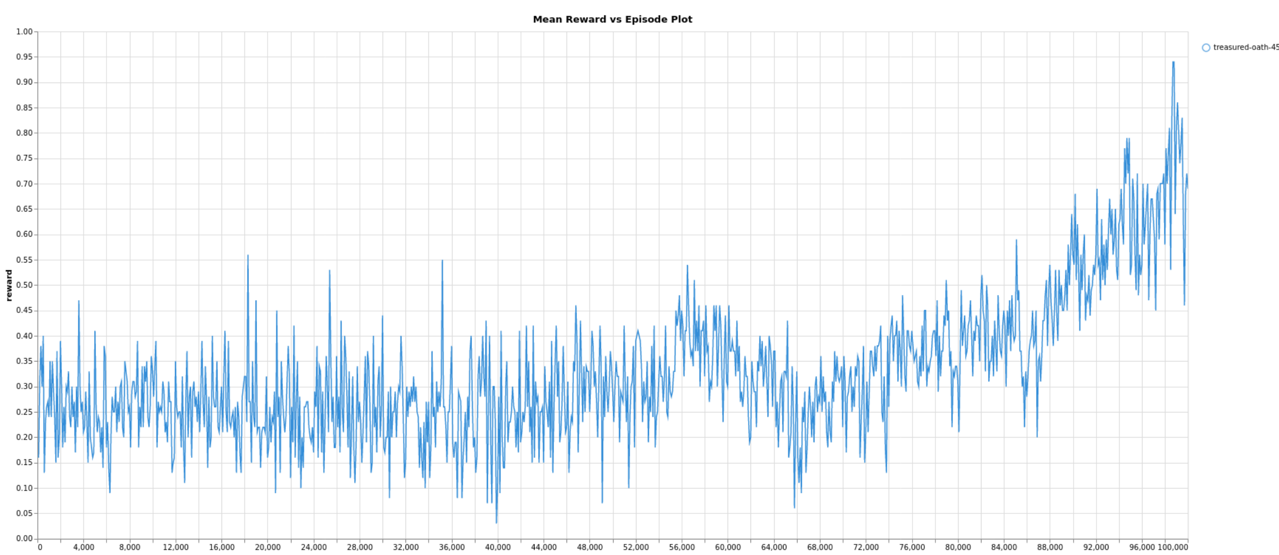
Trial 2: Vanilla DQN with Dueling

```
0: Testing ...Action : 1
0: Episode 100 reward: 1.0
0: Run 100 episodes
0: Mean: 0.39
0: rewards [0.0, 0.0, 1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 2.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.
0: running time 15.034950971603394
```

Trained model file: vanilla_dueldqn_model_revnov16_4.pth

Mean Reward: 0.39

Code name: agent_dqn_epi.py



Link: https://wandb.ai/usivaraman/usivaraman-train_sample/runs/azywnzpv?workspace=user-usivaraman

Hyperparameters:

Total Number of Episodes	1 million
Epsilon Start	1
Epsilon End	0.025
Epsilon decay after episode number	400000
Learning Rate	1.5 e^-4
Target Update frequency	5000
Epsilon decay	-
Epsilon Decay type:	Linear Epsilon decay for every episode start

Trial 3: Vanilla DQN with Dueling

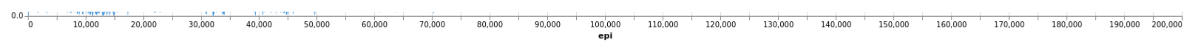
Test Mean Reward for 100 Episodes:

```
57986   0: Testing ...Action : 2
57987   0: Testing ...Action : 2
57988   0: Testing ...Action : 2
57989   0: Testing ...Action : 2
57990   0: Testing ...Action : 2
57991   0: Episode 100 reward: 56.0
57992   0: Run 100 episodes
57993   0: Mean: 22.98
57994   0: rewards [14.0, 40.0, 58.0, 29.0, 88.0, 13.0, 9.0, 16.0, 31.0, 14.0, 36.0, 9.0, 11.0, 51.0, 100
57995   0: running time 95.69260239601135
```

Mean: 22.98

Trained model file: vanilla_dueldqn_model_revnov16_2

Code name: agent_dqn_duel.py



Link: https://wandb.ai/usivaraman/usivaraman-train_sample/runs/1h7okxft?workspace=user-usivaraman

Mean Reward vs Episode plot

Hyperparameters:

Total Number of Episodes	1. 75 million
Epsilon Start	0.02
Epsilon End	0.005
Epsilon decay after episode number	500
Learning Rate	1.5 e^-4
Target Update frequency	5000
Epsilon decay	500
Epsilon Decay type:	Geometric Epsilon decay for each step
Choosing action	Load based on model

Trial 4: Vanilla DQN with Dueling

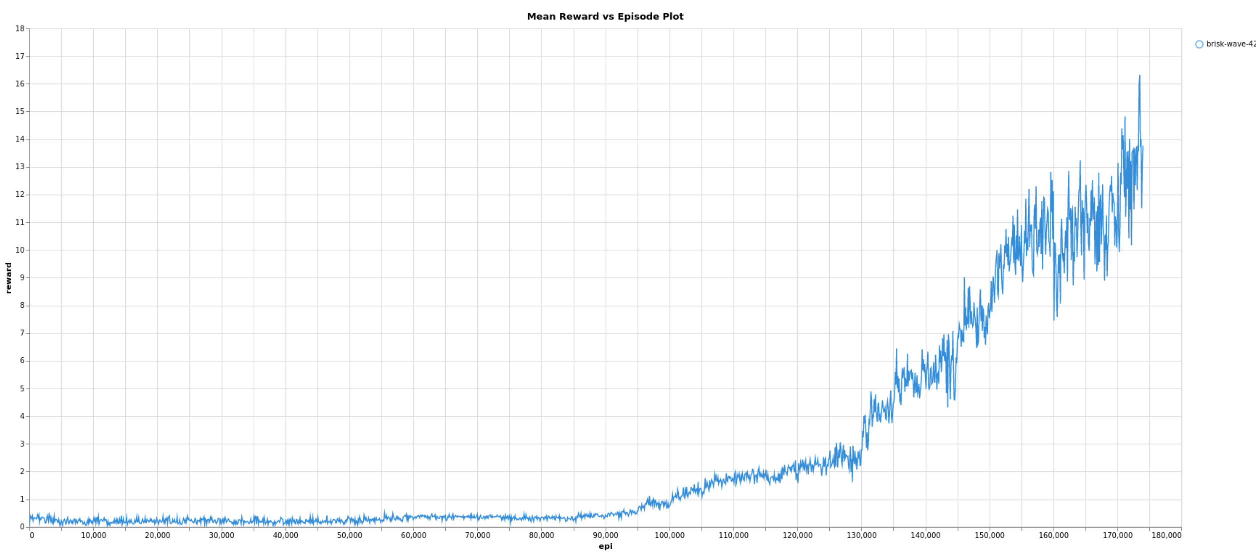
Test Mean Reward for 100 Episodes:

```
0: Testing ...Action : 2
0: Testing ...Action : 0
0: Testing ...Action : 3
0: Testing ...Action : 0
0: Episode 100 reward: 43.0
0: Run 100 episodes
0: Mean: 42.21
0: rewards [66.0, 6.0, 1.0, 52.0, 7.0, 19.0, 16.0, 52.0, 18.0, 29.0, 53.0, 1.0, 25.0, 27.0, 4.0, 7
0: running time 108.72107577323914
0: wandb: Waiting for W&B process to finish... (success).
0: wandb: Synced golden-waterfall-72: https://wandb.ai/usivaraman/usivaraman-train_sample/runs/1xu
```

Mean: 42.21 Trained model file: vanilla_dueldqn_model_revnov16_3

Code name: agent_dqn_duel.py

Mean Reward vs Episode plot



https://wandb.ai/usivaraman/usivaraman-train_sample/runs/3sk02pya?workspace=user-usivaraman

Hyperparameters:

Total Number of Episodes	1. 75 million
Epsilon Start	0.02
Epsilon End	0.005
Epsilon decay after episode number	500
Learning Rate	1.5 e^-4
Target Update frequency	5000
Epsilon decay	500
Epsilon Decay type:	Geometric Epsilon decay for each step
Choosing Action:	Load from model, if a random sample is greater than epsilon, else choose 1 out of 4 possible

Algorithm Variables:

Trial 4

Choosing Action: Type 1

```
if random.random() > EPSILON :
    observation = observation/255
    observation = observation.transpose(2,0,1)
    observation   = torch.FloatTensor(np.float32(observation)).unsqueeze(0)
    q_value = self.Q_net.forward(observation)
    action  = q_value.max(1)[1].data[0]
    action = int(action.item())
else :
     action = random.randrange(4)
```

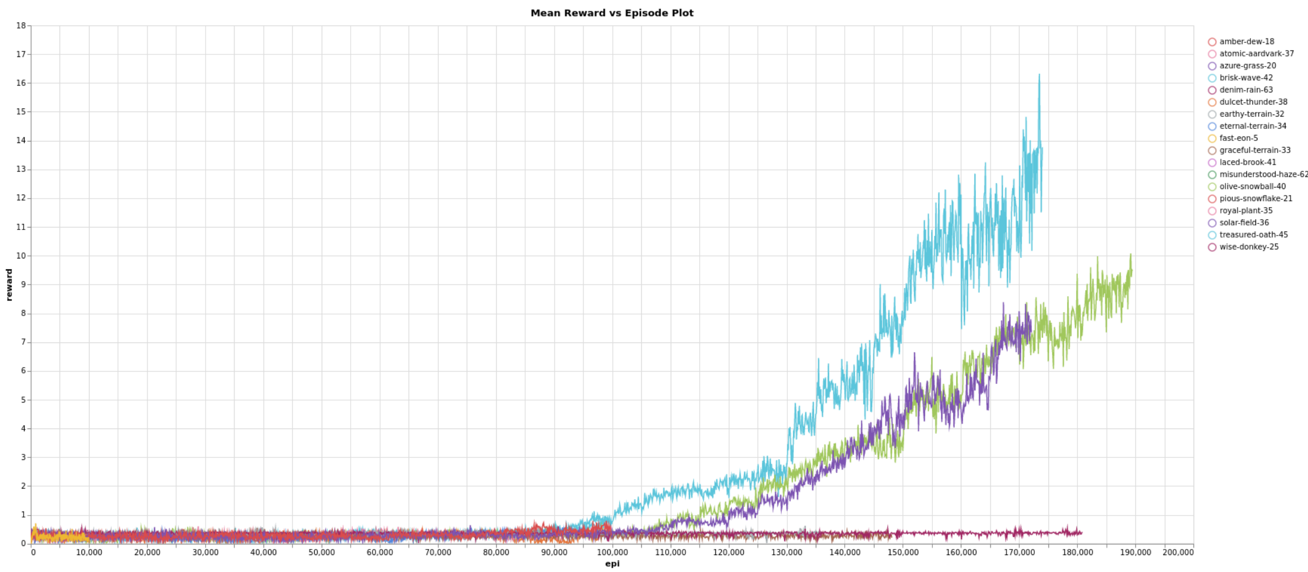
Trial 3

Choosing Action: Type 2

```
observation = observation/255
observation = observation.transpose(2,0,1)
observation   = torch.FloatTensor(np.float32(observation)).unsqueeze(0)
q_value = self.Q_net.forward(observation)
action  = q_value.max(1)[1].data[0]
action = int(action.item())
```

<u>Plots of Various DQN</u>

Overall Plot showing the Mean reward vs every 100th Episode



Violet – Vanilla Double DQN with Dueling trained for 1.7 million Episode with learning rate 1.5 e^-4

Blue – Vanilla DQN with Dueling with action selection method 1 with learning rate 1.5 e^-4

Green – Vanilla DQN with Dueling with action selection method 2 with learning rate 1.5 e^-4

Red – Vanilla Double DQN with Dueling trained for 1 million Episode with learning rate 1.5 e^-4

Orange - Vanilla Double DQN with Dueling trained for 2 million Episode with learning rate 1.5 e^-4

Others – Various Vanilla DQN and Vanilla DQN with Dueling tested for different combinations of hyperparameters

Resource: wandb

Link for plot: https://wandb.ai/usivaraman/usivaraman-train_sample?workspace=user-usivaraman

Overall Inference:

1. I found that for most of model architectures, there was increase in mean reward after 0.9 million episodes with same learning rate of 1.5 e^-4
2. Having an epsilon start of 0.02 and decaying it geometrically had better mean reward than starting with epsilon of 1 and geometrically decaying it
3. For Vanilla Double DQN with Dueling, trained over 2 million episodes, learning rate of 1.5 e^-4 gave better reward convergence than learning rate of 1.7 e^-4
4. Among the model architecture I have trained, Dueling DQN gave better reward convergence for similar hyperparameters as shown in the graph.
5. Both Vanilla Dueling DQN and Vanilla Double DQN with Dueling had better reward convergence, but Vanilla Dueling DQN had faster mean reward increase than Vanilla Double DQN with Dueling
6. Choosing actions-based sampling from model, based on constraints with current epsilon gives faster reward increase than just loading actions based on model (blue gives faster mean reward increase than green in the graph).