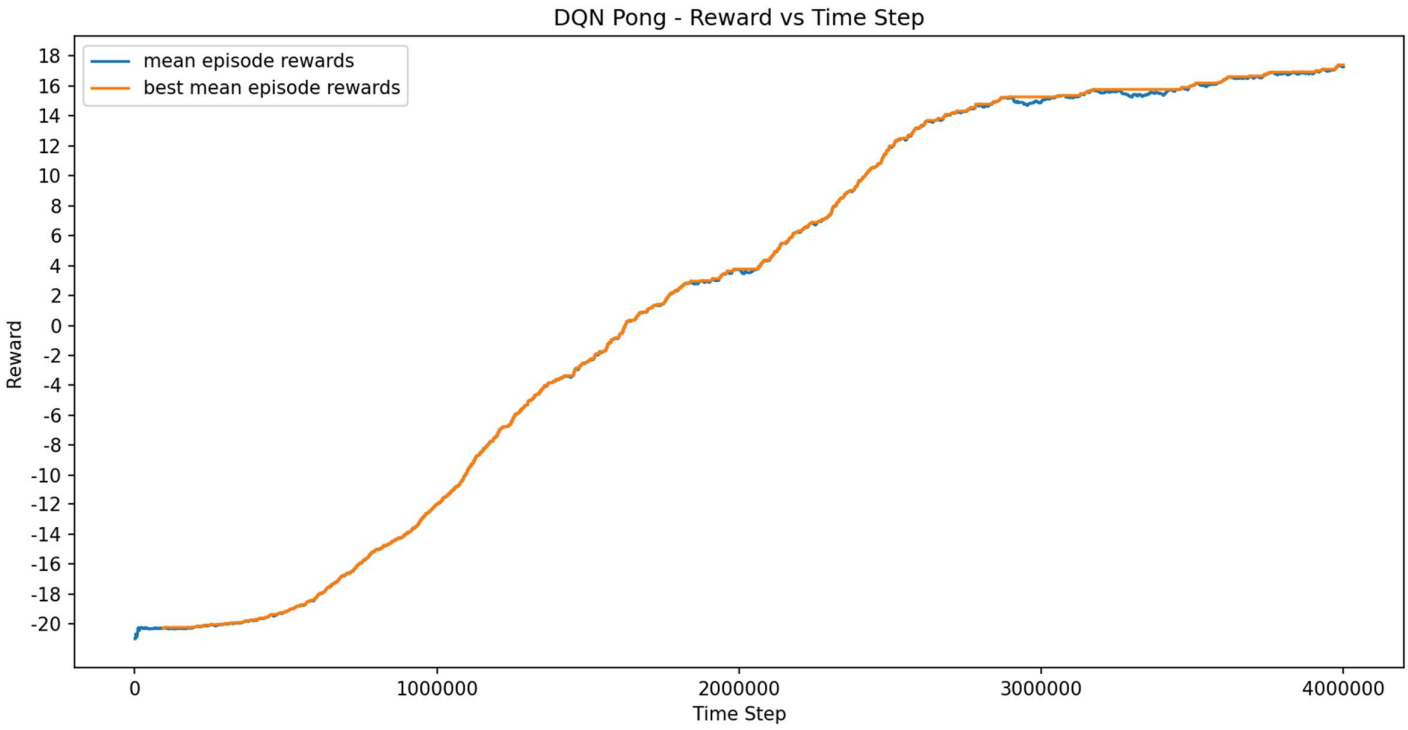
RL Project – DQN

Q1

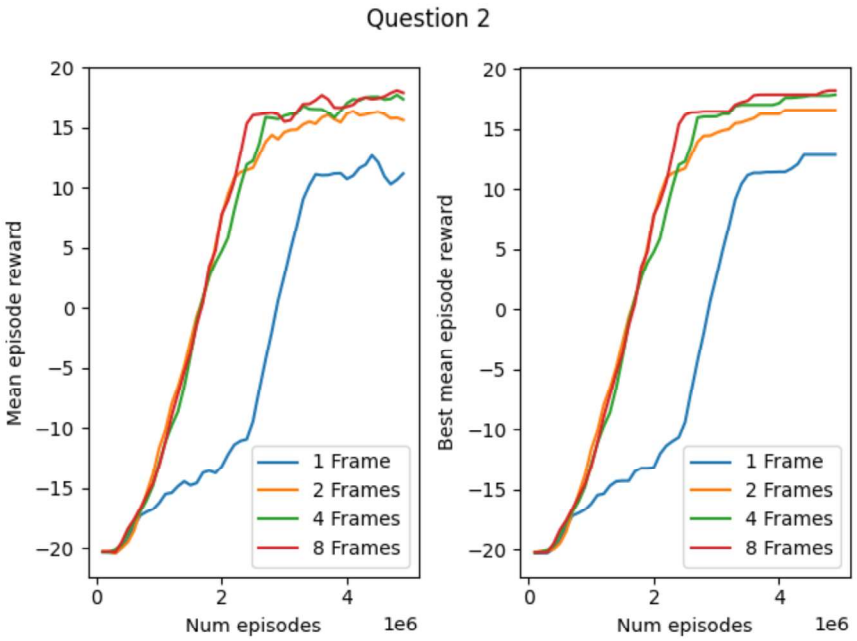
Attached is the plot with the default hyper-parameters after 5M episodes:



The best mean episode reached was 17.83 after 4,460K episodes.

Q2

We chose to experiment with the number of historic images provided as observations to the policy. The default was 4 images, we experimented with the values {1,2,8} as well. We hypothesized that the number of historic images provided is critical to generate a successful policy, and that there is a dependency on prior timesteps (thus the Markovian assumption will be invalidated if the state doesn't contain enough images). Also in a more intuitive manner, a player that's required to decide based on a single image, will have a harder time determining the trajectory of the ball compared to a player with a long sequence of images.
Attached is a plot conveying the results:



As can be seen, the best policy was indeed the one with longest history. It reached the highest mean reward of 18.16 (albeit not by a large margin) but more importantly it reached it much faster than the other policies – it reached a mean reward of 15 after 2,380K episodes vs 2,640K in the default policy and 3,230K with a 2 frame history. The single frame history reached a max reward of 12.83. This is consistent with the hypothesis that a single frame isn't adequate to determine the trajectory of the ball and hence limits the max possible score.

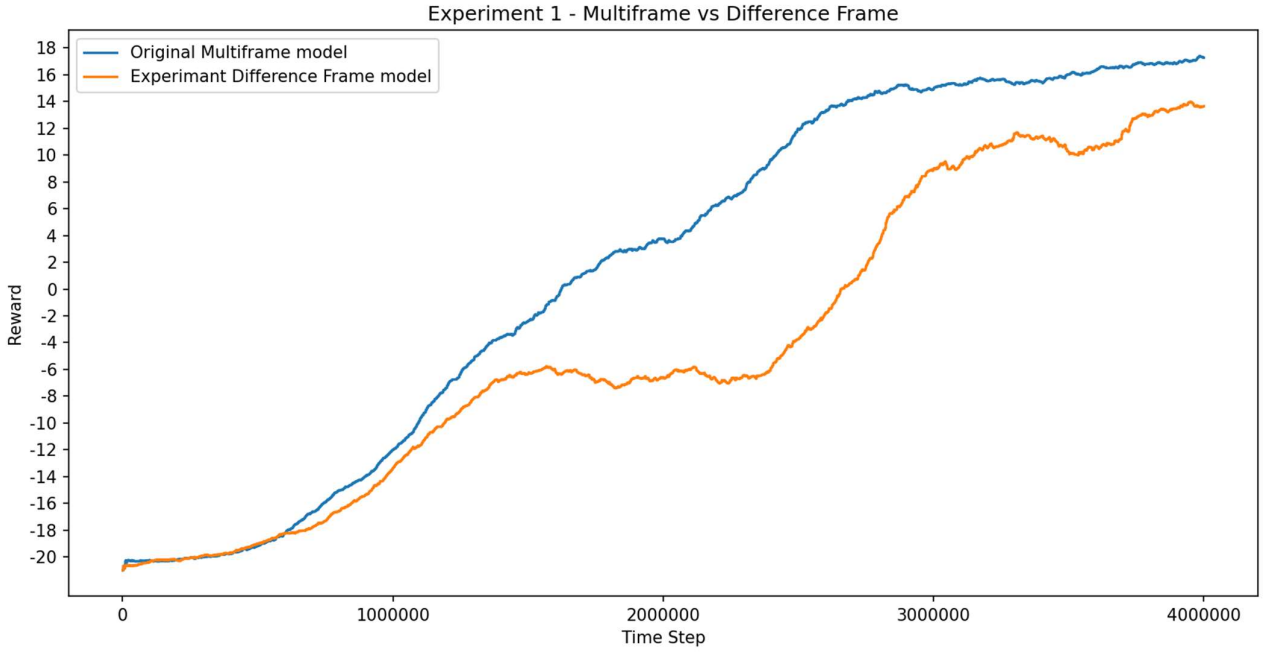
Bonus:

Experiment 1:

In the first experiment we tried to improve the efficiency of the network and reduce the training time by reducing the number of input channels. In particular, we observed that in the game Pong the difference between four consecutive frames is relatively small and maybe the network doesn't need all the background information it stores. Using the observation we passed through the network not all 4 frames in the channel dimension, but only one frame of the difference (subtraction) between them. We thought that the difference frame can encode the same amount of information and hence not degrading the learning results. Since the change between frames is small, most of the result frame was a zero tensor which enable us to use "sparse tensor" object that supposed to be more efficient.

In the plot below we compare the learning curve of the original model from question 1 together with our experiment in terms of mean episode reward. All hyperparameters were the same, the only difference is the input frame:

The results show that the original model outperforms our experiment which suggests that taking all 4 frames is better than their difference.



Experiment 2:

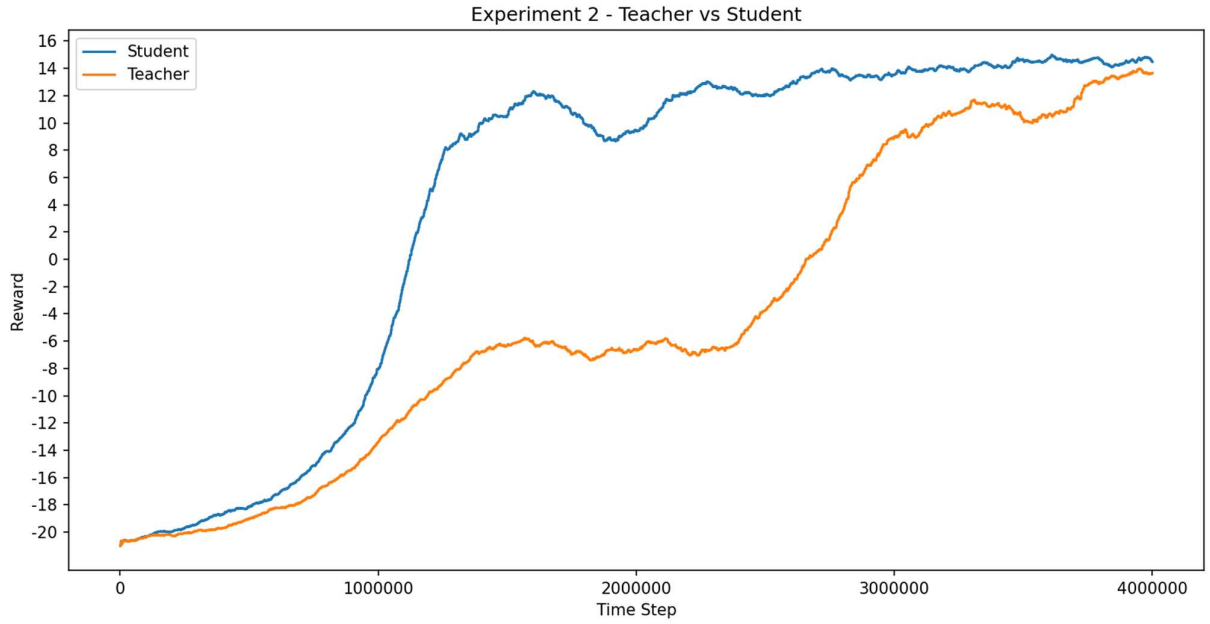
In the next experiment we tried to figure out if we can improve the model from experiment 1 by letting it "teach" a new model how to play. In particular, we took the model from the previous experiment which had best mean reward 13.8, denote it as $Q_{teacher}$. With the help

of the teacher model we trained a student model Q_{stude} . We did this by defining the output tensor of $Q_{teacher}$ to be the label of $Q_{student}$ and train it with those labels. We hoped that not only the student will be able to learn fast, but also that it will beat the teacher in terms of mean rewards.

The change in the learning process was as follows:

```
#####
optimizer.zero_grad()
states, actions, rewards, next_states, dones =
replay_buffer.sample(batch_size)
states = np.diff(states, n=3, axis=1)
q_pred = Q_student.forward(torch.from_numpy(states / 255).float())
q_label = Q_teacher.forward(torch.from_numpy(states / 255).float())
loss = F.mse_loss(q_label, q_pred)
loss.backward()
optimizer.step()
#####
```

The plot below compares the learning curves of the teacher and the student:



The results show that not only the student learns fast (which is not surprising), the best mean reward is slightly better than the teacher. It suggests that by choosing the right architecture, it is plausible to achieve better performance of a new model by using an old trained model.