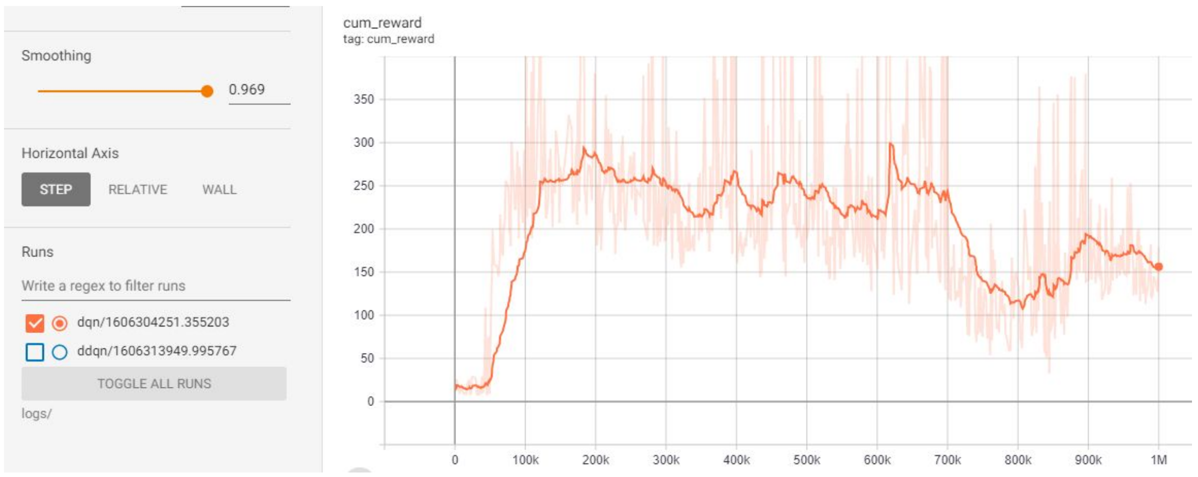
DQN and DDQN Report

Prashant Pathak MT19051

1) Run given code and generate results

We ran DQN code and generated Below graph. We can see that the cumulative reward gradually increases and reach its maximum values.

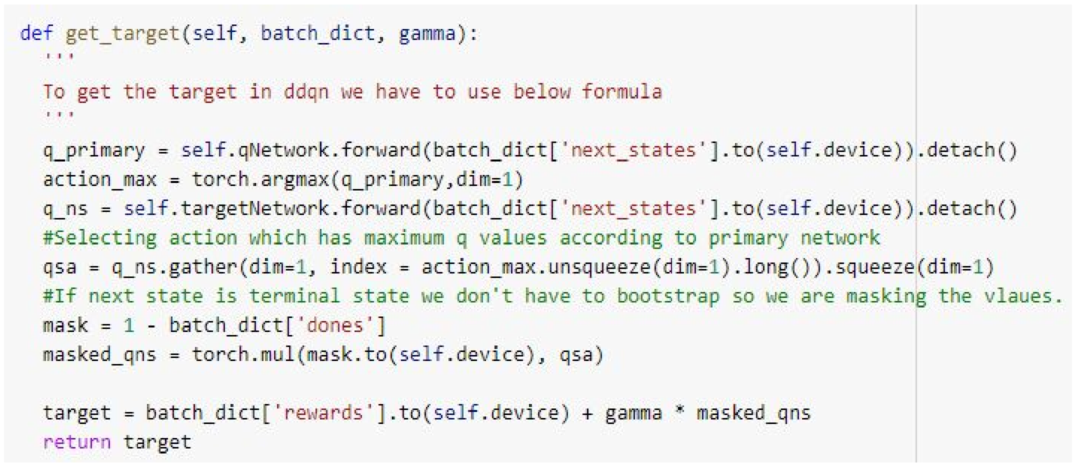


2) Edit to incorporate Double Q learning and generate results

To change DQN code to DDQN we only changed how we use to calculate target in DQN. We change it to incorporate the following formula:

$$Y^{\text{DoubleDQN}} = R_t + \gamma Q(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta), \theta^-)$$

Let θ and θ− be weights of primary and target Q-network respectively
We change below code:



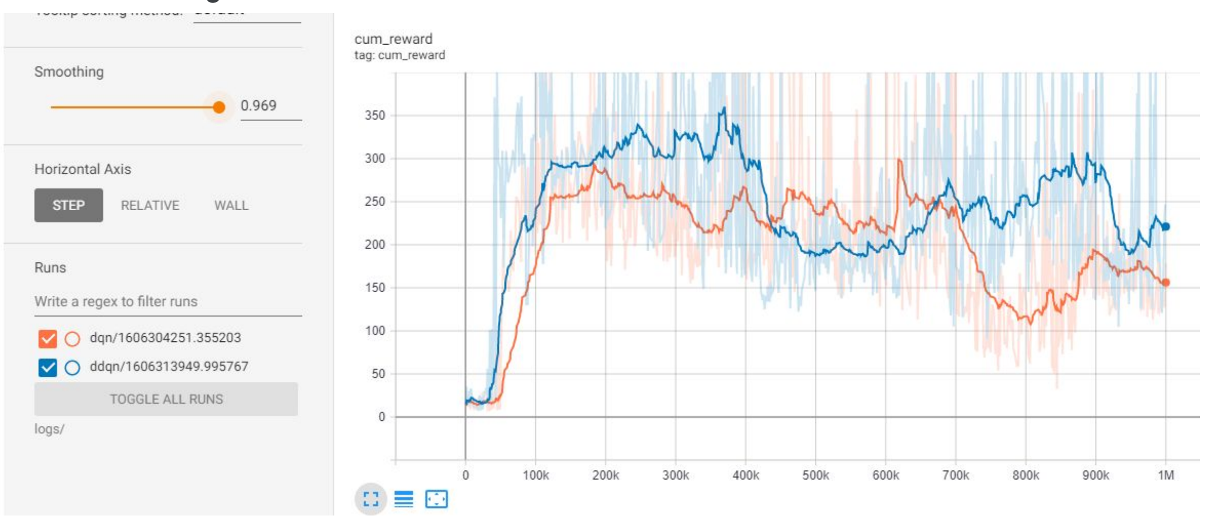
```
def get_target(self, batch_dict, gamma):
  '''
  To get the target in ddqn we have to use below formula
  '''
  q_primary = self.qNetwork.forward(batch_dict['next_states'].to(self.device)).detach()
  action_max = torch.argmax(q_primary,dim=1)
  q_ns = self.targetNetwork.forward(batch_dict['next_states'].to(self.device)).detach()
  #Selecting action which has maximum q values according to primary network
  qsa = q_ns.gather(dim=1, index = action_max.unsqueeze(dim=1).long()).squeeze(dim=1)
  #If next state is terminal state we don't have to bootstrap so we are masking the vlaues.
  mask = 1 - batch_dict['dones']
  masked_qns = torch.mul(mask.to(self.device), qsa)

  target = batch_dict['rewards'].to(self.device) + gamma * masked_qns
  return target
```

3) Compare DQN and DDQN

a) Reward

Smoothing .969



Smoothing 0.969

Horizontal Axis: STEP RELATIVE WALL

Runs

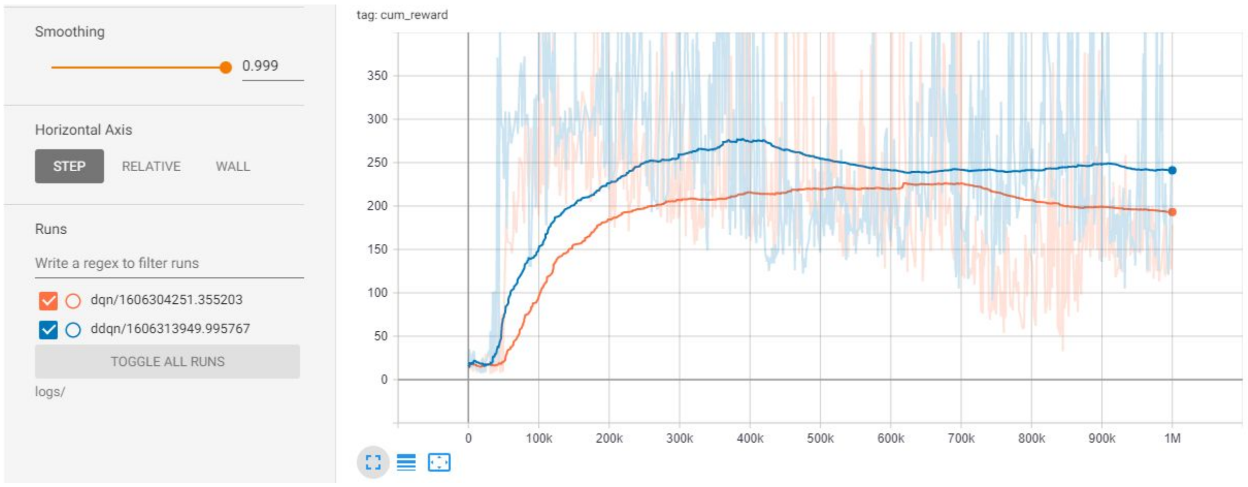
Write a regex to filter runs

- ☑ dqn/1606304251.355203
- ☑ ddqn/1606313949.995767

TOGGLE ALL RUNS

logs/

Smoothing .969



Smoothing 0.999

Horizontal Axis: STEP RELATIVE WALL

Runs

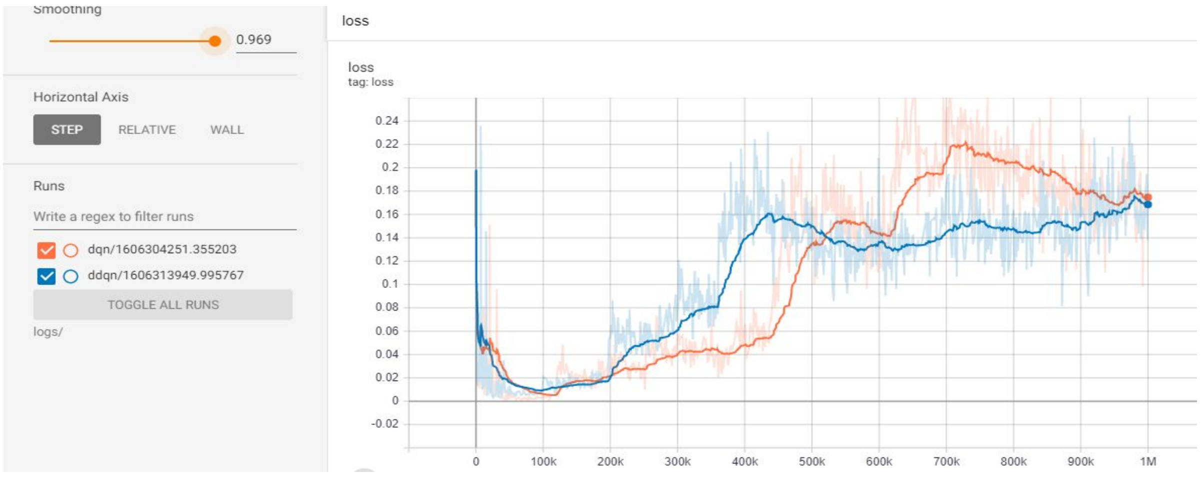
Write a regex to filter runs

- ☑ dqn/1606304251.355203
- ☑ ddqn/1606313949.995767

TOGGLE ALL RUNS

logs/

b) loss


Smoothing 0.969

Horizontal Axis: STEP

Runs: dqn/1606304251.355203, ddqn/1606313949.995767

Observation and Possible Reason

1) In both the smoothing we can see that DDQN converges faster and to better values.
   In DQN while bootstrapping we take a max over the estimates. This introduces a maximization bias in learning. This lead to longer training time and sub-optimal greedy policy.
   DDQN does not have this issue.

2) In the loss graph, we can see that value is increasing, in the end, this means that we are exploring and learning we state values. So this we why our reward first decreases and then increase.