# CS 6700 Reinforcement Learning

## Weekly written Assignment 4

**Submitted by : CS17S008 Nitesh Methani**

---

# DQN

I understood that although Q-learning is a very powerful algorithm, its main weakness is lack of generality. Q-learning resembles infact like dynamic programming. Q-learning agent does not have the ability to estimate value for unseen states. To deal with this problem, DQN get rid of the two-dimensional array by introducing Neural Network.
In order to transform an ordinary Q-Network into a DQN, following improvements were made:

1. **CNN**
   DQN leverages CNN to estimate the the Q-value function. The input to the network is the raw image of the current game situation. It went through several layers including convolutional layer as well fully connected layer.The output is the corresponding Q-value for each of the action.
   We train the network based on the Q-learning update equation. The target Q-value for Q-learning is:
   $$\nabla_j + \gamma max_{a'} \mathcal{Q}(\phi_{j+1}, a'; \theta^-)$$
   The $\phi$ is equivalent to the state $s$, while $\theta$ stands for the parameters or weights of the CNN. The loss function for the network thus is the squared error between target Q-value and the output of the network:
   $$\mathcal{L}(\theta) = (\nabla_j + \gamma max_{a'} \mathcal{Q}(\phi_{j+1}, a'; \theta^-) - \mathcal{Q}(\int, \dashv; \theta_\rangle))^2$$
   The instabilities of the RL are addressed with two key ideas:

2. **Experience Replay** :
   Since training samples in typical RL setup are highly correlated, the convergence becomes harder. This technique addresses this issue by randomly sampling from the data and thereby smoothing over changes in the data-distribution. To perform experience replay we store the agents experience $e_t = \{s_t, a_t, r_t, s_{t+1}\}$ at each time step $t$ in a dataset $\mathcal{D} = \{e_1, \ldots, e_t\}$.

3. **Separate Target Network** : This second network is used to generate the target Q-values that will be used to compute the loss for every action during training. We can't use the Q-network to update the values because at every step of training, the Q-networks values shift, and if we are using a constantly shifting set of values to adjust our network values, then the value estimations can easily get out of control. The network can become destabilized by falling into feedback loops between the target and

estimated Q-values. In order to mitigate that risk, the target networks weights are fixed, and only periodically or slowly updated to the primary Q-networks values. In this way training can proceed in a more stable manner.
Instead of updating the target network periodically and all at once, we will be updating it frequently, but slowly.

By using these techniques many challenging tasks in RL are being solved.