# DQN Agent (TD)

* Initialise the Neural Network
* Initialise a Database to store experience
  + Each instance of experience will have the following form:
  + [Initial State, Action, Immediate Reward, Next State, Terminal State Flag]
* Set up Hyperparameters for Gamma, Epsilon, Batch Size etc
* **For** N episodes**:**
  + Reset the environment
  + Calculate Epsilon based upon the number of episodes the agent has experienced.
  + **While** the episode is not Done**:**
    - Draw an action per an Epsilon Greedy policy and take the action.
    - Receive from the environment a reward and new state
    - Generate an entry for the experience database
  + **If** the required number of steps since last training have been achieved**:**
    - Sample a batch of experience from the experience database
    - For each initial state for each piece of experience in the database calculate the value of this state per the bellman equation
    - Train the network using the initial state as X and the Value (per Bellman) as Y
    - Using a squared loss function within the network, the weights are updated per:

# Advantage Actor Critic Agent

* Initialise Two Neural Networks, an Actor and a Critic
  + The Actor is responsible for modelling the policy; given a state it will return the optimal parameters of the policy
    - The policy is normal distribution, and the Actor predicts the mean of this distribution. The standard deviation is decayed manually as this is much more stable. Sigma here is equivalent to Epsilon in the DQN.
  + The Critic is responsible for modelling the value function
* Set up Hyperparameters
* Initialise an experience database
* **For** N episodes**:**
  + Reset the environment
  + Calculate Sigma and Actor learning rate based upon the number of episodes which have been experienced
  + **While** the episode is not Done**:**
    - Draw an action from the policy
    - Receive from the environment a reward and new state
    - Generate an entry for the experience database
  + At the end of the episode back-propagate the reward received across the episode. This is used instead of the bellman equation as this model learns on policy.
  + **If** the required number of steps since last training have been achieved**:**
    - Train the neural networks
    - The critic uses a standard squared loss function, and is trained with initial state as X and Monte Carlo reward as Y.
    - Calculate the advantage for each state, as
    - And then the Actor may be trained per:
    - Clear the experience database