# DQN Agent (TD)

* Initialise Two Neural Networks (one Offline / one Online)
* Initialise a Database to store experience
  + [State, Action, Immediate Reward, Next State]
* Set up Hyperparameters for Gamma, Epsilon, Batch Size etc
* Initialise the experience database by acting randomly for N episodes
* Perform initial fitting of the online network based upon the random data and immediate rewards
* Copy the Online network parameters to the offline network
* **For** N episodes**:**
  + Reset the environment
  + If the required number of episodes have been met since the Offline network was last updated, update its parameters to the Online networks
  + Calculate Epsilon based upon the number of episodes the agent has experienced.
  + **While** the episode is not Done**:**
    - Act randomly with a probability of Epsilon, or act greedily towards the frozen network with probability 1 – Epsilon
    - 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**:**
      * Train the Online (only one forward/backward pass) network based upon the last N pieces of experience and some random experience sampled from memory.
  + When the episode terminates, save the terminal reward, and the epsilon it was estimated under for display.

# Q Actor Critic Agent

* Initialise Two Neural Networks (one Offline / one Online) for Value Function approximation
* Initialise the Approximations of the Policy Function. This is typically a neural net, as it allows for much more complexity in the policy function rather than assuming normality.
* Initialise a database to store experiences as before.
* Set up Hyperparameters for Gamma, Epsilon, Batch Size, Learning Rate for each network etc
* Initialise the experience database by acting randomly for N episodes.
* Perform initial fitting of both value and policy networks based upon the random data, and immediate rewards.
* Copy the Online network parameters to the Offline networks.
* **For** N episodes**:**
  + Reset the environment
  + If the required number of episodes have been met since the Offline network was last updated, update their parameters to the Online networks
  + Calculate Epsilon based upon the number of episodes the agent has experienced.
  + **While** the episode is not Done**:**
    - Act randomly with a probability of Epsilon, or sample an action per the Offline Policy
    - 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**:**
      * Train the Online (only one forward/backward pass) networks based upon the last N pieces of experience and some random experience sampled from memory.
  + When the episode terminates, save the terminal reward, and the epsilon it was estimated under for display.