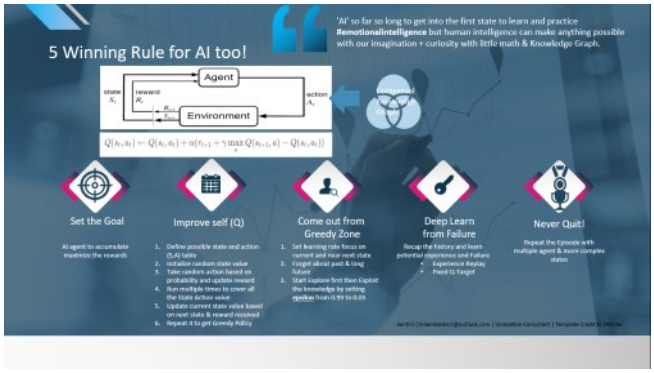
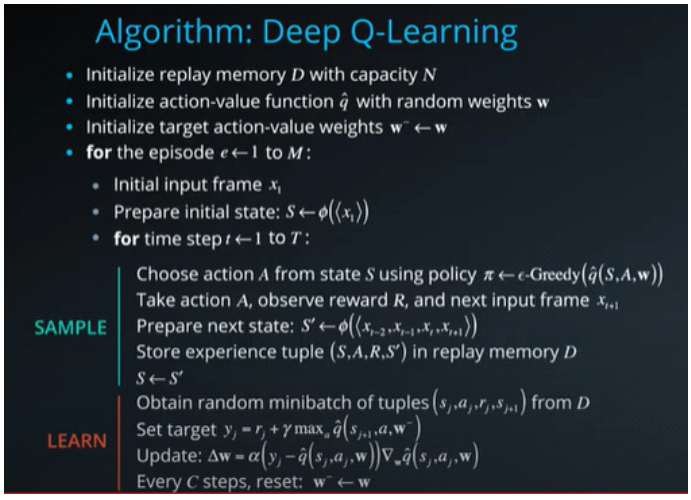
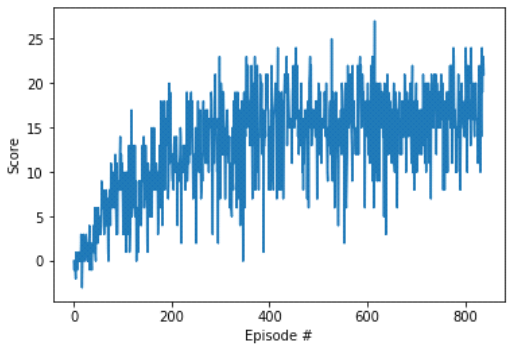
AI Navigator - Report

06 June 2020 13:11

S.No	Topic	Description
1.	Project Goal	Train a AI agent to collect min +13 rewards as a score within 100 consecutive episodes
2.	Solution Approach	1. Set the Goal for the AI - Agent 1. Initialize Agent , Network Model and integrate with environment 2. Reset the Environment and initialize 'Q table ' which help Agent to select Optimum Policy based on 'action-value' from the Q table 3. Start with zero /random step but improve self - Q(S,A) find the 'action-value' for each state and do iterative steps to find the maximum 4. Form a Greedy policy - Best Action to take to get maximize rewards 5. Explore and Exploit the environment using discounted learning , Epsilion start from .99 to 0.05 6. Store Memories and learn from failure or potential actions using Agent.Step 7. Using Fixed target and Replay memories and take function approximation using Deep Qnetwork algorithm in Simple , Here is the 5 Winning rules for AI too 
3. 1	Core Module - Qdeepnetwork (nn_model.py)	Build a Convolutional Neural Network Model with 3 layers once for input possible 37 state vectors then 64 hidden layers and 4 output layers to choose the best probable actions for the given environment state 1. Initialize and set the properties for Feedforward Convolutional Neural Network with relu as an activation function
3.2	Agent (agent.py)	Define a Ai agent with 2 important properties like • 'Qnetwork table ' to store 'action-value' matrix for each state and corresponding actions taken • self-memory to store experiences Define methods • Act • Step • Learn 
3.3	Navigator.ipynb	Python notebook contains code to train as well as test the trained agent score 1. Import Unity Environment and load the banana brain 2. Initialize AI Agent and take random action to see the score 3. Define DQN and test_run function which actually make the agent interact through Python API and train and save weights and re run 4. Set 10000 Episodes and started trained and at 800th episodes average score crossed 17 and network weights been saved after some trail and error approach by changing the epsilion value and seed value mostly 5. Once get the avg scroe >17 , loaded the weightage file and test the agent performance by setting the 'train_mode= False' while re initiating the environment 6. Tested and confirm Avg scores is 13+ within 50 consequtive episodes and results been captured 7. History shows number of trial runs and training parameters
4.	Checkpointai17.pt	Trained Neural Network weightage source to drive the the Ai Agent to collect as many as Yellow bananas. Generated based on around 800 episodes ran

5.	Rewards plot	
		`#v9.4 -High score - repeat once to get threshold 17`
		`firstAI1 = AiAgent(state_size=37,action_size=4,seed=24)`
		
		`num_episodes=10000`
		`max_t=1000`
		`eps_start=0.9786`
		`eps_end=0.0005`
		`eps_decay=0.9799`
		`scores = dqn(firstAI1,num_episodes,max_t,eps_start,eps_end,eps_decay)`
		
		`Episode 100 Average Score: 3.88`
		`Episode 200 Average Score: 9.03`
		`Episode 300 Average Score: 12.54`
		`Episode 400 Average Score: 14.14`
		`Episode 500 Average Score: 15.86`
		`Episode 600 Average Score: 15.16`
		`Episode 700 Average Score: 15.33`
		`Episode 800 Average Score: 16.35`
		`Episode 839 Average Score: 17.02`
		`Environment solved in 739 episodes! Average Score: 17.02`

	Ideas for Future work	1. Change CNN network architecture having additional hidden layer to see the impact 2. Improving the DQN Learning using Prioritized Experience Replay 3. Explore how Double DQN , Dueling DQN improve the learning plot 4. Try build own Unity Environment and store the Experience as Knowledge Graph
		THANKS TO UDACITY TEAM!