DQN Paper Review

Question 1. (150 words, 2 points) What is Experience Replay and how did the DeepMind team implement it in DQN? What are the advantages and disadvantages to using Experience Replay?

Experience replay is a biologically inspired mechanism termed that randomizes over the data, thereby removing correlations in the observation sequence and smoothing over changes in the data distribution. We know that deep neural networks require data to be independent and identically distributed. But samples obtained by the Q-learning are correlated because we take the samples obtained by the network under the current parameters to update the current network parameters. As a result, the training of the network becomes unstable. Using experience replay, we will randomly draw the samples obtained under the parameters at different moments when you update the parameters. This will make the correlation between samples small.

Specifically, experience replay

- stores the agent's experiences at each time-step $e_t = (s_t, a_t, r_t, s_{t+1})$ in a dataset $D_t = \{e_1, \dots, e_t\}$, pooled over many episodes (where the end of an episode occurs when a termi- nal state is reached) into a replay memory.
- At the network weights update step, randomly pick a $(s, a, r, s') \sim U(D)$, drawn uniformly at random from the pool of stored samples as training data.

Traditional Q-learning method is based on

Advantages:

- Each step of experience is potentially used in many weight updates, which allows for greater data efficiency.
- Learning directly from consecutive samples is inefficient, owing to the strong correlations between the samples; randomizing the samples breaks these correlations and therefore reduces the variance of the updates.
- When learning on-policy the current parameters determine the next data sample that the parameters are trained on. By using experience replay the behaviour distribution is averaged over many of its previous states, smoothing out learning and avoiding oscillations or divergence in the parameters.

Disadvantages:

- In practice, the algorithm only stores the last N experience tuples in the replay memory, and samples uniformly at random from D when performing updates. This approach is in some respects limited because the memory buffer does not differentiate important transitions and always overwrites with recent transitions owing to the finite memory size N.

- Similarly, the uniform sampling gives equal importance to all transitions in the replay memory. A more sophisticated sampling strategy might emphasize transitions from which we can learn the most, similar to prioritized sweeping.

Question 2. (150 words, 2 points) Explain what Figure 4 represents and its significance.

Figure 4 is a two-dimensional t-SNE embedding of the representations in the last hidden layer assigned by DQN to game states experienced while playing Space Invaders. The plot was generated by letting the DQN agent play for 2 h of real game time and running the t-SNE algorithm on the last hidden layer representations assigned by DQN to each experienced game state. The points are coloured according to the state values (V, maximum expected reward of a state) predicted by DQN for the corresponding game states (ranging from dark red (highest V) to dark blue (lowest V)). The screenshots corresponding to a selected number of points are shown.

The DQN agent predicts high state values for both full (top right screenshots) and nearly complete screens (bottom left screenshots) because it has learned that completing a screen leads to a new screen full of enemy ships. Partially completed screens (bottom screenshots) are assigned lower state values because less immediate reward is available. The screens shown on the bottom right and top left and middle are less perceptually similar than the other examples but are still mapped to nearby representations and similar values because the orange bunkers do not carry great significance near the end of a level.

Significance:

t-SNE tends to map the DQN representation of perceptually similar states to nearby points. Figure 4 bottom right, top left and middle shows instances in which the t-SNE algorithm generated similar embeddings for DQN representations of states that are close in terms of expected reward but perceptually dissimilar, consistent with the notion that the network is able to learn representations that support adaptive behaviour from high-dimensional sensory inputs. Furthermore, it shows that representations learned by DQN are able to generalize to data generated from policies other than its own.

Question 3. (250 words, 2 points) In algorithm 1, what is the motivation behind choosing a random action with probability e? Why not just choose the action with the most expected value? Why does the probability e decrease over time? Propose another possible scheme for choosing an action besides choosing a random action that would also satisfy this motivation and how you might implement it.

Motivation: Selecting a random action with probability e is to ensure adequate exploration of the action space. In practice, the behaviour distribution is often selected by an e-greedy policy that follows the greedy policy with probability 1-e and selects a random action with probability e.

With choosing the action with the most expected value only, we won't have enough exploration of action spaces. However, if we only explore states with the hopes of finding a better reward, we may never find a better state and never maximize our rewards. The tradeoff is commonly known as the exploration vs. exploitation problem.

Probability e decreases over time because we want to explore at the beginning and high probability helps with exploration. As the training goes, we want to do exploitation and low probability helps with that.

One alternative of using the e-greedy approach is known as the intrinsic curiosity module (ICM). This approach consists of a separate network that tries to predict a next state encoding of the state at t+1 given the action a_t and the learned encoded state s_t. It will use the difference of the predicted state encoding at s_t+1 and the learned state encoding at s_t+1 to use as the reward for the agent. This way, the agent is "curious" about what affect its actions will have on the world, and learn to explore. The ICM learns an inverse dynamics model. This means the feature encoding function learn by trying to predict the action taken given s_t and s_t+1. This way, there is an emphasis on the action that is taken during state encoding and how it affects the environment.

[https://arxiv.org/abs/1705.05363]

Question 4. (300 words, 5 points) Research and explain one of the following alternative architecture that improves on DQN: Deep Attention Recurrent Q Network, Dueling Architectures for Q-Learning, Deep Double Q-Learning Network. You should read one paper on the topic of your choice, summarize the paper, and indicate how it improves on DQN. Please cite the paper you refer to.

One of the main issues with vanilla DQN is its optimism about current states. More specifically, vanilla DQN believes the Q value of a particular action given a state is very good, but it may not be the optimal action to take. So a vanilla DQN will get stuck taking nonoptimal actions believing them to be the best. This is the main problem addressed in the deep double Q-learning network (DDQN) paper. The DDQN uses the same architecture as the vanilla DQN, but does it in a much more clever way. With a DQN, the Q value is calculated using $Q(s,a) = r(s,a) + \gamma Q(s',a)$ where s is the current state, a is the action, r is the reward function, s' is the next state given action a, and $\gamma$is the discount factor. In a deep learning approach, we use a target network that is the same as the main policy network. The weights of the target network are copied directly from the policy network and are updated frequently, but not every time step. In the case of DDQN, we take advantage of the target network more. The Q value of the DDQN will be computed as $Q(s,a) = r(s,a) + \gamma Q^t(s', argmax_a(Q(s',a)))$ where Q^t is the Q value of the target network. With this equation, we find the index of the highest Q value of the main policy model and use it to compute the value of the target network. The target network is still set by copying the weights from the policy network, so the architecture of the problem stays the same. In this case we see a much more robust model. DDQN not only finds a more proper estimation of the Q values with a much lower variance, but also outperforms vanilla DQN on the selected tasks shown in the paper.

[https://arxiv.org/abs/1509.06461]