Reinforcement Learning DQN Lab

Neo Nkosi: 2437872
Joshua Moorhead: 2489197
Naomi Muzamani: 2456718
PraiseGod Emenike: 2428608

Summary of the DQN Algorithm

The Deep Q-Network (DQN) is a reinforcement learning algorithm that approximates the Q-value function using a deep neural network to make decisions in environments with large state and action spaces. The goal is to train an agent to learn optimal policies for maximizing rewards by interacting with the environment.

Training Process (Figure 3 Explanation)

The training loop consists of the following key steps:

1. **Action Selection:** The agent chooses an action using an epsilon-greedy policy (either a random action with probability ϵ or the action with the highest predicted Q-value).
2. **Interaction with the Environment:** The agent executes the chosen action in the environment and observes the next state and the reward.
3. **Memory Update:** The transition (state, action, reward, next state, done) is stored in the replay buffer (Replay Memory).
4. **Optimization:** After every step, the agent samples a random mini-batch of transitions from the replay memory. It then updates the policy network using the Bellman equation and backpropagation to minimize the temporal difference (TD) error between the predicted and target Q-values.
5. **Target Network Update:** Periodically, the target network is updated by copying the weights from the policy network to provide more stable targets during optimization.

The DQN uses two neural networks:

- **Policy Network:** Predicts the Q-values for each action given the current state.
- **Target Network:** Provides the target Q-values during training and is updated less frequently to ensure stability.

Q-Network Architecture

The Q-Network is typically a convolutional neural network (CNN) that takes in the state representation (e.g., stacked frames in Atari games) and outputs Q-values for each possible action. The architecture generally consists of convolutional layers followed by fully connected layers, with a final layer corresponding to the number of actions in the environment.

Hyper-Parameters

Parameter	Value Used	Role Description
Batch Size	64	Determines the number of transitions sampled from the replay buffer to update the network.
Replay Buffer Size	100,000	Sets the maximum number of transitions stored in the replay memory.
Learning Rate (lr)	1e-5	Determines the step size for the gradient descent optimization used to update the Q-network.
Discount Factor (γ)	0.99	Used for the factor by which future rewards are discounted, encouraging long-term goals.
Epsilon (ϵ)	Starts at 1.0, decays to 0.1	Probability of choosing a random action, balancing exploration and exploitation.
Epsilon Decay Rate	0.000625	The rate at which epsilon decays after each episode, gradually reducing exploration. To control the decay of epsilon, this decay rate is subtracted from epsilon at each time step.
Target Update Interval	5 episodes	Frequency at which the target network is updated to match the policy network.
Number of Episodes	1500	Total number of episodes for training the agent.