1. Environment Setup

FrozenLake-v1 Environment:

A slippery gridworld provides a challenging stochastic environment: the agent navigates an area toward a goal, while trying to avoid "holes" (failure states).

Configured with is\_slippery=True, which adds randomness to the movement of the agent and makes path planning less deterministic.

First, a 10x10 random map was generated where 30% of the tiles are frozen. This setup ensures variability in state transitions and enhances the difficulty in learning optimal policies.

2. Agent Features

Deep Q-Network:

The DQN makes use of a neural network for approximating the Q-value function Q(s, a), which estimates the expected discounted reward for every action in any state.

Main Network Components:

Architecture of Network:

Four hidden layers, each containing 256 neurons, utilize LeakyReLU for activation to combat vanishing gradients and improve the dynamics of learning.

Dropout layers after each hidden layer will be set to 20% to avoid overfitting issues and ensure better generalization.

The output layer shall predict the Q-values directly for all actions.

Weights are initialized using the Xavier/Glorot Initialization method for stable training. Double DQN:

Makes use of double networks: Policy Network: It chooses the most promising action based on its current knowledge. Target Network: It values the actions for training, updated less frequently to stabilize learning. The above mechanism helps in reducing the overestimation bias in Q-value predictions. Replay Memory:

Stores past transitions tuples of state, action, reward, next\_state, and done for enabling the agent to learn from diverse experiences.

Random sampling breaks correlations between sequential experiences, making gradient updates stable. Experience Replay:

Training is done every few episodes using batches of transitions sampled from the replay buffer.

This reduces computational overhead and makes sure the variance in the data being used for training is enough.

3. Hyperparameters

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Value | Description |
| Episodes | |  | | --- | | 100,000 |  |  | | --- | |  | | |  | | --- | | Ensures sufficient time for exploration and policy convergence. |  |  | | --- | |  | |
| Max Steps | 500 | |  | | --- | | Limits each episode’s length to encourage efficient exploration. |  |  | | --- | |  | |
| Learning Rate | 0.0001 | |  | | --- | | Enables gradual, stable updates to the policy network. |  |  | | --- | |  | |
| Gamma | 0.6 | |  | | --- | | Balances short-term and long-term rewards by applying a moderate discount factor. |  |  | | --- | |  | |
| Batch Size | 64 | |  | | --- | | Processes 64 transitions per training step, balancing memory usage and statistical robustness. |  |  | | --- | |  | |
| Epsilon Decay | 0.0001 | |  | | --- | | Gradual reduction in exploration, enabling the agent to shift towards exploitation as it learns. |  |  | | --- | |  | |
| Replay Memory Size | 500,000 | Stores a large number of transitions, ensuring diverse training samples. |

4. Evaluation Criteria

Performance Metrics:

Cumulative Rewards: Total rewards collected by the agent over many episodes show its overall effectiveness.

Success Rate: How often the agent reaches the goal within the episode.

Learning Curve: Is plotted to track improvements in the agent's performance over time, indicating convergence toward optimal policies.

Exploration versus Exploitation: Balancing trying new actions, exploration, with leveraging learned actions, exploitation. Monitored through the agent's epsilon decay schedule and action-selection trends.

Adaptability: Evaluates the capability of an agent to adapt to the stochastic dynamics in an environment, such as slipping movements and transitions of unpredictable states.

Consistency: This addresses the stability of the learned policy-making agent, where its performance should be dependable in various episodes with different map configurations.