📝 理解報告：基本 DQN 與經驗回放機制

🔷 目標

了解並實作一個**基本的深度 Q 網路（Deep Q-Network, DQN）**，應用在簡單的強化學習環境中，並透過**經驗回放（Experience Replay）**來穩定訓練過程。

🔶 環境

一般會使用 **CartPole-v1**（來自 OpenAI Gym）作為基本 DQN 的測試環境，因為它簡單、明確，目標是讓機器人平衡桿子不倒。

🔶 核心組件說明

1. 深度 Q 網路（DQN）

- DQN 使用神經網路近似 Q 函數。
- Agent 根據以下策略選擇行動：

$$a = \arg\max_a Q(s, a; \theta)$$

- 訓練時的目標 Q 值由 Bellman 方程式給出：

$$Q_{\text{target}} = r + \gamma \max_a Q(s', a'; \theta^-)$$

其中 θ^-是目標網路的參數（與主網路分開）。

2. 神經網路架構（範例）

```
class DQN(nn.Module):
    def __init__(self, state_dim, action_dim):
        super(DQN, self).__init__()
```

```
        self.net = nn.Sequential(
            nn.Linear(state_dim, 128),
            nn.ReLU(),
            nn.Linear(128, action_dim)
        )

    def forward(self, x):
        return self.net(x)
```

◆ 經驗回放（Experience Replay Buffer）

- 儲存 agent 在環境中經歷的資料：
  (state, action, reward, next_state, done)
- 每次訓練時從記憶庫中隨機抽樣，能打破資料之間的相關性，增加訓練穩定性。

實作範例：

```
class ReplayBuffer:
    def __init__(self, capacity):
        self.buffer = deque(maxlen=capacity)

    def add(self, state, action, reward, next_state, done):
        self.buffer.append((state, action, reward, next_state, done))

    def sample(self, batch_size):
        batch = random.sample(self.buffer, batch_size)
        states, actions, rewards, next_states, dones = zip(*batch)
        return np.array(states), actions, rewards,
np.array(next_states), dones

    def __len__(self):
        return len(self.buffer)
```

◆ 訓練流程簡述

1. 初始化主網路與目標網路。
2. 使用 ε-greedy 策略與環境互動（隨機與最大 Q 值混合選擇）。
3. 將每一步資料存入 replay buffer。
4. 每個訓練步驟從 buffer 中隨機抽樣，計算 Q 值與損失：

$$\text{Loss} = \text{MSE}(Q_{\text{pred}}, Q_{\text{target}})$$

5. 更新主網路參數，並定期將參數同步到目標網路。

✅ 小結

基本 DQN 搭配經驗回放機制，能有效解決傳統 Q-learning 的不穩定問題。透過隨機抽樣過去經驗，能提升學習效率與穩定性，是強化學習中非常重要的基礎技術之一，特別適用於離散動作空間的問題。