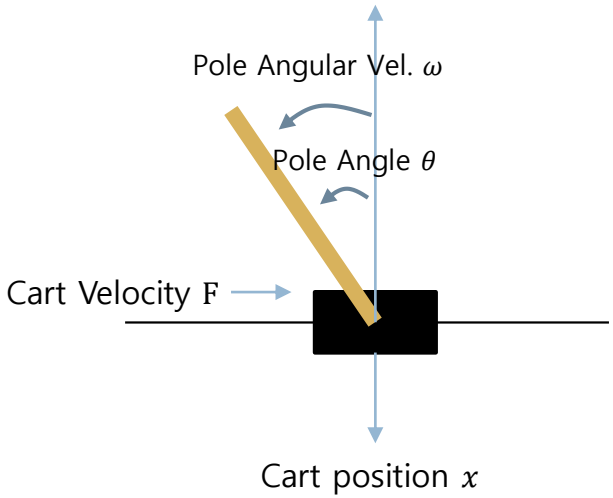
Cart Pole Example

- **Cart Pole Example**
  - https://gymnasium.farama.org/environments/classic_control/cart_pole/
- **Action space:**
  - 0: Push cart to the left
  - 1: Push cart to the right
- **Observation Space**
  - it returns 4 values with shape (4,)

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -0.418 rad (-24°)	~ 0.418 rad (24°)
3	Pole Angular Velocity	-Inf	Inf



- **Rewards**
  - a reward of +1 for every step taken

Cart Pole Example

- **1. 라이브러리 불러오기**

```
import gymnasium as gym
import numpy as np
import random
from collections import deque
import torch
import torch.nn as nn
import torch.optim as optim
```

- **2. 하이퍼파리미터 설정**

```
env = gym.make("CartPole-v1")

STATE_SIZE = env.observation_space.shape[0]
ACTION_SIZE = env.action_space.n

# 하이퍼파라미터
LR = 1e-3           # learning rate
GAMMA = 0.99        # discount factor
EPSILON = 1.0       # initial exploration rate
EPSILON_MIN = 0.05  # minimum epsilon
EPSILON_DECAY = 0.995
BATCH_SIZE = 64
MEMORY_SIZE = 10000
TARGET_UPDATE = 100  # target network 갱신 주기
NUM_EPISODES = 300

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print('device:{}'.format(device))
```

Cart Pole Example

- **3. Q-Network 정의**

```
class QNetwork(nn.Module):
    def __init__(self, state_size, action_size):
        super(QNetwork, self).__init__()
        self.fc1 = nn.Linear(state_size, 64)
        self.fc2 = nn.Linear(64, 64)
        self.fc3 = nn.Linear(64, action_size)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = torch.relu(self.fc2(x))
        return self.fc3(x)

    def sample_action(self, state, eps):
        if random.random() < eps:
            return random.randint(0,1)
        else:
            state = torch.FloatTensor(state).to(device)
            out = self.forward(state)
            return out.argmax().item()
```

Cart Pole Example

- **4. Replay Buffer**

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

    def store(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.vstack(states),
                np.array(actions),
                np.array(rewards),
                np.vstack(next_states),
                np.array(dones))

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

Cart Pole Example

- **5. 학습용 함수 정의**

```
# Q 네트워크 두 개 정의: online / target
q_net = QNetwork(STATE_SIZE, ACTION_SIZE).to(device)
target_net = QNetwork(STATE_SIZE, ACTION_SIZE).to(device)
target_net.load_state_dict(q_net.state_dict())  # 동일하게 초기화


optimizer = optim.Adam(q_net.parameters(), lr=LR)
memory = ReplayBuffer(MEMORY_SIZE)
```

Cart Pole Example

- **6. 학습 함수**

```
def train_step():
    if memory.size() < BATCH_SIZE:
        return

    states, actions, rewards, next_states, dones = memory.sample(BATCH_SIZE)

    states = torch.FloatTensor(states).to(device)
    actions = torch.LongTensor(actions).unsqueeze(1).to(device)
    rewards = torch.FloatTensor(rewards).unsqueeze(1).to(device)
    next_states = torch.FloatTensor(next_states).to(device)
    dones = torch.FloatTensor(dones).unsqueeze(1).to(device)

    # 1 Q(s,a) 예측값
    q_values = q_net(states).gather(1, actions)

    # 2 target 값 계산 (Fixed Q-Target)
    with torch.no_grad():
        next_q_values = target_net(next_states).max(1, keepdim=True)[0]
        target = rewards + (1 - dones) * GAMMA * next_q_values

    # 3 손실 계산
    loss = nn.MSELoss()(q_values, target)

    # 4 경사하강법으로 업데이트
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()
```

Cart Pole Example

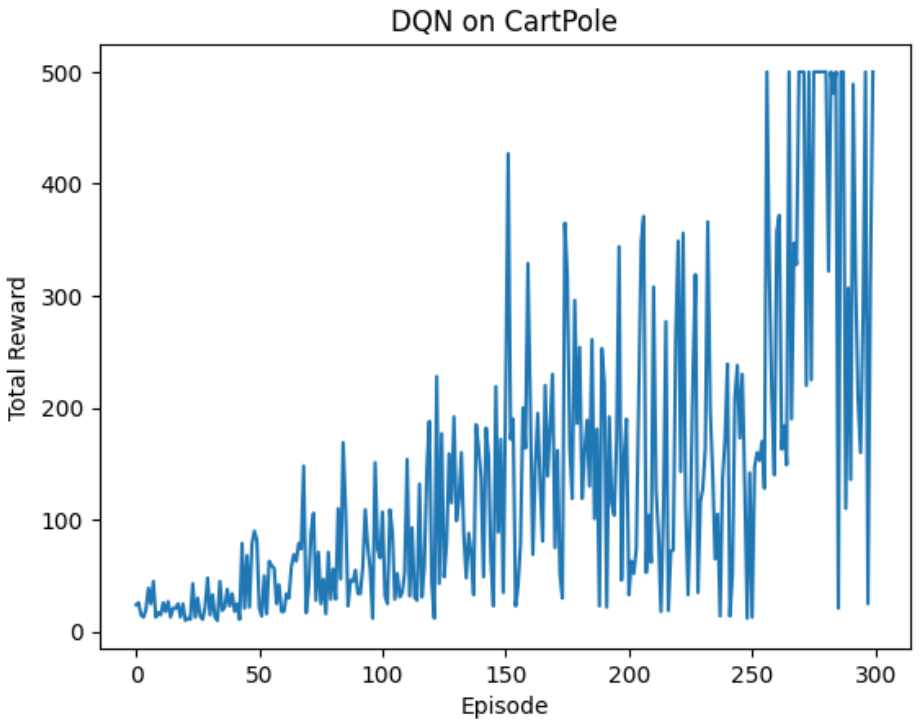
- **7. 학습 루프**

```
returns = []

for episode in range(NUM_EPISODES):
    state, _ = env.reset()
    total_reward = 0

    for t in range(500):
        action = q_net.sample_action(state, EPSILON)
        next_state, reward, terminated, truncated, _ = env.step(action)
        done = terminated or truncated  # 에피소드 종료 여부 통합

        memory.store(state, action, reward, next_state, done)   # 경험 저장

        state = next_state
        total_reward += reward

        train_step()  # 한 스텝 학습

        # 타깃 네트워크 동기화
        if t % TARGET_UPDATE == 0:
            target_net.load_state_dict(q_net.state_dict())

        if done:
            break

    EPSILON = max(EPSILON_MIN, EPSILON * EPSILON_DECAY)
    returns.append(total_reward)
    print(f"Episode {episode+1} | Reward: {total_reward:.1f} | Epsilon: {EPSILON:.3f}")
```

Cart Pole Example

- **8. 학습결과 시각화**

```
import matplotlib.pyplot as plt


plt.plot(returns)
plt.xlabel("Episode")
plt.ylabel("Total Reward")
plt.title("DQN on CartPole")
plt.show()
```



Cart Pole Example

- **9. 시뮬레이션**

```
import time
env = gym.make("CartPole-v1", render_mode='human')
state, info = env.reset()


for i in range(300):
    action = q_net.sample_action(state, 0)
    next_state, reward, terminated, truncated, info = env.step(action)
    env.render()


    time.sleep(0.01)


    state = next_state
    if terminated:
        state, info = env.reset()
env.close()
```