R E P O R T

[인공지능특강] Homework2

과목명	인공지능특강
분반	1 분반
교수	한연희
학번	2331036013
이름	조 재 민
제출일	2024년 06월 07일 금요일

서론) Homework의 내용과 목적

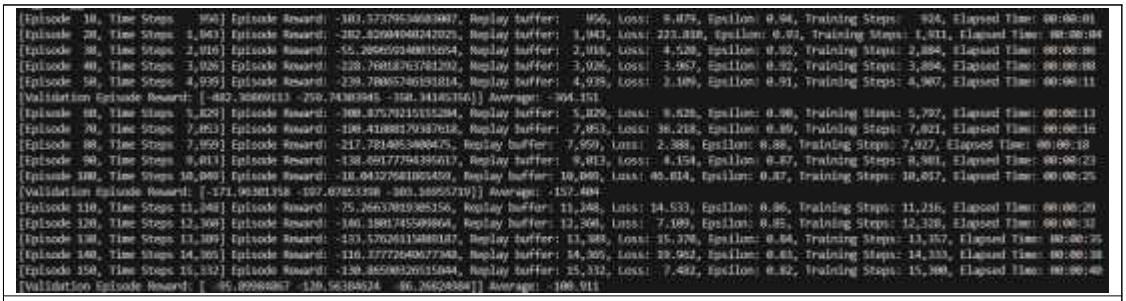
본 과제의 목적은 Deep Q-Network(DQN) 알고리즘의 파생 모델인 Double DQN과 Dueling DQN을 실제 코드로 구현하고, 이들의 성능을 비교하는 데 있다. 이러한 비교를 통해 각 알고리즘의 장단점을 명확히 파악하고, 다양한 강화 학습 문제에서의 적용 가능성을 평가하고자 한다. 각 알고리즘의 공정한 평가를 위하여 각각 학습을 5회 수행하며, 하이퍼파라미터는 모두 동일하게 설정하였다.

```
config = {
    "env_name": ENV_NAME,                                  # 환경의 이름
    "max_num_episodes": 1_500,                             # 훈련을 위한 최대 에피소드 횟수
    "batch_size": 32,                                      # 훈련시 배치에서 한번에 가져오는 랜덤 배치 사이즈
    "learning_rate": 0.0001,                               # 학습률
    "gamma": 0.99,                                         # 감가율
    "steps_between_train": 1,                              # 훈련 사이의 환경 스텝 수
    "target_sync_step_interval": 500,                      # 기존 Q 모델을 타깃 Q 모델로 동기화시키는 step 간격
    "replay_buffer_size": 300_000,                          # 리플레이 버퍼 사이즈
    "epsilon_start": 0.95,                                 # Epsilon 초기 값
    "epsilon_end": 0.01,                                   # Epsilon 최종 값
    "epsilon_final_scheduled_percent": 0.75,               # Epsilon 최종 값으로 스케줄되는 마지막 에피소드 비율
    "print_episode_interval": 10,                          # Episode 통계 출력에 관한 에피소드 간격
    "train_num_episodes_before_next_test": 50,                          # 검증 사이 마다 각 훈련 episode 간격
    "validation_num_episodes": 3,                      # 검증에 수행하는 에피소드 횟수
    "episode_reward_avg_solved": 200,                  # 훈련 종료를 위한 검증 에피소드 리워드의 Average
}
```

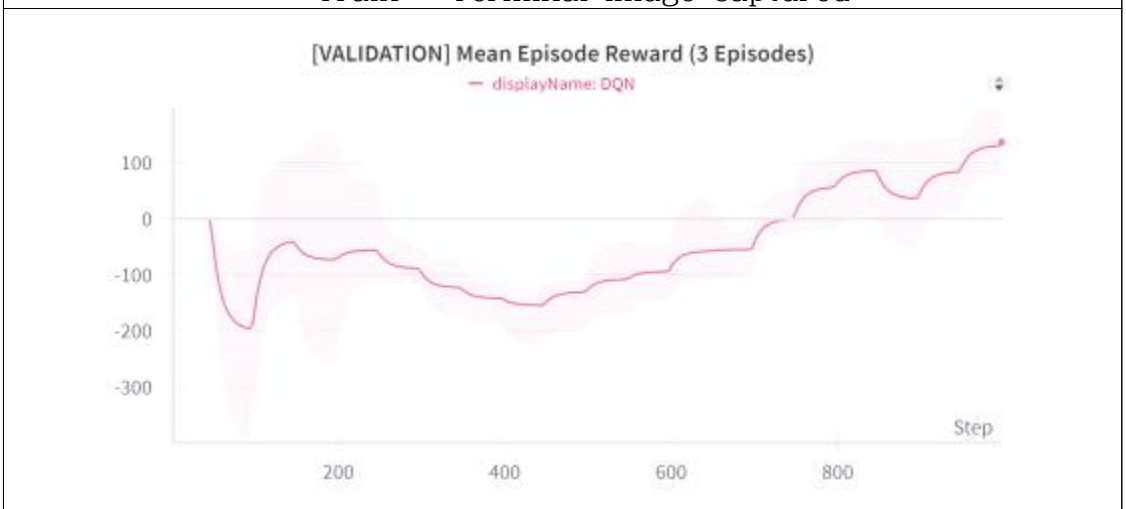
<하이퍼파라미터 설정값>

본론)

본론1_DQN)



Train - Terminal image captured



[VALIDATION] Mean Episode Reward (3 Episodes)

학습 5회 수행, 스무딩 강도 : 0.5

DQN 학습 과정은 Homework1에서 수행했던 코드와 동일하게 진행했으므로 설명은 생략함

본론2_Double_DQN)

$$Q_\theta(s,a) \leftarrow Q_\theta(s,a) + \alpha(r + \gamma \max_{a'} Q_{\bar{\theta}}(s',a') - Q_\theta(s,a))$$

```
with torch.no_grad():
    q_prime_out = self.target_q(next_observations)
    # next_state_values.shape: torch.Size([32, 1])
    max_q_prime = q_prime_out.max(dim=1, keepdim=True).values
    max_q_prime[dones] = 0.0

    # target_state_action_values.shape: torch.Size([32, 1])
    targets = rewards + self.gamma * max_q_prime
```

<DQN의 Q값 추정 방식>

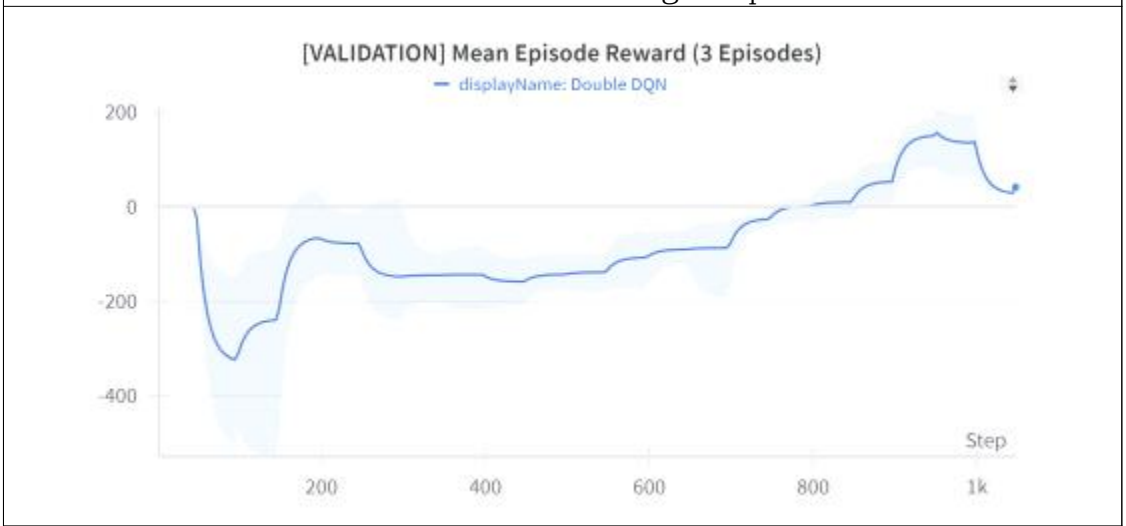
↓

$$Q_\theta(s,a) \leftarrow Q_\theta(s,a) + \alpha(r + \gamma Q_{\bar{\theta}}\left(s', arg\max_{a'} Q_\theta(s',a')\right) - Q_\theta(s,a))$$

```
with torch.no_grad():
    q_prime_out = self.target_q(next_observations)
    # next_state_values.shape: torch.Size([32, 1])
    next_actions = q_prime_out.argmax(dim=1, keepdim=True)

    q_target_out = self.target_q(next_observations)

    max_q_prime = q_target_out.gather(dim=-1, index=next_actions)
    max_q_prime[dones] = 0.0

    # target_state_action_values.shape: torch.Size([32, 1])
    targets = rewards + self.gamma * max_q_prime
```

<Double DQN의 Q값 추정 방식>

Train - Terminal image captured



[VALIDATION] Mean Episode Reward (3 Episodes)

학습 5회 수행, 스무딩 강도 : 0.5

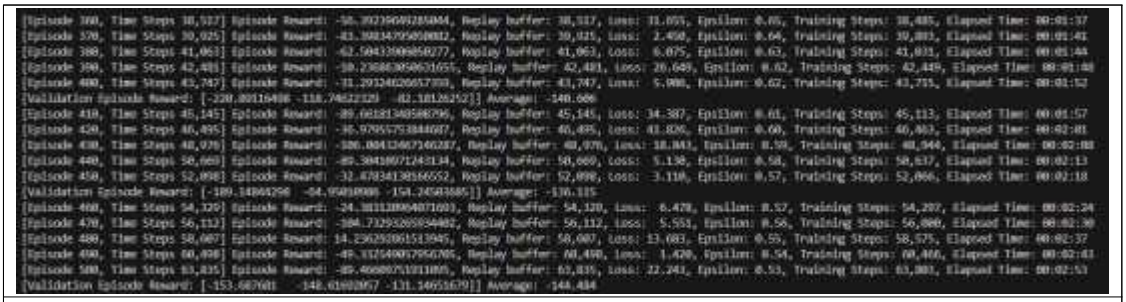
본론3_Dueling_DQN)

$$Q(s,a) = V(s) + \left(A(s,a) - \frac{1}{|A|}\sum_{a'} A(s,a')\right)$$

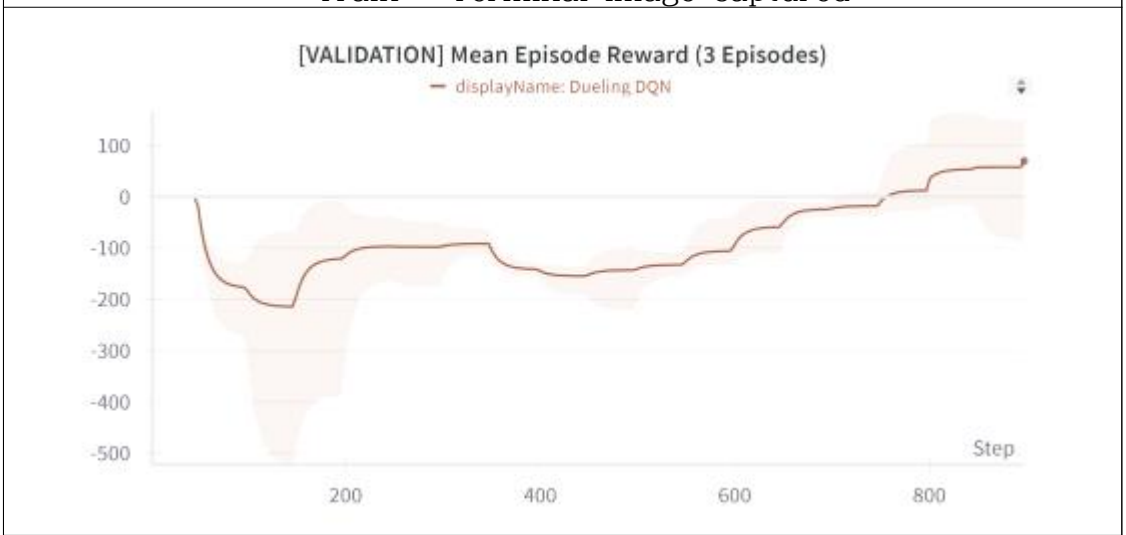
<Dueling DQN의 Q값 추정 방식>

```python
class QNet(nn.Module):
    def __init__(self, n_features, n_actions):
        super(QNet, self).__init__()
        self.n_features = n_features
        self.n_actions = n_actions

        # 기존 DQN과 달리, Dueling DQN를 위해 두 개의 스트림을 추가
        self.fc1 = nn.Linear(n_features, 128)
        self.fc2 = nn.Linear(128, 128)

        self.value_stream = nn.Linear(128, 1)
        self.advantage_stream = nn.Linear(128, n_actions)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))

        # Dueling DQN: Value와 Advantage를 계산하여 Q 값을 산출
        value = self.value_stream(x)
        advantage = self.advantage_stream(x)

        # Dueling DQN의 수식을 만족
        q_values = value + (advantage - advantage.mean(dim=1, keepdim=True))
        return q_values

    def get_action(self, state, epsilon):
        if np.random.rand() < epsilon:
            return np.random.randint(self.n_actions)
        else:
            state = torch.FloatTensor(state).unsqueeze(0)
            q_values = self.forward(state)
            return q_values.argmax().item()
```

<Dueling DQN의 Q값 추정을 위해 Q-Network 및 산식을 수정함>



Train - Terminal image captured



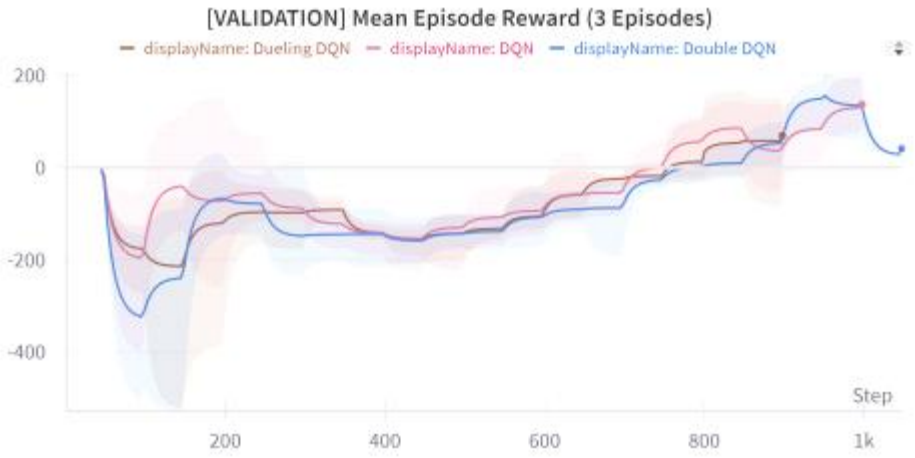
[VALIDATION] Mean Episode Reward (3 Episodes)

학습 5회 수행, 스무딩 강도 : 0.5

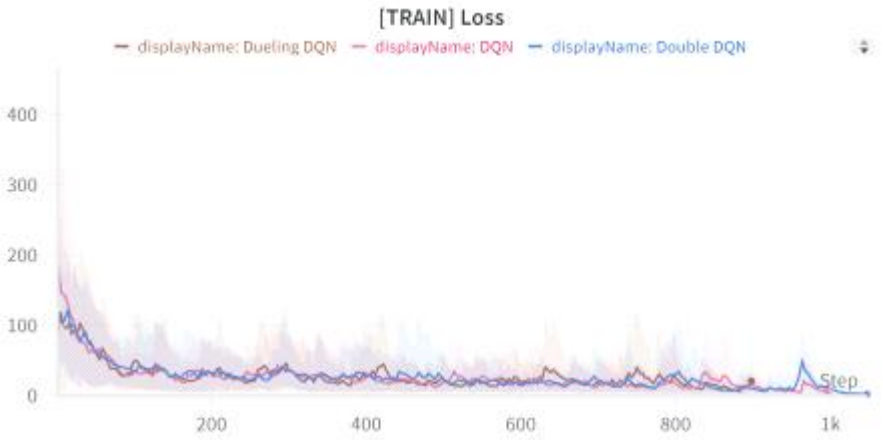
본론4_test_model)

DQN	dqn_LunarLander-v2_200.7_2024-06-05_23-21-21.pth [EPISODE: 0] EPISODE_STEPS: 251, EPISODE REWARD: 245.8 [EPISODE: 1] EPISODE_STEPS: 355, EPISODE REWARD: 265.5 [EPISODE: 2] EPISODE_STEPS: 259, EPISODE REWARD: 237.0 dqn_LunarLander-v2_208.8_2024-06-05_21-21-34.pth [EPISODE: 0] EPISODE_STEPS: 400, EPISODE REWARD: 205.9 [EPISODE: 1] EPISODE_STEPS: 459, EPISODE REWARD: 269.9 [EPISODE: 2] EPISODE_STEPS: 472, EPISODE REWARD: 217.0 dqn_LunarLander-v2_238.9_2024-06-05_22-04-45.pth [EPISODE: 0] EPISODE_STEPS: 556, EPISODE REWARD: 209.2 [EPISODE: 1] EPISODE_STEPS: 612, EPISODE REWARD: 179.2 [EPISODE: 2] EPISODE_STEPS: 365, EPISODE REWARD: 241.5 dqn_LunarLander-v2_256.2_2024-06-05_22-44-47.pth [EPISODE: 0] EPISODE_STEPS: 450, EPISODE REWARD: 228.4 [EPISODE: 1] EPISODE_STEPS: 403, EPISODE REWARD: 185.3 [EPISODE: 2] EPISODE_STEPS: 268, EPISODE REWARD: 278.8 dqn_LunarLander-v2_256.9_2024-06-05_23-57-27.pth [EPISODE: 0] EPISODE_STEPS: 314, EPISODE REWARD: 263.5 [EPISODE: 1] EPISODE_STEPS: 527, EPISODE REWARD: 231.6 [EPISODE: 2] EPISODE_STEPS: 358, EPISODE REWARD: 236.7 DQN : [Test] Mean of Episode rewards: 226.7008349781355 DQN : [Test] Standard Dev. of Episode rewards: 38.79233141255183
Double DQN	dqn_LunarLander-v2_203.6_2024-06-06_02-11-46.pth [EPISODE: 0] EPISODE_STEPS: 448, EPISODE REWARD: 258.3 [EPISODE: 1] EPISODE_STEPS: 278, EPISODE REWARD: 242.5 [EPISODE: 2] EPISODE_STEPS: 344, EPISODE REWARD: 254.0 dqn_LunarLander-v2_214.8_2024-06-06_00-29-39.pth [EPISODE: 0] EPISODE_STEPS: 511, EPISODE REWARD: 159.2 [EPISODE: 1] EPISODE_STEPS: 570, EPISODE REWARD: 193.7 [EPISODE: 2] EPISODE_STEPS: 463, EPISODE REWARD: 205.6 dqn_LunarLander-v2_219.7_2024-06-06_02-48-36.pth [EPISODE: 0] EPISODE_STEPS: 483, EPISODE REWARD: 223.3 [EPISODE: 1] EPISODE_STEPS: 661, EPISODE REWARD: 202.1 [EPISODE: 2] EPISODE_STEPS: 1000, EPISODE REWARD: 111.0 dqn_LunarLander-v2_226.4_2024-06-06_01-36-43.pth [EPISODE: 0] EPISODE_STEPS: 377, EPISODE REWARD: 249.0 [EPISODE: 1] EPISODE_STEPS: 420, EPISODE REWARD: 265.7 [EPISODE: 2] EPISODE_STEPS: 357, EPISODE REWARD: 238.6 dqn_LunarLander-v2_245.2_2024-06-06_01-00-32.pth [EPISODE: 0] EPISODE_STEPS: 371, EPISODE REWARD: 295.0 [EPISODE: 1] EPISODE_STEPS: 185, EPISODE REWARD: 264.8 [EPISODE: 2] EPISODE_STEPS: 236, EPISODE REWARD: 249.8 Double DQN : [Test] Mean of Episode rewards: 225.75732537560125 Double DQN : [Test] Standard Dev. of Episode rewards: 41.62504050340304
Dueling DQN	dqn_LunarLander-v2_203.6_2024-06-06_02-11-46.pth [EPISODE: 0] EPISODE_STEPS: 315, EPISODE REWARD: 219.4 [EPISODE: 1] EPISODE_STEPS: 347, EPISODE REWARD: 239.3 [EPISODE: 2] EPISODE_STEPS: 1000, EPISODE REWARD: 130.9 dqn_LunarLander-v2_210.3_2024-06-06_19-33-40.pth [EPISODE: 0] EPISODE_STEPS: 497, EPISODE REWARD: 234.2 [EPISODE: 1] EPISODE_STEPS: 480, EPISODE REWARD: 202.5 [EPISODE: 2] EPISODE_STEPS: 425, EPISODE REWARD: 243.2 dqn_LunarLander-v2_216.5_2024-06-06_21-14-43.pth [EPISODE: 0] EPISODE_STEPS: 504, EPISODE REWARD: 224.6 [EPISODE: 1] EPISODE_STEPS: 406, EPISODE REWARD: 261.8 [EPISODE: 2] EPISODE_STEPS: 373, EPISODE REWARD: 251.7 dqn_LunarLander-v2_228.1_2024-06-06_20-45-20.pth [EPISODE: 0] EPISODE_STEPS: 649, EPISODE REWARD: 221.9 [EPISODE: 1] EPISODE_STEPS: 389, EPISODE REWARD: 233.8 [EPISODE: 2] EPISODE_STEPS: 602, EPISODE REWARD: 203.2 dqn_LunarLander-v2_229.8_2024-06-06_20-21-47.pth [EPISODE: 0] EPISODE_STEPS: 450, EPISODE REWARD: 223.5 [EPISODE: 1] EPISODE_STEPS: 445, EPISODE REWARD: 250.5 [EPISODE: 2] EPISODE_STEPS: 421, EPISODE REWARD: 239.7 Dueling DQN : [Test] Mean of Episode rewards: 227.93042318433386 Dueling DQN : [Test] Standard Dev. of Episode rewards: 28.362524804067316

결론)



<DQN, Double DQN, Dueling DQN의 Mean episode reward>



<DQN, Double DQN, Dueling DQN의 Loss>

DQN, Double DQN, Dueling DQN의 에피소드 보상 추이를 비교 분석한 결과, DQN과 Double DQN은 학습 성능 면에서 통계적으로 유의미한 차이를 보이지 않았습니다. 이러한 결과는 Double DQN이 DQN의 overestimation bias를 줄이는 데 효과적이지만, 전반적인 보상 추이에서는 큰 개선을 가져오지 못했음을 시사합니다.

반면, Dueling DQN은 다른 두 기법에 비해 약 10% 더 빠른 학습 속도를 나타냈으며, 이는 state-value function과 action-advantage function을 분리하여 네트워크의 학습 효율을 극대화한 결과로 볼 수 있습니다. 특히, 모델 테스트 시 Dueling DQN은 보상의 분산이 더 낮게 나타나, 모델의 안정성이 상대적으로 높다는 것을 확인할 수 있었습니다.

한편, 본 과제에서 사용한 환경(Lunar Lander)이 state와 action이 비교적 단순한 환경이었기 때문에, Double DQN과 Dueling DQN의 구조적 장점이 인상적인 효과를 발휘하지 못한 것으로 판단됩니다. 복잡한 행동 공간에서는 이러한 기법들이 더 큰 성능 향상을 가져올 수 있을 것으로 예상됩니다.

따라서, Dueling DQN이 Q-learning 기반 알고리즘 중 더 효율적이고 안정적인 학습을 제공할 수 있는 가능성이 높을 것 같습니다. 특히, 고차원 행동 공간에서는 Dueling DQN의 장점이 더욱 두드러질 것으로 기대됩니다.