JKSCI

한국컴퓨터정보학회논문지
Journal of The Korea Society of Computer and Information
Vol. 30 No. 9, pp. 1-10, September 2025
https://doi.org/10.9708/jksci.2025.30.09.001

Enhanced Deep Q-Learning with Multiple Replay Memories: A Heuristic-Based Approach

Junha Hwang*

*Professor, Dept. of Computer Engineering, Kumoh National Institute of Technology, Gumi, Korea

[Abstract]

Deep Q-Learning (DQL) is a representative deep reinforcement learning method. Deep Q-Network (DQN), based on DQL, has improved the performance by replaying past experiences stored in a replay memory. In the DQN, past experiences are randomly sampled from a single replay memory for training. This paper proposes a heuristic-based approach using multiple replay memories to further enhance the performance of DQL. Specifically, experiences are categorized based on knowledge of the target problem and stored in their corresponding replay memories. During training, a fixed number of experiences are randomly sampled from each replay memory. The proposed method was applied to the Cart Pole, the Mountain Car, and the Ball Avoider problems, and comparative experiments were conducted to compare with existing methods. The experimental results show that the proposed method can significantly improve the performance of DQL.

▸ **Key words**: Reinforcement Learning, Deep Q-Learning, Deep Q-Network, Replay Memory, Multiple Replay Memories

[요 약]

심층 Q-학습은 대표적인 심층 강화 학습 방법 중 하나이다. 심층 Q-학습을 기반으로 한 심층 Q-네트워크는 리플레이 메모리에 저장된 기존 경험들을 재생함으로써 심층 Q-학습의 성능을 향상시켰다. 심층 Q-네트워크에서는 하나의 리플레이 메모리로부터 기존 경험들을 무작위로 추출하여 학습을 수행한다. 본 논문에서는 심층 Q-학습의 성능을 더욱 향상시키기 위해 다중 리플레이 메모리를 활용하는 휴리스틱 기반 접근법을 제안한다. 구체적으로는 대상 문제에 대한 지식을 기반으로 경험을 분류하고 각각 해당 리플레이 메모리에 저장하며, 학습 시에는 각 리플레이 메모리로부터 일정 개수의 경험들을 무작위로 추출한다. Cart Pole, Mountain Car, Ball Avoider 문제를 대상으로 제안 기법과 기존 기법들에 대한 비교 실험을 수행한 결과, 제안한 기법을 통해 심층 Q-학습의 성능을 크게 향상시킬 수 있음을 확인하였다.

▸ **주제어**: 강화 학습, 심층 Q-학습, 심층 Q-네트워크, 리플레이 메모리, 다중 리플레이 메모리

• First Author: Junha Hwang, Corresponding Author: Junha Hwang
*Junha Hwang (jhhwang@kumoh.ac.kr), Dept. of Computer Engineering, Kumoh National Institute of Technology
• Received: 2025. 07. 21, Revised: 2025. 08. 14, Accepted: 2025. 08. 28.

http://www.ksci.re.kr pISSN:1598-849X | eISSN:2383-9945

I. Introduction

기계 학습의 한 분야인 강화 학습은 마르코프 결정 문제로 모델링되는 순차적 의사 결정 문제를 해결하는 방법이다. Fig. 1은 강화 학습의 구성 요소 및 학습 과정을 보여준다[1]. 에이전트는 현재 상태를 기준으로 정책에 따라 행동을 결정한다. 행동을 수행하면 환경은 그에 따른 보상을 반환하고 현재 상태는 다음 상태로 변하게 된다. 강화 학습의 목표는 장기적인 보상의 합을 최대화하는 최적 정책을 찾는 것이다. 강화 학습 방법은 정책 자체를 토대로 의사 결정을 하는 정책 기반 강화 학습과 상태의 가치를 토대로 의사 결정을 하는 가치 기반 강화 학습으로 나뉜다.

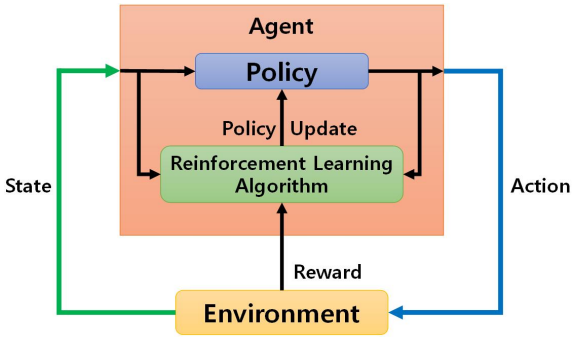

Fig. 1. Block Diagram of Reinforcement Learning

심층 강화 학습은 강화 학습과 딥 러닝을 결합한 방법으로 정책 또는 상태의 가치를 평가하기 위해 인공 신경망을 사용한다. 심층 Q-학습은 대표적인 가치 기반 심층 강화 학습 방법으로 어떤 상태에서 특정 행동을 취할 시 궁극적으로 얻게 되는 보상의 합을 예측하는 행동-가치 함수를 인공 신경망으로 표현하여 학습시킨다. 따라서 학습이 잘 이루어진 행동-가치 함수가 있다면 현재 상태와 행동을 입력하여 출력값이 가장 큰 행동을 실행하면 된다.

심층 Q-학습을 기반으로 하는 심층 Q-네트워크(DQN)는 Atari 게임들을 학습하기 위한 합성공 신경망으로 심층 Q-학습의 성능을 크게 향상시켰다[2, 3]. DQN의 가장 큰 특징은 리플레이 메모리의 활용이다[4]. 기존 심층 Q-학습의 경우 현재 경험 하나에 대한 학습을 수행하는 반면에, DQN에서는 과거의 경험들을 리플레이 메모리에 저장하고 학습 시 리플레이 메모리로부터 여러 개의 경험들을 무작위로 선택하여 학습을 수행하는 경험 재생 기법을 사용한다. 경험 재생은 학습 효율을 높이고 보다 안정적인 학습을 가능하게 함으로써 많은 심층 강화 학습 알고리즘의 표준 요소로 자리잡고 있다[5].

본 연구에서는 각 경험의 종류에 주목한다. 대상 문제의 지식을 활용하여 경험들을 몇 개의 종류로 분류하고, 해당 종류별로 리플레이 메모리를 준비한다. 각 경험은 해당 종류의 리플레이 메모리에 저장되며 학습을 위한 경험 선택 시 각 리플레이 메모리로부터 일정 개수의 경험들을 무작위로 선택한다. 본 논문에서는 해당 기법을 다중 리플레이 메모리 기법이라 부른다.

제안 기법을 Cart Pole, Mountain Car, Ball Avoider 문제에 적용하고 단일 리플레이 메모리와의 성능을 비교한다[6, 7]. 당초 본 연구는 Ball Avoider 문제의 해결을 위한 다중 리플레이 메모리의 활용으로부터 출발하였다. 즉, 기존 DQN의 단순한 파라미터 조정만으로는 해결이 어려워 다중 리플레이 메모리를 적용하게 되었다. 본 연구에서는 이를 확장하여 Cart Pole 문제와 Mountain Car 문제에도 적용함으로써 다중 리플레이 메모리의 적용 방안을 제시하고 그 효과를 검증한다. 이상을 요약한 본 논문의 기여 사항은 다음과 같다.

- DQN을 기반으로 한 다중 리플레이 메모리 구조 제안
- 대상 문제별 다중 리플레이 메모리의 적용 방안 제시
- 실험을 통한 다중 리플레이 메모리의 유용성 검증

본 논문의 구성은 다음과 같다. 2장에서는 관련 연구에 관해 기술하고 3장에서는 본 연구에서 제안한 기법에 대해 설명한다. 4장에서는 실험 결과를 제시하고 분석하며 5장에서는 결론 및 향후 과제를 설명한다.

II. Related Works

1. Deep Q-Network

DQN의 성공 요인은 크게 두 가지이다. 첫 번째는 리플레이 메모리를 이용한 경험 리플레이이며[2], 두 번째는 학습 네트워크와는 별도로 정답을 제공해 주는 타겟 네트워크를 활용하는 것이다[3]. 본 연구는 타겟 네트워크와는 무관하므로 리플레이 메모리의 활용에 관해서만 설명한다.

리플레이 메모리를 활용한 DQN 알고리즘은 Fig. 2와 같다[2]. 먼저 리플레이 메모리의 크기를 설정하고 인공 신경망으로 표현되는 $Q(s,a)$ 함수를 초기화한다. $Q(s,a)$는 상태 s에서 행동 a를 수행했을 때 최종적으로 얻게 되는 누적 보상의 최댓값을 의미한다. 이 값을 잘 예측할 수 있다면 항상 가장 큰 값을 얻게 되는 행동 a를 취하면 된다. Q 함수를 표현하는 인공 신경망은 여러 번의 에피소드를 수행함에 따라 지속적으로 업데이트된다. 에피소드란

한 번의 초기 상태부터 종료 상태까지의 일련의 상태 변화를 의미한다. 대상 문제에 따라 에피소드별로 종료 상태까지의 스텝 수가 같을 수도 있고 다를 수도 있다. 하나의 에피소드 내에서는 Fig. 2의 4라인과 같이 현재 상태를 초기 상태로 초기화한 후 하나의 에피소드가 종료될 때까지 다음 내용을 반복 수행한다. 먼저 Q 함수를 기반으로 $\epsilon-greedy$ 정책을 사용하여 수행할 행동을 선택한다. 즉, ϵ의 확률로 무작위 행동을 선택하고 $(1-\epsilon)$의 확률로 가장 큰 값의 행동을 선택한다. 선택된 행동을 수행하면 환경으로부터 보상을 받게 되고 다음 상태로 이동한다. 한 번의 스텝을 통해 얻는 경험 $(s_t,\ a_t,\ r_t.\ s_{t+1})$은 리플레이 메모리에 저장된다. $(s_t,\ a_t,\ r_t.\ s_{t+1})$은 각각 (현재 상태, 행동, 보상, 다음 상태)를 의미한다. 그리고 5~9라인과 같이 한 번의 에피소드를 통해 수집한 경험들을 리플레이 메모리에 저장한 후 에피소드가 종료되면 10라인과 같이 리플레이 메모리로부터 C개의 경험들을 선택하고 인공 신경망을 업데이트하기 위한 학습을 수행한다. 이때 손실 함수는 15라인과 같이 계산된다. 할인 계수 γ는 0과 1 사이의 값으로 미래 보상에 대한 감가율을 의미한다. DQN은 Fig. 2와 같이 매 에피소드마다 한 번의 학습을 수행할 수도 있지만 매 스텝마다 학습을 수행할 수도 있다. 본 연구에서는 기법 간의 공정한 비교를 위해 매 에피소드가 끝날 때마다 한 번의 학습을 수행한다.

```
1: Initialize replay memory D to capacity N
2: Initialize action-value function Q(s, a)
3: for episode = 1, M
4:     Initialize current state s_1 to starting state
5:     for t = 1, T
6:         With probability ϵ select a random action a_t
7:             else action a_t = argmax_a Q(s_t, a)
8:         Execute a_t and observe reward r_t, next state s_{t+1}
9:         Store transition (s_t,a_t,r_t.s_{t+1}) in D
10:    Sample C random transitions (s_j,a_j,r_j.s_{j+1}) from D
11:    if s_{j+1} is terminal state
12:        y_j = r_j
13:    else
14:        y_j = r_j + γ max_{a'} Q(s_{j+1}, a')
15:    Perform a gradient descent on (y_j - Q(s_j, a_j))^2
```

Fig. 2. DQN Algorithm

2. Prioritized Experience Replay (PER)

우선순위 경험 재생(PER) 기법은 본 연구와 같이 각 경험에 주목하는 기법으로 2015년 처음 제안된 이후 현재까지도 강화 학습의 성능 향상을 위해 많이 적용되고 있다. 따라서 본 연구에서는 다중 리플레이 메모리 기법의 성능을 검증하기 위해 기존 DQN 뿐만 아니라 PER과의 비교 결과를 검토한다.

기존 DQN에서 리플레이 메모리로부터 각 경험들이 선택될 확률은 모두 동일하다. 그런데 PER에서는 경험마다 학습에 대한 기여도가 다르다고 본다[8]. 여기서 학습에 대한 기여도로는 식 1과 같은 TD-오차(Time Difference-error) δ의 절댓값이 주로 사용된다. 현재 목푯값은 $r_t+\gamma\max_a Q(s_{t+1},a)$고 학습 대상값은 $Q(s_t,a_t)$이므로 둘 사이의 차이가 클수록 학습에 더 많이 기여한다고 추정한다. 따라서 PER에서는 식 1의 절댓값에 비례하여 경험들을 선택한다.

$$\delta=r_t+\gamma\max_a Q(s_{t+1},a)-Q(s_t,a_t) \qquad (1)$$

PER의 구체적인 알고리즘은 Fig. 3과 같다. 참고로 DQN과의 비교를 위해 Fig. 2와 동일한 형태로 기술하였다. 예를 들어, 기존 PER에서는 일정 스텝마다 한 번씩 학습을 수행하는 데 반해, Fig. 3에서는 Fig. 2와 같이 매 에피소드가 끝날 때마다 학습을 수행한다. Fig. 2와 비교할 때 3, 12, 14라인이 추가되었고, Fig. 2의 9, 10, 15라인은 각각 Fig. 3의 10, 11, 19라인과 같이 수정되었다.

```
1: Initialize replay memory D to capacity N
2: Initialize action-value function Q(s, a)
3: Initialize parameter values α, β
4: for episode = 1, M
5:     Initialize current state s_1 to starting state
6:     for t = 1, T
7:         With probability ϵ select a random action a_t
8:             else action a_t = argmax_a Q(s_t, a)
9:         Execute a_t and observe reward r_t, next state s_{t+1}
10:        Store transition (s_t,a_t,r_t.s_{t+1}) in D with maximal
               priority p_t=1 if the first step else max_i p_i
11:    Sample C transitions (s_j,a_j,r_j.s_{j+1}) from D with
           probability P(j) = p_j^α / Σ_i p_i^α
12:    Compute importance-sampling weight
           w_j = (N • P(j))^{-β} / max_i w_i
13:    Compute TD-error(δ_j) by expression (1)
14:    Update transition priority p_j = |δ_j|
15:    if s_{j+1} is terminal state
16:        y_j = r_j
17:    else
18:        y_j = r_j + γ max_{a'} Q(s_{j+1}, a')
19:    Perform a gradient descent on w_j • (y_j - Q(s_j, a_j))^2
```

Fig. 3. PER Algorithm

10라인에서는 새로운 경험을 저장할 때 해당 경험의 우선순위 값을 함께 저장하는데, 가장 최근의 경험이 가장 중요한 경험으로 인식될 수 있도록 지금까지 리플레이 메모리에 저장된 모든 경험들의 우선순위 값 중 가장 큰 값

을 부여한다. 단, 처음으로 저장되는 경험의 경우 1의 값을 갖는다. 13, 14라인에서는 학습을 위해 선택되는 경험들에 한해 식 1의 TD-오차의 절댓값으로 우선순위 값을 변경한다. 실제 각 경험의 샘플링 확률은 11라인의 식과 같다. 즉 모든 경험들의 우선순위 값들의 합을 기준으로 한 비율이 확률이 된다. 실전에 있어서 p_j 값은 0.01과 같은 작은 값을 일괄적으로 더하여 원래 값이 0인 경우에도 선택될 기회를 부여한다. 여기서 파라미터 α는 0과 1 사이의 고정값으로 작은 값일수록 DQN과 같은 무작위 샘플링에 가까워지고 큰 값일수록 우선순위를 더 많이 반영한 샘플링이 된다. 그런데 우선순위가 높은 경험들 위주로 선택하다 보면 다양성이 결여되어 결국 학습 효율이 떨어질 수 있다. 이를 보완하기 위해 12라인과 같은 중요도 샘플링 가중치를 계산하고 19라인과 같이 이 값을 반영하여 인공신경망의 가중치를 수정하게 된다. 이를 통해 우선순위가 높은 경험에 의해서는 인공 신경망 가중치를 작게 조정하고, 우선순위가 낮은 경험에 의해서는 더 크게 조정하게 된다. 12라인에서 사용된 파라미터 β의 값은 0과 1 사이의 값으로 중요도 샘플링 가중치가 학습에 미치는 영향을 제어하는 데 사용된다. 이 값이 작을수록 기존 우선순위 값이 그대로 학습에 더 많이 반영되고 이 값이 커질수록 우선순위에 의한 반영은 약해지게 된다. 보통 학습 시작 시 0.4 등과 같은 값으로 설정하여 우선순위 값이 큰 경험을 더 많이 반영하고, 학습이 진행됨에 따라 최종 1.0까지 조금씩 증가시킴으로써 점점 더 다양한 경험이 학습에 반영될 수 있도록 유도한다.

기존 연구 [9]에서는 PER의 한계를 지적하고 있는데, 그 근거 중 하나는 학습이 진행됨에 따라 인공 신경망이 업데이트되면 기존 경험들의 우선순위 값 또한 기존과 달라진다는 것이다. 그렇다고 매 스텝마다 모든 경험의 우선순위 값을 재계산하는 것은 현실적으로 불가능하다고 지적하고 있다. 이에 따라 PER를 개선한 다양한 방법들에 관한 연구가 진행되어 왔다[10, 11].

3. Target Problems

본 연구에서 제안하는 기법은 대상 문제에 대한 이해로부터 출발한다. 본 절에서는 본 연구에서 활용한 3가지 대상 문제에 관해 설명한다.

3.1 Cart Pole

첫 번째 대상 문제는 OpenAI Gym에서 제공하는 CartPole-v1 문제이다[6]. Cart Pole 문제는 Fig. 4와 같이 트랙 위의 카트와 폴로 구성되며, 카트와 폴은 서로 연결되어 있다[12]. 에이전트는 카트의 왼쪽이나 오른쪽으로 일정한 힘을 가할 수 있으며, 500 타임 스텝동안 넘어지지 않도록 균형을 유지하는 것이 목표이다. 구체적인 Cart Pole 문제의 정의는 Table 1과 같다.

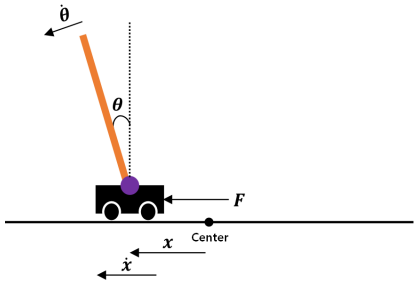

Fig. 4. Cart Pole System

Table 1. Definition of the Cart Pole Problem

Item	Explanation
Action	· 0 : Push cart to the left · 1 : Push cart to the right
State	· Cart position(x) : -4.8 ~ 4.8 · Cart velocity($\dot{x}$) : -Inf ~ Inf · Pole angle(θ,rad) : -0.418 ~ 0.418 · Pole angular velocity($\dot{\theta}$) : -Inf ~ Inf
Reward	· +1 for each step
Starting State	· Each state value : random in range (-0.05, 0.05)
Terminal State	· Pole angle > ±12° (fail) · Cart position > ±2.4 (fail) · Episode length = 500 steps (success)

3.2 Mountain Car

두 번째 대상 문제 또한 OpenAI Gym에서 제공하는 MountainCar-v0 문제이다[6, 13]. Fig. 5와 같이 계곡의 특정 지점에 위치한 자동차를 매 스텝마다 전략적으로 좌우 한 방향으로 가속시켜 오른쪽 언덕 꼭대기에 도달시켜야 한다[13]. 그것도 200 타임 스텝 내에 최대한 빨리 도달하는 것이 목표이며, 200 타임 스텝이 될 때까지 성공하지 못하면 해당 에피소드는 실패로 간주된다. 구체적인 Mountain Car 문제의 정의는 Table 2와 같다.

에이전트가 취한 행동 action에 따른 속도(v)와 위치(p)의 변환식은 각각 식 2, 식 3과 같다. 여기서 force는 0.001이고 gravity는 0.0025이다. 자동차의 위치는 [-1.2, 0.6]으로 제한되고, 속도는 [-0.07, 0.07]로 제한된다. 이와 같이 행동, 즉 가속뿐만 아니라 힘(force)과 중력(gravity)이 속도에 영향을 미치기 때문에 단순히 오른쪽 방향으로 가속하는 것만으로는 동력이 부족하여 목표 위치에 도달하지 못한다. 따라서 오른쪽 또는 왼쪽 언덕에서

내려올 때 반대 방향으로의 가속을 통해 추진력을 얻어야 목표 지점까지 도달할 수 있는 힘이 생기게 된다.

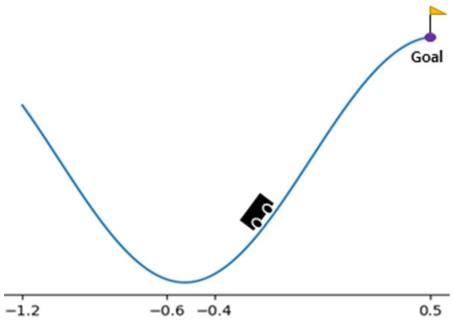

Fig. 5. Mountain Car System

Table 2. Definition of the Mountain Car Problem

Item	Explanation
Action	· 0 : Accelerate to the left · 1 : Don't accelerate · 2 : Accelerate to the right
State	· Car position(x) : -4.8 ~ 4.8 · Car velocity : -0.07 ~ 0.07
Reward	· -1 for every step
Starting State	· Car position : random in range (-0.6, -0.4) · Car velocity : 0
Terminal State	· Car position ≥ 0.5 (success) · Episode length = 200 steps (fail)

$$v_{t+1} = v_t + (action - 1) \times force - cos(3 \times p_t) \times gravity \quad (2)$$

$$p_{t+1} = p_t + v_{t+1} \quad (3)$$

3.3 Ball Avoider

Ball Avoider 문제는 Fig. 6과 같이 가로×세로 크기가 800×400인 직사각형에 포함된 초록색 공 1개와 빨간색 공 5개로 구성된다[7]. 빨간색 공은 각각 x, y 방향으로의 일정 속도를 가지고 직선으로 이동하고, 초록색 공은 매 순간 동, 서, 남, 북 방향 중 한 방향으로 이동한다. 에이전트는 초록색 공의 방향을 조정하여 500 타임 스텝동안 빨간색 공 또는 벽에 부딪히지 않도록 하는 것이 목표이다. 구체적인 Ball Avoider 문제의 정의는 Table 3과 같다.

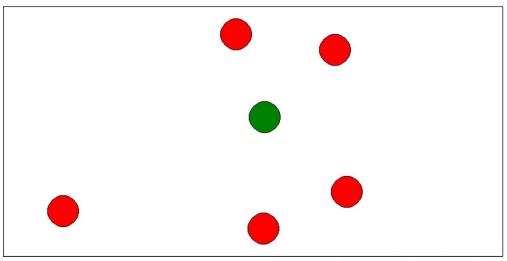

Fig. 6. Ball Avoider System

Table 3. Definition of the Ball Avoider Problem

Item	Explanation
Action	· 0 : Turn to the East · 1 : Turn to the West · 2 : Turn to the South · 3 : Turn to the North
State	· Ball position(x, y) : (0 ~ 799, 0 ~ 399) · Green ball velocity(x, y) : (10, 0) or (-10, 0) or (0, 10) or (0, -10) · Red ball velocity(x, y) : (-10 ~ 10, -10 ~ 10)
Reward	· +1 for each step
Starting State	· Green ball position : (400, 200) · Green ball velocity : (10, 0) · Red ball position : random in range [50, 350] or range [450, 750] · Red ball velocity : random in range [-10, -1] or range [1, 10]
Terminal State	· Collision with the red ball (fail) · Collision with the wall (fail) · Episode length = 500 steps (success)

III. Multiple Replay Memories Approach

본 연구에서 제안하는 다중 리플레이 메모리를 활용한 DQN 알고리즘은 기존 DQN 알고리즘인 Fig. 2에서 다음과 같이 3가지가 달라진다.

① k 개의 리플레이 메모리 [D_1, ..., D_k]를 준비하고 각각 일정 크기 [N_1, ..., N_k]로 설정한다. 리플레이 메모리의 개수 k는 경험 종류의 개수와 동일하다.
② 경험 (s_t, a_t, r_t. s_{t+1})를 저장할 때 경험의 종류에 따라 해당 종류의 리플레이 메모리 D_i로 저장한다.
③ 리플레이 메모리로부터 경험들을 샘플링할 때 각 리플레이 메모리로부터 [C_1, ..., C_k] 개수만큼 선택한다.

사실상 경험의 종류에 따른 리플레이 메모리를 준비하고 활용하는 것 외에는 기존 DQN과 동일하다고 할 수 있다.

본 연구에서는 단일 리플레이 메모리를 사용하는 기존 DQN을 SRM(Single Replay Memory)이라 부르고 다중 리플레이 메모리를 사용하는 방법을 MRM(Multiple Replay Memory)이라 부른다. 다음은 SRM, PER, MRM을 리플레이 메모리 개수와 샘플링 방법 등에 따라 비교 정리한 내용이다.

- 리플레이 메모리 개수 : SRM과 PER은 1개, MRM은 대상 문제 및 활용 방식에 따라 2개 이상을 사용한다.
- 각 경험의 분류 방법 : PER은 TD-에러에 따라 경험을 구분하고 MRM은 대상 문제의 지식을 활용하여 각

경험의 종류를 구분한다. SRM은 특별한 구분이 없다.

- 샘플링 방법 : SRM은 무작위 추출, PER은 우선순위 값에 기반한 확률($P(j)$)에 비례한 확률적 추출, MRM은 리플레이 메모리별로 무작위 추출을 사용한다.
- 실행 속도 : SRM이 가장 빠르고, 우선순위 값 계산 등 부가적인 작업이 필요한 PER의 속도가 가장 느리다.
- 장단점 : PER은 TD-에러라는 훌륭한 학습 지표가 있으므로 자동화에 유리하지만, 모든 대상 문제에서 TD-에러 지표가 효과적이라는 보장이 없다. MRM은 대상 문제마다 분류 기준을 마련해야 하므로 적용에 어려움이 있지만, 대상 문제에 적합한 분류 기준을 찾을 수 있다면 매우 효과적인 학습이 가능하다.

SRM에서 리플레이 메모리를 활용하여 경험 리플레이 기법을 사용하는 이유는 서로 독립적인 경험들을 샘플링하여 학습함으로써 학습의 일반화 능력 및 안정성을 향상시키기 위함이다. 그런데, 경험 리플레이를 위해 선택되는 경험들이 특정 종류에 국한된다면 학습이 제대로 진행되기 어려울 수 있다. 예를 들어, Ball Avoider 문제에서 충돌에 의해 종료 상태가 되는 경험이 선택되지 않는다면 충돌을 피하기 위한 학습이 불가능하게 된다. 본 연구에서는 다중 리플레이 메모리의 활용을 통해 샘플링 편향 현상을 줄임으로써 학습의 성능이 향상되기를 기대한다.

Table 4는 대상 문제별로 SRM과 MRM의 적용 방식을 나타낸 것이다. Table 4에서 k가 1인 경우 SRM을 의미하고 k가 2 이상인 경우 MRM을 의미한다. PER은 단일 리플레이 메모리를 사용하므로 SRM 방식과 동일하다. MRM은 대상 문제에 따라 다양한 방식으로 적용이 가능한데, Table 4는 최종 실험에 활용한 적용 방식을 의미한다. Ball Avoider 문제의 경우 기존 연구 [7]에서 적용한 3가지 종류로 나누는 방식 외에 12가지 종류로 나누는 방식을 추가로 적용하였다.

Cart Pole 문제에서 MRM은 3개의 리플레이 메모리로 구성된다. 하나는 폴이 중앙의 왼쪽으로 기울어져 있다가 중앙 방향으로 이동하는 경험을 저장하고, 또 하나는 오른쪽에서 중앙 방향으로 이동하는 경험을 저장한다. 나머지 경험들은 마지막 리플레이 메모리에 저장된다. SRM과 MRM의 공정한 비교를 위해 MRM의 메모리 크기의 합을 SRM의 메모리 크기와 동일하게 설정하였으며, 학습을 위해 선택되는 총 경험 개수 또한 동일하게 설정하였다.

Table 4. Application Methods of SRM and MRM for Each Problem

Problem	k	types of D_i	N_i	C_i
Cart Pole	1	-	3000	128
	3	Move from left to center	1000	32
		Move from right to center	1000	32
		The others	1000	64
Mountain Car	1	-	3000	128
	3	Move right and Accelerate right	1000	43
		Move left and Accelerate left	1000	43
		The others	1000	42
Ball Avoider	1	-	30000	640
	3	Collision with the wall	2000	256
		Collision with the red balls	10000	256
		The others	18000	128
	12	Collision with the east wall	500	64
		Collision with the west wall	500	64
		Collision with the south wall	500	64
		Collision with the north wall	500	64
		Collision with the east red ball	2500	64
		Collision with the west red ball	2500	64
		Collision with the south red ball	2500	64
		Collision with the north red ball	2500	64
		The others : move east	4500	32
		The others : move west	4500	32
		The others : move south	4500	32
		The others : move north	4500	32

Mountain Car 문제는 자동차가 오른쪽으로 이동 시 오른쪽으로 가속하는 경우와 왼쪽으로 이동 시 왼쪽으로 가속하는 경우를 별도의 리플레이 메모리로 저장하였다. 이는 거의 정답에 가까운 매우 강력한 휴리스틱이라 할 수 있다. 그럼에도 불구하고 매 스텝마다 -1을 얻게 되는 기존 보상을 그대로 사용하면 경험들 사이의 우열 정보가 부족하여 학습이 제대로 진행되기 어렵다. 따라서 기존 연구 [3]과 유사하게 식 4와 같은 보상을 사용한다. 즉, 자동차의 최저점 위치인 -0.5를 기준으로 멀리 떨어진 경우 더 큰 보상을 받게 된다. 단, 목표 지점인 0.5에 도달하면 200의 보상을 받는다. 식 4에서 가중치 역할을 하는 10의 값은 여러 번의 실험을 통해 학습이 더 잘 진행될 수 있는 값으로 결정한 것이다.

$$r_t = (p_{t+1} + 0.5)^2 \times 10 \tag{4}$$

Ball Avoider 문제의 첫 번째 MRM은 벽에 충돌하여 실패하는 경우와 빨간색 공과 충돌하여 실패하는 경우를 각각 별도의 리플레이 메모리로 구성하였으며, 두 번째 MRM은 첫 번째 MRM보다 경험들의 종류를 보다 세분화하였다. Ball Avoider 문제의 경우 다른 문제들에 비해 학습이 매우 어려웠다. 따라서 리플레이 메모리의 크기와 샘플링 개수를 훨씬 크게 설정하였다.

IV. Experimental Results

본 연구의 실험은 AMD Ryzen 9 3900X 12-Core Processor, 3.8GHz, 48GB RAM, 윈도우 10 64비트의 PC 환경에서 수행되었으며, 프로그램 구현을 위해 Python 3.7, Tensorflow 2.2.3, gym 0.26.2 라이브러리를 사용하였다. 실행 환경 및 프로그래밍 환경은 중요하지 않다. 어떤 환경에서든 SRM, PER, MRM 사이의 공정한 비교가 가능하면 된다. 다만 위 프로그래밍 환경이 Python 3.10, Tensorflow 2.10 등 몇몇 프로그래밍 환경에서 수행한 결과보다 실행 속도가 더 빠르고 성능 또한 약간 우수함을 확인하였다. 물론 SRM, PER, MRM 사이의 상대적인 결과는 어떤 환경에서든 동일한 결과를 보였다.

모든 대상 문제 및 실험에 있어서 학습을 위한 인공 신경망으로는 완전 연결 순방향 신경망을 사용하였다. 은닉층 활성화 함수로는 relu 함수, 출력층 활성화 함수로는 linear 함수를 사용하였다. 손실 함수로는 mse를 사용하였으며 옵티마이저는 Adam 옵티마이저를 사용하였다. 그리고 1000개 이상의 경험이 수집된 이후로 매 에피소드가 끝난 후 학습을 수행하였다. 이외의 파라미터 값들은 Table 5와 같다. 일부 파라미터들은 문제의 특성에 따라 다르게 설정하였는데, 그러한 경우에도 Cart Pole과 Mountain Car는 동일하게 설정하였다. 중요한 것은 하나의 대상 문제에 적용되는 모든 기법들은 동일한 파라미터 값을 사용한다는 것이다.

ϵ 값은 최초값부터 최종값까지 조금씩 감소하되 감소 방법으로는 모든 대상 문제에 대해 linear epsilon decay 방법을 사용하였다[3]. 첫 번째 에피소드에서의 ϵ 값은 최초값으로 설정되고 총 에피소드의 2분의 1에 해당하는 에피소드에서의 ϵ 값이 최종값이 되도록 한다. 그 사이의 ϵ 값은 매 에피소드마다 선형적으로 감소하게 되며, 2분의 1 이후의 ϵ 값은 최종값을 유지한다.

Table 5. Parameter Settings

Parameter	Cart Pole/ Mountain Car	Ball Avoider
# of hidden layers	4	9
# of neurons in each hidden layer	256, 128, 64, 32	4096, 2048, 1024, 512, 256, 128, 64, 32, 16
learning rate	0.001	0.0001
Initial ϵ	0.9	0.3
final ϵ	0.01	0.01
discount factor	0.9	0.9
# of episodes	1000	30000

PER에서 α 값의 경우 Cart Pole 문제는 0.4, Mountain Car와 Ball Avoider 문제는 0.8을 사용하였다. β 값의 초기값은 모든 대상 문제에 있어서 0.4를 사용하였고 마지막 에피소드 수행 시 1.0이 되도록 선형적으로 증가시켰다.

각 대상 문제에 적용된 기법은 기존 DQN(SRM), 우선순위 경험 재생(PER), 다중 리플레이 메모리(MRM) 이렇게 총 3가지이며, 대상 문제마다 각 기법별로 총 30회의 실험을 수행하였다.

Table 6은 Cart Pole 문제에 대한 실험 결과로 1000회의 에피소드 수행 중 성공한 에피소드의 횟수를 요약한 것이다. MRM은 30회 실험 중 평균 성공 횟수가 54.93회이다. 평균 성공 횟수가 30.53회인 SRM에 비해서는 성능이 크게 향상되었다고 볼 수 있다. 그러나 PER의 경우 평균 성공 횟수가 157.97회로 MRM보다 훨씬 우수한 성능을 발휘하였다.

Table 6. Success Count in Cart Pole

Metric	SRM	PER	MRM
Average	30.53	157.97	54.93
Maximum	94	284	153
Minimum	1	27	3
SD	23.31	54.96	36.88
CV	0.76	0.35	0.67
p-value	-	1.28×10^{-14}	0.001 1.1×10^{-11}

Table 6에는 평균, 최대, 최소 외에 표준 편차(Standard Deviation, SD), 변동 계수(Coefficient of Variation, CV), 독립표본 t-검정에서의 단측 검정 결과인 p-value를 함께 표시하였다. 기법별로 실험 결과의 범위가 달라지기 때문에 표준 편차는 특별한 정보를 제공하지 못한다. 대신 평균 대비 표준 편차를 의미하는 변동 계수를 보면 성능이 가장 좋은 PER이 안정성 또한 가장 높음을 알 수 있다. PER의 p-value 값은 SRM과의 검정 결과를, MRM의 p-value 값들은 각각 SRM과 PER과의 검정 결과를 나타낸 것이다. p-value는 모두 매우 작은 값을 나타내고 있으며, 이를 통해 기법들의 성능 사이에 유의미한 차이가 있음을 확인할 수 있다.

Fig. 7은 Cart Pole 문제에 대한 에피소드별 성공 횟수의 누적치와 스텝 횟수를 그래프로 나타낸 것이다. 참고로 스텝 횟수는 보상의 합과 동일한 의미이며, Fig. 7의 (a), (b) 그래프 모두 각 에피소드별 값은 30회 실험의 평균값을 의미한다. 다만 (b) 스텝 횟수는 증감 추세를 보다 쉽게 파악할 수 있도록 20 이동 평균을 사용하였다. Fig. 7에서

도 Table 6과 같이 PER이 최종적으로 가장 좋은 성능을 발휘함을 확인할 수 있다. 그런데 Fig. 7 (b) 그래프를 보면 PER의 경우 학습 초반에는 다른 방법에 비해 성능이 좋지 않음을 확인할 수 있으며, fig. 7 (a) 그래프에서도 PER의 성공 횟수가 다른 방법들에 비해 조금 늦게 증가함을 알 수 있다. 기존 연구 [14]에서도 PER이 학습 초반에 성능이 낮음을 지적하였는데, 이는 학습 초반 우선순위 기반 샘플링 방식으로 인해 다양성이 저하된 것이 원인이며 후반으로 갈수록 중요도 샘플링을 사용하여 데이터의 활용도를 높임으로써 학습 효율을 개선하는 것으로 분석하였다. Cart Pole 문제의 실험 결과를 통해 MRM이 SRM보다 성능이 좋음을 확인할 수 있지만 PER과 비교할 때 MRM의 유용성을 확인하기는 어렵다.

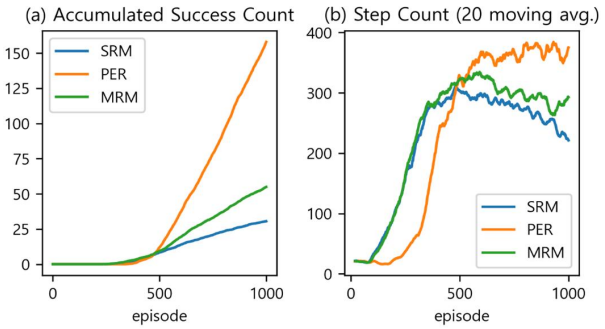

Fig. 7. Experimental Results on Cart Pole

Table 7은 Mountain Car 문제에 대한 실험 결과로 성공 횟수를 요약한 것이다. 기본적으로 Mountain Car는 SRM와 MRM을 통한 학습 성능이 우수한 편이다. 그 중에서도 MRM은 SRM보다 훨씬 성능이 우수하며 안정적인 학습 능력을 발휘한 것으로 판단된다. 반면에 PER의 경우 성능이 매우 저조한 것으로 나타났다. 기존 연구 [15]에서도 Mountain Car 문제에 대한 PER의 성능이 좋지 않은 것으로 판단하였다. 아울러 변동 계수를 통해 성능이 가장 우수한 MRM의 학습 안정성을 확인할 수 있으며, p-value 또한 매우 작은 값을 보여 기법들 사이의 차이가 뚜렷함을 알 수 있다.

Table 7. Success Count in Mountain Car

Metric	SRM	PER	MRM
Average	349.47	14.53	**494.87**
Maximum	588	103	**631**
Minimum	0	0	**267**
SD	164.35	23.92	80.58
CV	0.47	1.65	**0.16**
p-value	-	2.15×10^{-12}	4.24×10^{-5} 5.94×10^{-27}

Fig. 8은 Mountain Car 문제에 대한 에피소드별 성공 횟수의 누적치와 스텝 횟수를 나타낸 것이다. 스텝 횟수의 경우 초반에는 30회의 모든 실험에서 목표 지점에 도달하지 못한 채 200 스텝까지 진행 후 종료됨을 알 수 있고 이후로 학습을 통해 200 스텝 이전에 성공하는 경우가 점점 많아짐을 알 수 있다. 특히 MRM의 성능이 SRM이나 PER보다 훨씬 우수함을 확인할 수 있으며, 이를 통해 대상 문제에 따라 MRM이 유용할 수 있음을 알 수 있다.

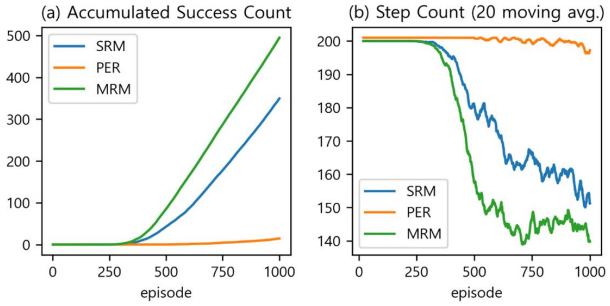

Fig. 8. Experimental Results on Mountain Car

Table 8은 Ball Avoider 문제에 대한 실험 결과로 성공 횟수를 요약한 것이다. 리플레이 메모리가 3개인 MRM과 12개인 MRM을 구별하기 위해 각각 MRM3과 MRM12로 표기하였다. Ball Avoider 문제의 경우 SRM과 PER에 의한 학습이 매우 어려운 것으로 보인다. 반면에 MRM은 상대적으로 매우 자주 성공하는 것으로 나타났다. 더욱이 MRM12가 MRM3보다 훨씬 더 자주 성공하고 있음을 알 수 있다. 이것만으로 리플레이 메모리의 개수가 많을수록 MRM의 성능이 향상된다고 볼 수는 없다. 대상 문제마다 적절한 리플레이 메모리 개수, 즉, 경험의 종류를 결정하는 것이 더욱 중요할 것으로 판단된다. 한편 변동 계수와 p-value를 통해 MRM12의 학습 안정성과 우수성을 재확인할 수 있다.

Table 8. Success Count in Ball Avoider

Metric	SRM	PER	MRM3	MRM12
Average	2.07	5.7	464.67	**1161.93**
Maximum	23	13	1822	**2012**
Minimum	0	0	1	**107**
SD	5.3	3.83	384.73	520.28
CV	2.56	0.67	0.83	**0.45**
p-value	-	0.0018	1.62×10^{-7} 1.86×10^{-7}	2.97×10^{-13} 3.2×10^{-13} 1.3×10^{-7}

Fig. 9는 Ball Avoider 문제에 대한 에피소드별 성공 횟수의 누적치와 스텝 횟수를 그래프로 나타낸 것이다. 여

기서 스텝 횟수는 500 이동 평균을 사용하였다. Fig. 9 (a)와 (b)를 통해 MRM의 성능이 SRM과 PER보다 훨씬 우수하며, MRM12가 MRM3보다 더 우수함을 알 수 있다. 이를 통해 MRM의 우수성과 더불어 MRM을 어떻게 적용하느냐에 따라 성능이 달라질 수 있음을 확인할 수 있다.

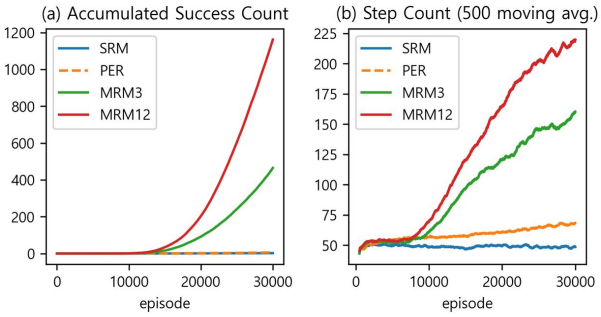

Fig. 9. Experimental Results on Ball Avoider

지금까지 실험을 통해 다중 리플레이 메모리의 효용성을 확인하였다. 추가로 Ball Avoider 문제에 대해서는 MRM12와 같은 다중 리플레이 메모리의 활용이 얼마나 타당한지를 분석하기 위한 실험을 수행하였다. Table 9는 Ball Avoider 문제에 있어서 Table 4에서 제시한 MRM12의 종류별 샘플링 개수인 [64, 64, 64, 64, 64, 64, 64, 64, 32, 32, 32, 32]에 비해 다른 기법들은 얼마나 유사하게 샘플링하는지를 확인하기 위한 실험 결과이다. Success Count는 성공 횟수를 의미하고 Step Count는 본격적인 학습이 시작되는 10000 에피소드 이후의 평균 스텝 수를 의미한다. Euclidean Distance는 유사도 지표로 사용한 유클리드 거리로 10000 에피소드부터 각 에피소드마다 각 종류별 샘플링 개수를 [64, 64, 64, 64, 64, 64, 64, 64, 32, 32, 32, 32]로 나누어 정규화한 후 [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]과의 유클리드 거리를 구해 평균을 취한 값이다. 참고로 MRM12는 유클리드 거리가 항상 0이므로 이를 제외하고 각 기법별로 3회의 실험 결과를 기술하였으며, 성능 지표 중 가장 중요한 Step Count 순으로 정렬하였다. Table 9를 통해 전반적으로 성능이 우수할수록 MRM12의 종류별 샘플링 개수와 더 유사하다는 것을 알 수 있다. 특히 MRM3과 같이 성능 차이가 뚜렷한 경우에는 유클리드 거리 또한 큰 차이를 보인다. 이는 무작위 샘플링이나 개별 경험 단위로 선택 여부를 결정하는 것보다 어떤 조합으로 경험들을 선택할 것인지가 학습 성능에 큰 영향을 미칠 수 있음을 시사한다. 본 연구에서 제시한 MRM12의 종류별 샘플링 조합이 가장 이상적인 상황은 아닐 수 있지만, Table 8, Fig. 9의 실험 결과와 더불어 Table 9의 실험 결과를 통해 매우 효과적임을 확인할 수 있다.

Table 9. Type-wise Selection Similarity w.r.t. MRM12 in Ball Avoider

Method	Trial Number	Success Count	Step Count	Euclidean Distance
MRM3	1	1038	157.64	157.73
	2	168	102.23	196.38
	3	56	70.80	250.70
PER	1	2	63.60	811.23
	2	5	59.44	810.31
	3	1	55.80	819.09
SRM	1	0	50.62	857.46
	2	1	45.55	859.83
	3	0	44.87	867.88

V. Conclusions and Future Works

본 논문에서는 심층 Q-학습에 있어서 다중 리플레이 메모리 기법의 적용 방안을 제시하였다. 기본적으로 심층 Q-네트워크 알고리즘을 그대로 적용하되 경험의 종류에 따라 별도의 리플레이 메모리를 두고 이를 활용한다. 3가지 대상 문제에 대한 실험 결과, 이와 같은 단순한 방법만으로도 대상 문제에 따라서는 학습 성능을 획기적으로 향상시킬 수 있음을 확인하였다. 중요한 것은 대상 문제에 따라 적절한 경험의 종류를 결정하는 것이고 이를 위해 대상 문제를 면밀히 분석할 필요가 있다. 그러나 대상 문제에 대한 지식이 부족하더라도 최소한 강화 학습 적용을 위한 현재 상태, 다음 상태, 보상 등의 정보가 주어지므로 다중 리플레이 메모리 기법을 우선적으로 적용해 볼 가치가 충분한 것으로 판단된다.

본 연구에서는 다중 리플레이 메모리의 유용성을 주로 실험을 통해 증명하였지만 이를 개선하기 위해서는 다음과 같은 추가 연구를 고려해 볼 수 있다. 첫째, 다중 리플레이 메모리 기법의 성능 향상을 위한 연구이다. 예를 들어, 각각의 다중 리플레이 메모리 내에서 경험 선택 시 우선순위 경험 재생 기법을 적용하는 방안을 고려해 볼 수 있다. 둘째, 다중 리플레이 메모리 기법의 자동화를 위해 다양한 대상 문제에 공통적으로 적용할 수 있는 경험 종류의 패턴을 찾는 것이다. 셋째, 다중 리플레이 메모리의 우수성에 대한 이론적 근거에 대한 연구이다. 가능하다면 다양한 종류의 경험을 함께 학습시켰을 때 얼마나 도움이 되는지 그리고 얼마나 다양한 경험들이 유지되고 선택되는지에 대한 정량적 지표를 개발하고 파악할 필요가 있다.

ACKNOWLEDGEMENT

This research was supported by Kumoh National Institute of Technology(2024~2025).

REFERENCES

[1] A. K. Shakya, G. Pillai, and S. Chakrabarty, "Reinforcement learning algorithms, A brief survey," Expert Systems With Applications, Vol. 231, 120495, Nov. 2023. DOI: 10.1016/j.eswa.2023.120495

[2] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Playing Atari with Deep Reinforcement Learning," arXiv preprint arXiv:1312.5602, 2013. DOI: 10.48550/arXiv.1312.5602

[3] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," Nature, Vol. 518, pp. 529-533, Feb. 2015. DOI: 10.1038/nature14236

[4] L. J. Lin, "*Reinforcement Learning for Robots Using Neural Networks*," Carnegie Mellon University, 1992.

[5] S. Zhang, and R. S. Sutton, "A Deeper Look at Experience Replay," Deep Reinforcement Learning Symposium, NIPS, 2017. DOI: 10.48550/arXiv.1712.01275

[6] Farama Foundation, Gymnasium, https://gymnasium.farama.org

[7] H. Park, M. Kang, and J. Hwang, "Multiple Replay Memory for Deep Q-Learning," Proceedings of UCWIT 2024, Nov. 2024.

[8] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," arXiv preprint arXiv:1511.05952, 2015. DOI: 10.48550/arXiv.1511.05952

[9] Y. Pan, J. Mei, A. Farahmand, et al., "Understanding and Mitigating the Limitations of Prioritized Experience Replay," Proceedings of the 38th Conference on Uncertainty in Artificial Intelligence, pp. 1561-1571, 2022.

[10] I. T. Nicholaus, and D. K. Kang, "Robust experience replay sampling for multi-agent reinforcement learning," Pattern Recognition Letters, Vol. 155, pp. 135-142, March 2022. DOI: 10.1016/j.patrec.2021.11.006

[11] Z. Lou, Y. Wang, S. Shan, et al., "Balanced prioritized experience replay in off-policy reinforcement learning," Neural Computing Applications, Vol. 36, No. 25, pp. 15721-15737, Sep. 2024. DOI: 10.1007/s00521-024-09913-6

[12] B. C. Han, M. J. Kang, and H. C. Kim, "Control Cart-Pole System Using Deep Reinforcement Learning," Proceedings of Information and Control Symposium, CICS´23, pp. 185-186, Oct. 2023.

[13] M. J. Kang, "DQN Reinforcement Learning for Mountain-Car in OpenAI Gym Environment," Proceedings of the Korean Society of Computer Information Conference, Vol. 32, No. 1, pp. 375-377, Jan. 2024.

[14] J. Yao, X. Li, Y. Zhang, et al., "Deep Q-Network Based on Important Experience Replay," In ISCTT 2022; 7th International Conference on Information Science, Computer Technology and Transportation, pp. 1-8, May 2022.

[15] T. Wan, and N. Xu, "Advances in Experience Replay," arXiv preprint arXiv:1805.05536, 2018. DOI: 10.48550/arXiv.1805.05536

Authors

Junha Hwang received the B.S., M.S. and Ph.D. degrees in Computer Engineering from Pusan National University, Korea, in 1995, 1997 and 2002, respectively. Dr. Hwang joined the faculty of the Department of Computer Engineering at Kumoh National Institute of Technology, Gumi, Korea, in 2002. He is currently a Professor in the Department of Computer Engineering, Kumoh National Institute of Technology. He is interested in AI, combinatorial optimization, and machine learning.