논문 2023-2-6 http://dx.doi.org/10.29056/jsav.2023.06.06

3D-Pinball 게임 기반 강화학습 알고리즘 및 모델 성능 분석

김경수*, 윤영선*†

Performance Analysis of Reinforcement Learning Algorithms and Models Based on the 3D-Pinball Game

Kyeongsoo Kim*, Young-Sun Yun*†

요 약

강화학습을 적용한 게임 연구는 인공지능이 사람보다 뛰어날 수 있다는 것을 입증하였으며, 다양한 방법들이 제안되었다. 본 논문에서는 기존 연구에서 뛰어난 성능을 보인 'Video Pinball'과 유사하지만, 화면 및 구성이 더 복잡한 '3D-Pinball'에 다양한 모델 구성 및 개선된 강화학습을 적용하였다. 시각적 데이터를 사용하는 CNN 기반의 모델과 고차원 데이터에서 적절한 특징을 추출하는 MLP 기반의 모델에 DQN, Double DQN, Dueling Double DQN 총 3가지 강화학습 알고리즘을 '3D-Pinball'에 적용하여 성능을 비교 및 평가한다. 실험은 빠른 진행을 위해 게임 플레이 시간 및 게임 목숨에 제한을 두고 진행했으며 실험 결과, MLP 기반의 Dueling Double DQN 알고리즘을 적용한 모델이 가장 높은 성능증가율과 게임 성적을 보여 제한된 환경에서는 단순한 모델의 성능이 우수함을 보였다. 따라서, 학습 및 게임 환경의 우수한 성능을 위해서는 제약사항을 고려한 모델 개발이 필요할 것으로 보인다.

Abstract

Game research applying reinforcement learning has demonstrated that AI can outperform humans. In this study, we extend this approach to '3D-Pinball', a game similar to 'Video Pinball' but with greater complexity. Three reinforcement learning algorithms – DQN, Double DQN, and Dueling Double DQN – are applied to both CNN-based models using visual data and MLP-based models extracting appropriate features from high-dimensional data. These are then evaluated in '3D-Pinball'. Restrictions on gameplay time and lives were implemented for efficient experimentation. Results showed that the Dueling Double DQN algorithm applied to the MLP-based model yielded the highest performance increase and game scores, underscoring the superior performance of simpler models in restricted environments. Consequently, it appears that model development considering constraints is required for superior performance in learning and game environments.

한글키워드 : 강화학습, 인공지능, 딥러닝, 게임, 전처리

keywords : Reinforcement Learning, Artificial Intelligence, Deep Learning, 3D-Pinball, Preprocessing

* 한남대학교 정보통신공학과
† 교신저자: 윤영선(email: ysyun@hnu.kr)

접수일자: 2023.05.29. 심사완료: 2023.06.09.
게재확정: 2023.06.20.

1. 서 론

강화학습(Reinforcement Learning)은 인공지능 연구의 핵심 분야 중 하나로, 인간이 복잡한 환경에서 어떻게 적응하고 행동하는지를 분석하여 컴퓨팅 개념에 적용한 것이다. 강화학습의 핵심은 시스템이 상호작용하는 환경(Environment)으로부터 얻은 정보(State)를 통해 최적의 행동(Action)을 학습하는 것이다.

DeepMind의 Atari 2600 게임 연구[1]는 인공지능이 강화학습을 통해 사람보다 뛰어난 성능을 보일 수 있다는 것을 입증한 바 있다. 본 연구는 기존 연구에서 가장 높은 성능을 보인 'Video Pinball'과 같은 게임 종류에 초점을 맞추었지만, 더 복잡한 화면 및 구성을 갖는 '3D-Pinball'에 다양한 강화학습 방법을 적용하여, 그 가능성과 성능 변화를 분석하고자 한다.

기존 게임 연구[1]에서는 DQN(Deep Q-Network) 알고리즘[2]을 사용하는 CNN (Convolutional Neural Networks)[11] 기반의 모델이 사용되었다. CNN 기반의 모델은 입력 이미지에서 특징을 추출하는 합성곱 연산을 사용하여, 이미지 분야에서 탁월한 성능을 보인다. 합성곱 연산을 이용하여 고차원의 시각적 데이터를 지역적 특징 단위로 해석하고 처리하여 상위층으로 특징 정보를 전달하기 때문에, 복잡한 구조와 많은 계산이 필요하고, 파라미터의 수가 많아진다는 단점을 가지고 있다. 이에 반해, MLP(Multilayer Perceptron)[12,13] 기반의 모델은 입력층, 은닉층, 출력층으로 구성된 간결한 구조를 가지며 입력 데이터 간의 복잡한 비선형 관계를 표현하고 학습 시간과 파라미터 수를 줄이는데 이점이 있다. 따라서 본 논문에서는 MLP 기반의 모델이 성능 측면에서 어떤 이점이 있는지, 얼마나 효율적인지에 대해 CNN 기반 모델과의 비교 분석을 진행한다.

기존 DQN 알고리즘을 개선하여 Double DQN 알고리즘(DDQN)이 제안된 연구[6]에서는 일부 Atari 게임의 성능 향상을 보였으며, Dueling Network와 DDQN 알고리즘을 결합한 방식[8]을 사용하여 성능 향상을 보이기도 하였다. 해당 연구를 바탕으로 본 논문에서는 DQN, DDQN, Dueling DDQN 총 3가지 알고리즘을 사용하여 성능을 평가했다.

3D-Pinball은 Microsoft의 Windows XP 운영체제에 포함되었던 아케이드 게임으로, 물리적 상호작용 및 시간에 따른 동적 전략이 필요한 게임이다. 3D-Pinball 게임은 높은 점수를 얻을 수 있는 목표 지점이 게임 진행에 따라 동적으로 변한다. 이는 Video Pinball 게임과의 가장 큰 차이점이다. 두 게임의 구성은 그림 1과 같다.

(a) Video Pinball (b) 3D-Pinball

그림 1. Video Pinball 및 3D-Pinball 화면 구성

Fig. 1. Screens of Video Pinball and 3D-Pinball

본 논문에서는 3D-Pinball 게임에 DQN[2], Double DQN[6]과 Dueling DDQN[8]의 3가지 강화학습 알고리즘을 적용한 MLP 기반 모델과 CNN 기반 모델의 성능 및 사용되는 파라미터 수를 비교하고자 한다. 이를 위해 강화학습을 위한 게임 환경을 구성하고, MLP 기반 모델의 특징 추출 방법과 보상 함수를 설계하는 방법을 제안한다.

본 논문의 구성은 다음과 같다. 2장에서는 강화학습 알고리즘에 대한 이론을 소개한다. 3장에

서는 3D-Pinball 게임의 구조와 특징을 소개하고, 강화학습 게임 환경을 구성하는 방법을 설명한다. 4장에서는 학습된 에이전트의 성능을 평가하고, 실험 결과를 분석한다. 마지막으로 5장에서는 결론을 맺는다.

2. 강화학습 이론

강화학습은 에이전트가 환경과 상호작용하며 보상을 최대화하는 행동 전략을 학습하는 것이다. 이 과정은 마르코프 결정과정(Markov Decision Process, MDP)으로 모델링된다. MDP는 상태의 집합 S, 행동의 집합 A, 환경의 전이 함수 $P(s_{t+1}|s_t, a_t)$, 보상 함수 $R(s_t, a_t, s_{t+1})$로 정의된다. 강화학습 문제를 해결하기 위해 하나의 에피소드로부터 발생한 궤적(τ)에서 얻는 이득은 식(1)과 같이 각 궤적에서의 보상(r_t)의 합으로 정의된다.

$$R(\tau) = r_0 + \gamma r_1 + \gamma^2 r_2 + ... + \gamma^T \tau_T = \sum_{t=0}^{T} \gamma^t r_t \quad (1)$$

γ는 감가율(Discount Factor)을 의미한다. MDP로 표현된 강화학습에서, 특정 정책(π)을 따르는 에이전트가 달성하고자 하는 목적(J)은 식(2)와 같다.

$$\mathrm{J}(\tau) = \mathbb{E}_{\tau \sim \pi}[\mathrm{R}(\tau)] = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \quad (2)$$

위의 MDP를 기반으로 어떤 수식을 강조하느냐에 따라, 크게 정책 기반(Policy Based) 알고리즘과 가치 기반(Value Based) 알고리즘으로 구분된다. 정책 기반 알고리즘은 행동 생성 함수인 정책 함수(π)를 학습하는 것이며, 가치 기반 알고리즘은 이득의 기댓값인 $\mathbb{E}_{\tau}[R(\tau)]$을 추정하는 것이다. 정책 기반 알고리즘은 식(3), (4)와 같이 최대 정책 함수(π)를 구성하는 θ를 찾는 방법이다.

$$\max_{\theta} \mathrm{J}(\pi_\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[R(\tau)] \quad (3)$$

$$\theta \leftarrow \theta + \alpha \nabla_\theta \mathrm{J}(\pi_\theta) \quad (4)$$

반면, 가치 기반 알고리즘은 이득의 기댓값에 따라 정책 π의 이득 $V^\pi(s)$과 특정 상태에서의 행동의 이득 $Q^\pi(s, a)$을 추정한다. 각 이득은 식(5), (6)으로 표현된다.

$$V^\pi(s) = \mathbb{E}_{s_0 = s, \tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \quad (5)$$

$$Q^\pi(s, a) = \mathbb{E}_{s_0 = s, a_0 = a, \tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r_t\right] \quad (6)$$

2.1 DQN(Deep Q-Network)[2]

DQN 알고리즘은 기존 Q-Learning[3] 알고리즘에서 사용되는 Q-Table을 인공신경망(Artificial Neural Network, ANN)[4]으로 교체한 형태이다. Q-Learning 알고리즘은 가치 함수 $Q^\pi(s, a)$를 계산하기 위해 모든 상태-행동 쌍에 대한 기댓값을 표 형태로 저장하는 Q-Table을 사용한다. 하지만 상태 및 행동의 차원이 증가함에 따라 Q-Table의 크기는 기하급수적으로 증가한다. 이 단점을 해결하기 위해 DQN은 식(7)과 같이 인공신경망을 통해 Q-Table을 비선형 함수로 근사하고 기댓값을 계산한다. 식(7)에서 $r(s_t, a_t)$는 상태 s_t에 대한 행동 a_t에 따른 보상량을 의미한다.

$$\left[Q^\pi(s_t, a_t; \theta) - \left(r(s_t, a_t) + \gamma \max_a Q^\pi(s_{t+1}, a; \theta)\right)\right]^2 \quad (7)$$

DQN 알고리즘에서는 학습 데이터의 상관관계(correlation)를 줄이기 위해 Experience Replay(ER) 방식을 사용한다. ER는 에이전트가 과거 행동들을 기억하는 저장 공간이며, 환경으로부터 얻어지는 모든 궤적을 (s_t, a_t, r_t, s_{t+1}) 형태로 메모리에 저장하고, 랜덤하게 k개의 데이터를 선택하여 학습을 진행한다.

2.2 Double DQN[6]

강화학습 에이전트는 초기 상태에서 행동의 이득 $Q^{\pi}(s,a)$를 추정해야 한다. 미래에 대한 정보가 없기 때문에, 처음 추정되는 가치 함수는 무작위로 선택된다. 이 상태로 학습을 진행하면 특정 상태와 행동에 대해서 과대추정 현상이 발생할 수 있다. 이 문제를 해결하기 위해 Double Q-Learning[5] 알고리즘이 제안되었으며, DQN 알고리즘에 이 방식을 결합한 것이 Double DQN이다.

Double DQN에서는 2개의 모델 중 하나의 모델 θ_A에 대해 학습을 진행하고, 다른 모델 θ_B은 일정 주기마다 학습된 모델을 이용하여 업데이트한다. 2개의 모델 파라미터를 일정 주기마다 업데이트하지 않고, 연속적인 환경에서 모델 파라미터를 업데이트하는 방식[7]도 제안되었다. 식(9)와 같이 학습이 진행될 때마다 모델 선택 가중치(τ_{soft})를 사용하여 업데이트한다.

$$\theta_B \leftarrow \tau_{soft}\theta_A + (1-\tau_{soft})\theta_B \tag{8}$$

본 논문에서는 연속 모델 학습 방법인 식(8)의 방법을 사용하여 실험을 진행했다.

2.3 Dueling Network[8]

DQN의 가치 함수 $Q^{\pi}(s,a)$는 현재 상태에 대해 행동 가치를 직접 표현한다. 반면에 함수 $Q^{\pi}(s,a)$를 가치 함수와 이득 함수로 구분하여 표현하는 방식이 Dueling Network에서 제안된 방식이다. 이득 함수는 식(9)과 같이 정의된다.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{9}$$

따라서, Dueling Network의 가치 함수는 모델 파라미터 θ에 가치 함수 파라미터 α와 이득 함수 파라미터 β를 이용하여 식(10)과 같이 정의할 수 있다.

$$Q^{\pi}(s,a;\theta,\alpha,\beta) = V^{\pi}(s;\theta,\beta) + A^{\pi}(s,a;\theta,\alpha) \tag{10}$$

식(10)과 같이 정의하는 경우, 최종 도출된 함수 $Q^{\pi}(s,a;\theta,\alpha,\beta)$의 값이 $V^{\pi}(s;\theta,\beta)$ 함수의 비중과 $A^{\pi}(s,a;\theta,\alpha)$ 함수의 비중이 얼마나 되는지 추정하기 힘들기 때문에, $Q^{\pi}(s,a;\theta,\alpha,\beta)$ 의 값을 식(11)와 같이 수정하여 학습을 진행한다.

$$V^{\pi}(s;\theta,\beta) + \left[A^{\pi}(s,a;\theta,\alpha) - \max_{a' \in |A|} A^{\pi}(s,a';\theta,\alpha)\right] \tag{11}$$

최종적으로 $V^{\pi}(s;\theta,\beta)$ 함수는 상태의 가치에 집중하며, $A^{\pi}(s,a;\theta,\alpha)$ 함수는 행동의 이득에 집중한다. 가치 함수와 이득 함수 자체의 의미가 변경될 수 있지만, 모델의 학습 안정성을 높이기 위해 식(12)과 같은 방법이 최종적으로 제안되었으며 Dueling Network의 성능평가 실험은 식(12)의 방법을 사용하여 진행되었다.

$$V^{\pi}(s;\theta,\beta) + \left[A^{\pi}(s,a;\theta,\alpha) - \frac{1}{|A|}\sum_{a'} A^{\pi}(s,a';\theta,\alpha)\right] \tag{12}$$

3. 실험 구성

본 장에서는 시스템 연구에서 사용된 행동, 상태 및 보상을 정의한다. 그리고 게임의 핵심 정보를 입력으로 사용하는 MLP 기반의 모델과 원시 픽셀을 입력으로 사용하는 CNN 기반 모델 방식을 설명한다.

3.1 시스템 구성

본 연구에서 사용한 전체 시스템은 그림 2와 같다. 3D-Pinball의 화면 이미지는 MLP 기반 모델의 경우, 이미지로부터 상태 정보를 추출하여 특징으로 사용되고, CNN 기반 모델은 이미지 전처리, Frame Skipping 과정을 거친 후 입력 특징으로 사용된다. 두 방식에 의하여 행동과 가치

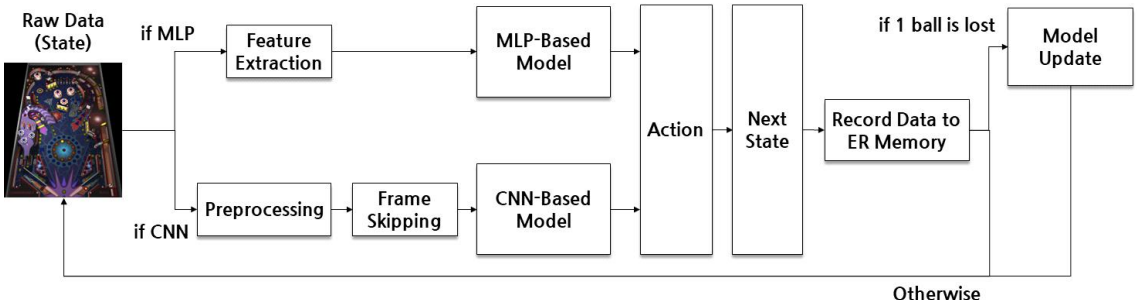

그림 2. 강화학습의 전체 과정
Fig. 2. The Entire Process of Reinforcement Learning

값이 산출되면, 에이전트에 의하여 다음 상태를 결정한 후 이전의 행동, 상태, 보상 및 다음 상태의 데이터를 ER 메모리에 저장한다. 학습 속도를 고려하고 단계적인 모델 업데이트를 위하여, 핀볼의 공이 소실되면 모델을 업데이트하도록 하였다. 만약 공을 잃지 않은 경우, 다음 상태를 현재 상태로서 사용하며 처음의 과정을 반복한다.

3.2 행동 정의

3D-Pinball 게임에서는 플런저, 좌측 및 우측 플리퍼, 틸트 총 4가지의 키를 조작할 수 있다. 3D-Pinball 게임에서 사용할 수 있는 모든 키에 대해 행동을 정의한 경우, 탐험(Exploration)으로 인한 틸트 및 플런저의 무작위 입력으로 인해 초반 학습 시간이 매우 길어지는 문제가 존재했다. 탐험은 항상 최댓값을 선택하는 DQN 알고리즘이 더욱 많은 상태 경험을 할 수 있도록 확률적으로 무작위 행동을 선택하게 하는 것이다. 따라서, 본 논문에서는 플런저와 틸트를 제외한 나머지 2개의 플리퍼 키의 행동을 정의하여 사용한다.

3.3 상태 정의 : MLP 기반 모델

MLP 기반의 모델은 공의 좌표 및 좌표 변화량과 플리퍼의 좌표로 이루어진 8가지 상태를 입력으로 사용한다. 각 플리퍼의 범위는 76개의 좌표 쌍으로 이루어지며, x, y좌표 합의 평균을 플리퍼의 대표 좌표로 사용했다. 학습에서 정의한 상태 정보는 표 1과 같다.

표 1. MLP 기반 모델의 상태 정의
Table 1. MLP-based Model State Definition

상태	정의
ball_x	current ball_x
ball_y	current ball_y
diff_ball_x	previous ball_x - ball_x
diff_ball_y	previous ball_y - ball_y
left_flipper_x	current left_flipper_x
left_flipper_y	current left_flipper_y
right_flipper_x	current right_flipper_x
right_flipper_y	current right_flipper_y

3.4 상태 정의 : CNN 기반 모델

CNN 기반의 모델은 게임 화면 이미지를 상태 입력으로 사용한다. 게임의 원본 이미지는 360x410 크기를 가지는 RGB 이미지이지만, 파라미터 수를 줄이기 위하여 Gray Scale 180x180으로 축소하여 사용하였다.

입력된 이미지에서 공의 움직임을 파악할 수 있도록 4개의 프레임을 묶어 하나의 입력으로 사용한다. 또한, 공의 움직임을 잘 파악할 수 있도록 연속된 프레임을 묶지 않고 일정 프레임을 건너뛰는 Frame Skipping을 사용한다. Frame Skipping 과정은 그림 3과 같다.

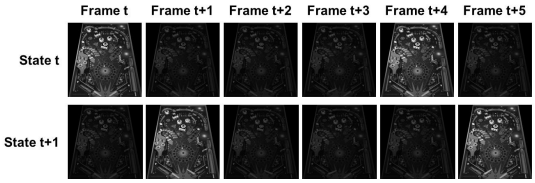

그림 3. 프레임 건너뛰기(Frame Skip=4)
Fig. 3. Frame Skipping(Frame Skip=4)

3.5 보상 정의

본 연구에서는 식(1)과 식(7)에 반영되는 궤적의 보상 r, 또는 $r(s_t,a_t)$를 부정적 행동과 긍정적 행동으로 구분하여 보상량을 정의한다. 부정적 행동은 게임에서 목숨을 잃거나, 플리퍼에서 공을 움직이지 않고 가두는 행위로 정의하며, 긍정적 행동은 게임 점수를 획득하는 경우로 정의한다. 정의된 보상은 표 2와 같다.

표 2. 보상 정의
Table 2. Reward Definition

상황	보상량
목숨을 잃은 경우	-10
공이 움직이지 않는 경우	-1
점수를 획득한 경우	점수변동량 * 0.001

3.6 모델 구성

MLP 기반 모델과 CNN 기반 모델 모두 DQN, Double DQN, Dueling DDQN 총 3가지의 알고리즘을 사용하여 모델을 구성했다. DQN 알고리즘을 적용한 MLP 및 CNN 기반의 모델은 Dueling Network 모델 구조의 가치 출력 레이어를 사용하지 않고 이득 출력 레이어만 사용하며, Dueling DDQN 알고리즘을 적용한 모델은 동일한 Dueling Network 모델을 2개 사용한다.

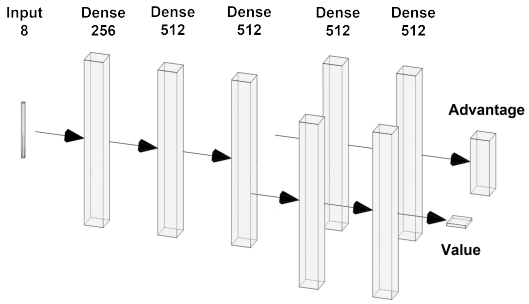

그림 4. MLP 기반 모델의 Dueling Network
Fig. 4. Dueling Network of the MLP-based Model

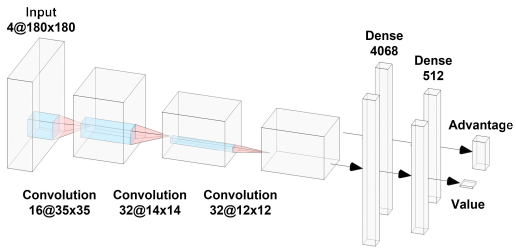

그림 5. CNN 기반 모델의 Dueling Network
Fig. 5. Dueling Network of the CNN-based Model

표 3. MLP 및 CNN 기반 모델의 파라미터 정보
Table 3. Parameter Information of MLP and CNN Based Model

모델	알고리즘	파라미터 수	파라미터 크기(MB)
MLP 기반	DQN	399,109	1.52
	DDQN	798,218	3.04
	DDDQN	1,849,868	7.06
CNN 기반	DQN	2,410,837	9.2
	DDQN	4,821,674	18.4
	DDDQN	9,542,316	36.4

Dueling Network를 사용한 MLP 기반의 모델은 그림 4과 같으며, CNN 기반의 모델은 그림 5와 같다. 구성된 모든 모델의 파라미터 정보는 표 3과 같다. DDQN은 Double DQN을 의미하며, DDDQN은 Dueling DDQN을 의미한다.

4. 실험 및 결과 분석

모든 실험은 파이썬 및 파이토치 라이브러리를 사용해 진행했다. 실험에서 사용한 활성화 함수는 ReLU, 최적화 방식은 Adam[9] Optimizer를 사용했고, 손실 함수는 Smooth L1[10] Loss를 사용했다. Smooth L1 Loss의 정의는 식(13)과 같으며, 절대 값 손실 함수의 특성을 갖는 L1 Loss와 제곱 손실 함수 특성을 갖는 L2 Loss의 조합으로, 예측값과 실제값의 차이가 작을 때는 L2 Loss를 사용하고, 차이가 클 때는 L1 Loss를 사용한다. 본 실험에는 1.0을 β 값으로 사용했다.

$$\begin{cases} 0.5(x_n - y_n)^2/\beta & \text{if } |x_n - y_n| < \beta \\ |x_n - y_n| - 0.5 \times \beta & \text{otherwise} \end{cases} \tag{13}$$

각 모델의 성능은 에피소드별 총 보상과 평가점수로 측정한다. 평가점수(E)는 플레이 시간(T)과 획득점수(S)의 합을 사용하였으며 서로 다른 크기의 T와 S를 보정하기 위하여 S는 제곱근을 취하였다. 평가점수의 계산은 식(14)와 같으며, α는 0.2의 값을 사용했다.

$$E = \alpha T + (1-\alpha)\sqrt{S} \tag{14}$$

표 4. 하이퍼파라미터 정보
Table 4. Hyperparameter Information

항목	MLP 기반	CNN 기반
learning rate	0.0001	
gamma	0.995	
tau	0.0005	
exp. start	0.9	
exp. end	0.01	
exp. decay	5000	
update freq.	4	
batch size	128	64
ER size	100,000	50,000
frame_skip	-	4
frame_stack	-	4

표 5. 실험 PC 사양
Table 5. Experimental PC Specifications

구분	사양
CPU	AMD Ryzen 7 3800XT
GPU	NVIDIA RTX 2080 SUPER
RAM	32GB
OS	Windows 11

실험에서 사용한 하이퍼파라미터와 PC의 사양 정보는 표 4 및 표 5와 같다. 표 4의 exp. 항목은 탐험(Exploration)을 의미한다.

4.1 실험 조건

3D-Pinball은 기본적으로 3개의 공을 가지고 1개씩 소모하는 형식으로 진행된다. 각 모델의 성능 변화를 빠르게 관찰하기 위해 실험의 에피소드는 3개의 공을 모두 사용하지 않고, 1개의 공만 사용하여 진행했다. DQN 알고리즘은 항상 최대값을 선택하기 때문에, 플리퍼에 공을 가두는 행위가 자주 관찰된다. 해당 상황으로 훈련 시간이 길어지는 문제를 해결하기 위해 최대 게임 플레이 시간을 200초로 제한하여 진행했다. 추가로 게임 조작 요소인 플런저를 사용하지 않았기 때문에 시작 지점에 공이 위치한 경우 자동으로 공을 발사하며 공이 발사된 시점부터 게임 플레이 시간을 측정했다. 본 논문의 모든 실험은 2000번의 에피소드를 기준으로 진행되었다.

CNN 기반 모델의 경우 4x180x180 크기를 가지는 데이터를 입력으로 사용한다. ER 메모리에는 현재 상태(state)와 다음 상태(next_state)가 동시에 저장되기 때문에 메모리 제약사항이 존재한다. 표 4와 같이, CNN 기반 ER 메모리의 크기가 50,000일 경우 약 12.06GB 공간의 메모리가 필요하다. 반면, MLP 기반 모델은 같은 조건에서 CNN 기반 모델 메모리 요구량의 약 0.012%만 사용한다.

4.2 MLP 기반 모델 실험 결과

그림 6은 MLP 기반 모델의 에피소드별 보상의 이동평균선을 나타내며 그림 7은 에피소드별 평가점수의 이동평균선을, 그림 8은 모든 모델의 에피소드별 평균 손실을 나타낸다.

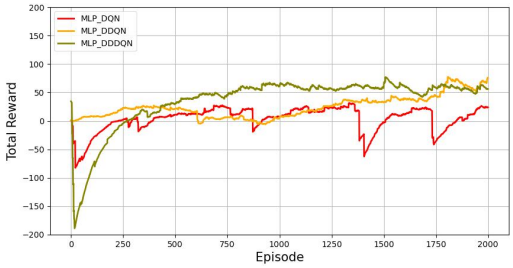

그림 6. MLP 기반 모델의 에피소드별 총 보상
Fig. 6. Total Rewards per Episode for MLP-based Model

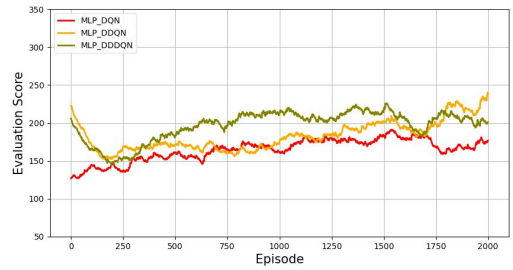

그림 7. MLP 기반 모델의 평가점수
Fig. 7. Evaluation Score per Episode for MLP-based Model

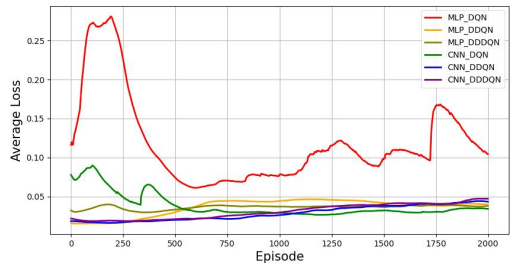

그림 8. 모든 모델의 에피소드별 평균 손실
Fig. 8. Average Loss per Episode for All Models

DQN 알고리즘은 특정 시점마다 보상이 하락하는 경우가 자주 관찰되었으나, 평가점수에서는 지속적인 성능 향상이 관찰되었다. 그림 8에서, DQN 알고리즘을 적용한 모델의 평균 손실이 불안정한 모습을 보여준다. 따라서 DQN 알고리즘이 다른 알고리즘에 비해 과대추정 문제로 학습 속도가 느리다는 것을 알 수 있다. DDQN 알고리즘의 경우 성능 향상의 정체 없이 보상 및 평가점수가 단조롭게 증가하는 형태를 보인다. DDDQN 알고리즘은 가장 빠른 성능 변화가 관찰되었으나, 750 에피소드 이후 성능이 변하지 않는 현상이 관찰되었다.

4.3 CNN 기반 모델 실험 결과

그림 9는 CNN 기반 모델의 에피소드별 보상의 이동평균선을 나타내며 그림 10은 에피소드별 평가점수의 이동평균선을 나타낸다.

DQN 알고리즘은 특정 시점의 보상 하락이 MLP 기반 모델과 유사하게 관찰되었다. 하지만 평가점수는 큰 변화를 보이지 못했다. DDQN 알고리즘은 1000 에피소드 이후 보상 및 평가점수의 지속적인 성능 향상이 관찰되었다. 하지만 MLP 기반 모델과는 다르게 DDDQN 알고리즘은 성능 향상이 관찰되지 않았다.

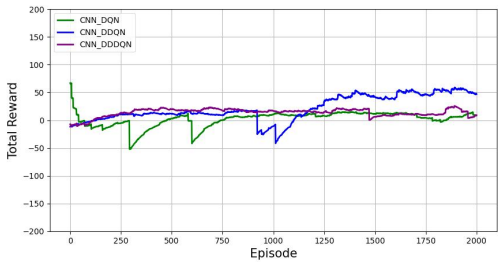

그림 9. CNN 기반 모델의 에피소드별 총 보상
Fig. 9. Total Rewards per Episode for CNN-based Model

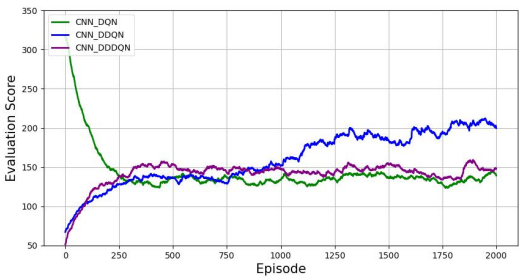

그림 10. CNN 기반 모델의 평가점수
Fig. 10. Evaluation Score per Episode for CNN-based Model

표 6. 구간별 종합 성능평가
Table 6. Comprehensive Performance Evaluation per Segment

모델	알고리즘	평균 총 보상			평균 평가점수 (단위: 10^{-3})		
		1~1000 에피소드	1001~2000 에피소드	전체 에피소드	1~1000 에피소드	1001~2000 에피소드	전체 에피소드
MLP 기반	DQN	0.36	9.17	4.79	6.32	7.86	7.1
	DDQN	11.21	44.5	27.84	6.82	10.33	8.57
	DDDQN	13.68	**57.84**	**35.78**	**8.89**	**10.43**	**9.66**
CNN 기반	DQN	-10.49	9.21	-0.64	4.5	4.73	4.61
	DDQN	6.84	41.48	24.15	4.95	9.34	7.14
	DDDQN	**16.09**	13.24	14.66	5.43	5.33	5.38

4.4 결과 분석

표 6은 MLP 및 CNN 기반 모델의 모든 알고리즘의 구간별 평균 보상 및 평가점수를 나타낸다. 모든 항목 중 MLP 기반의 DDDQN 알고리즘이 1~1000 에피소드 구간의 평균 총 보상 항목을 제외하고 가장 큰 값을 기록했다.

실험 결과, CNN 기반 모델은 MLP 기반 모델보다 약 6배 더 많은 파라미터를 가지고 있는 것을 고려하면 충분한 학습 요구량에 도달하지 못해 CNN 기반 모델의 성능이 상대적으로 낮게 나타난 것으로 판단된다.

CNN 기반 DQN 알고리즘은 과대추정 문제가 성능 향상을 방해한 것으로 보이며, DDQN 알고리즘은 이를 개선한 접근 방식으로 인해 성능 향상이 나타난 것으로 분석된다. DDDQN 알고리즘은 많은 파라미터 개수에 충분한 학습이 이루어지지 못해 성능이 낮게 나온 것으로 판단된다.

본 논문의 실험은 3개의 공을 모두 사용하지 않고 200초의 시간제한을 두어 빠른 학습을 목표로 진행되었다. 게임 진행시간이 제한된 환경에서는 적절한 특징을 추출하여 훈련하는 MLP 기반의 방식이 효과적인 것으로 보인다.

5. 결 론

본 논문은 게임 화면 이미지를 사용하는 CNN 기반의 모델과 상태 정보를 추출하여 사용하는 MLP 기반 모델에 DQN, DDQN, DDDQN 알고리즘을 적용하여 강화학습 에이전트의 성능을 평가하였다. 또한, 고차원 데이터에서 적절한 특징을 추출하는 방법을 제안하고 보상 함수를 설계했다. 실험 결과, MLP 기반 DDDQN 알고리즘을 사용한 모델이 빠른 성능변화를 보여주었으며 가장 높은 성능을 기록했다.

본 논문에서 진행된 실험은 빠른 학습을 위해 게임의 모든 기능을 사용하지 않고 진행되었다. 이에 따라 학습 환경의 제약사항으로 인해 CNN 기반 모델의 성능이 MLP 기반 모델에 비해 낮게 나타난 것으로 분석된다. 추후 연구에서는 환경의 제약사항을 제거하고 하이퍼파라미터의 조정을 시도하여 각 기반 모델의 성능 변화를 측정할 예정이다.

참 고 문 헌

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al., “Human-level control through deep reinforcement learning”, Nature, Vol. 518, No. 7540, pp. 529-533, (2015). DOI: 10.1038/nature14236

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al., “Playing Atari with Deep Reinforcement Learning”, arXiv preprint arXiv:1312.5602. (2013). DOI: 10.48550/arXiv.1312.5602

[3] C. J. C. H. Watkins and P. Dayan, “Technical Note: Q-Learning”, Machine Learning, Vol. 8, No. 3, pp. 279-292, (1992). DOI: 10.1007/BF00992698

[4] W. McCulloch. and W. Pitts., “A logical calculus of the ideas immanent in nervous activity”, The bulletin of mathematical biophysis, Vol. 5, No. 4, pp. 115-133, (1943). DOI: 10.1007/BF02478259

[5] H. van Hasselt, “Double Q-learning”, Proceedings of the 24th Annual Conference on Neural Information Processing Systems, pp. 2613-2621, (2010). https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf

[6] H. van Hasselt, A. Guez, and D. Silver, “Deep Reinforcement Learning with Double Q-learning”, arXiv preprint arXiv:1509.06461. (2015). DOI: 10.48550/arXiv.1509.06461

[7] T.P. Lillicrap, J.J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al., “Continuous Control with Deep Reinforcement Learning”, arXiv preprint arXiv:1509.02971. (2015). DOI: 10.48550/arXiv.1509.02971

[8] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. Freitas, “Dueling Network Architectures for Deep Reinforcement Learning”, arXiv preprint arXiv:1511.06581. (2015). DOI: 10.48550/arXiv.1511.06581

[9] D.P. Kingma, L.J. Ba, “Adam: A Method for Stochastic Optimization”, “International Conference on Learning Representations (ICLR)”, (2015). DOI: 10.48550/arXiv.1412.6980

[10] R. Girshick, “Fast R-CNN”, arXiv preprint arXiv:1504.08083. (2015). DOI: 10.48550/arXiv.1504.08083

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet Classification with Deep Convolutional Neural Networks”, Advances in neural information processing systems, pp. 1097-1105, (2012). https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[12] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, “Learning representations by back-propagationg errors”, Nature, vol. 323, pp. 533-536, (1986), DOI: 10.1038/323533a0

[13] C.M. Bishop, “Neural Networks for Pattern Recognition”, Oxford University Press, (1995), ISBN: 0198538642

저 자 소 개

김경수(Kyeongsoo Kim)

2021.2 한남대학교 정보통신공학과 졸업
2022.2-현재 한남대학교 정보통신공학과 석사 과정
<주관심분야> 강화학습, 인공지능, 딥러닝 등

윤영선(Young-Sun Yun)

2001.2 KAIST 전산학과 박사
2006.4-2007.2 한국전자통신연구원 초빙연구원
2012.8-2013.7 Univ. of Washington 방문학자
2001.3-현재 한남대학교 교수
<주관심분야> 음성인식, 음성변환, 화자인식, 인공지능, 저작권침해, 유사도, 완성도 감정, 오픈소스 등