Journal of Korea Robotics Society (2019) 14(3):157-162
https://doi.org/10.7746/jkros.2019.14.3.157 ISSN: 1975-6291 / eISSN: 2287-3961

심층 큐 신경망을 이용한 게임 에이전트 구현

Deep Q-Network based Game Agents

한 동 기[1] · 김 명 섭[1] · 김 재 윤[1] · 김 정 수[†]
Dongki Han[1], Myeongseop Kim[1], Jaeyoun Kim[1], Jung-Su Kim[†]

Abstract: The video game Tetris is one of most popular game and it is well known that its game rule can be modelled as MDP (Markov Decision Process). This paper presents a DQN (Deep Q-Network) based game agent for Tetris game. To this end, the state is defined as the captured image of the Tetris game board and the reward is designed as a function of cleared lines by the game agent. The action is defined as left, right, rotate, drop, and their finite number of combinations. In addition to this, PER (Prioritized Experience Replay) is employed in order to enhance learning performance. To train the network more than 500000 episodes are used. The game agent employs the trained network to make a decision. The performance of the developed algorithm is validated via not only simulation but also real Tetris robot agent which is made of a camera, two Arduinos, 4 servo motors, and artificial fingers by 3D printing.

Keywords: Deep Reinforcement Learning, DQN (Deep Q-Network), Tetris, PER (Prioritized Experience Replay)

1. 서 론

강화 학습과 딥러닝 기법을 결합한 심층 강화 학습은 게임, 로봇 등의 분야에서 이전까지 이루지 못했던 수준의 뛰어난 성능을 달성했고 이에 따라 많은 혁신적인 연구 결과가 만들어지고 있다. 처음으로 심층 강화 학습의 잠재력을 보여준 DQN (Deep Q-Network) 알고리즘은 컴퓨터 게임 화면의 픽셀 값을 입력으로 하는 인공 신경망을 사용하여 행동 가치 함수(action value function)를 근사하는 일관된 방법을 통해 아타리(Atari) 2600에 속한 대부분의 게임에서 사람을 뛰어넘는 성능에 도달할 수 있다는 것을 입증했다[1].

테트리스는 세계적으로 잘 알려진 비디오 게임으로 사각형 4개를 각기 다른 형태로 조합하여 만든 7가지 모양의 블록을 하나씩 순차적으로 쌓는 게임이다. 블록의 순서는 무작위로 생성되며 블록을 쌓아서 행을 빈 공간 없이 채우면 해당 행이 제거된다. 테트리스 게임의 목적은 블록을 빈 공간이 없도록 적절히 쌓아서 최대한 많은 행을 제거하는 것이다. 단순한 게임 규칙과는 달리 테트리스 보드에 블록이 쌓일 수 있는 형태의 경우의 수는 20×10의 정규 보드 크기를 기준으로 약 $2^{200}\approx1.6\times10^{60}$ 가지로 무수히 많으며, 테트리스의 최적 정책을 찾는 문제는 NP hard 라는 것이 알려져 있다[2]. 테트리스 게임의 최적 정책을 찾는 방법에 관한 기존 연구들에서는 보드의 형태를 효과적으로 반영하면서 경우의 수를 줄이는 여러 가지 수작업 특징 정의(hand-crafted feature extraction)와 더불어 다양한 최적화 방법이 제안되었다. [3]에서는 현재 테트리스 보드 형태의 좋고 나쁨을 나타내는 평가 함수를 Bertsekas 특징에 대한 선형 함수로 나타내고, 이를 cross-entropy method 기법으로 최적화하였다. [4]의 경우 평가 함수와 비슷한 의미를 가지는 상태 가치 함수(state value function)를 dellacherie-thiery 특징에 대한 선형 함수로 표현하고, 이를 근사 동적 계획법(approximate dynamic programing) 기반으로 최적화하는 방법을 사용하였다. 이와 같이 기존 연구에서 제안된 방법들은 정의된 특징에 대한 선형 함수 구조로 평가 함수 또는 가치 함수를 표현하고, 유전자 알고리즘 기반의 최적화를 적용하는 접근법이 주를 이루었다. 하지만 이러한 방식은 특징을 어떻게 정의하느냐에 따라 성능의 편차가 크고, 현재 보드 상태에 대한 평가 함수를 찾기 위해 가능한 모든 다음 상황을 시뮬레이션하여 다음 상황에 대한 함수 값이 가장 큰 경우를 찾는 방

Received : Mar. 15. 2019; Revised : May. 29. 2019; Accepted : Jun. 18. 2019
※ This work was supported by the Technology Innovation Program (or Industrial Strategic Technology Development Program) (10080636, Development of AI-based CPS technology for Industrial robot applications) funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea)
1. Students, Dept. of EIE, SeoulTech, Seoul, Korea (dongki.hann, kimmyungsup57@gmail.com, kywxo1@naver.com)
† Associate Professor, Corresponding author: Dept. of EIE, SeoulTech, Seoul, Korea (jungsu@seoultech.ac.kr)

Copyright ⓒKROS

식의 model-based 접근법이다. 또한 테트리스 게임의 중요한 특징 중 하나는 하나의 행동(action)을 취했을 때 그것이 한 줄을 제거하지 못하는 경우가 많고 따라서 보상을 얻지 못하게 되는 경우가 많다. 이러한 희소 보상 성질을 가지는 MDP는 [1]에서는 잘 해결하지 못하였다.

본 논문에서는 보드의 형태 자체를 상태로서 사용하여 테트리스의 MDP를 정의하였고, model-free 접근법으로 심층 강화 학습 알고리즘을 적용해 테트리스 게임의 인공지능 에이전트의 개발 및 성능 향상을 이끌었다. 또한 희소 보상을 해결하기 위한 한 방편으로 행동 묶음(group action)을 적용하였다.

2. 심층 강화 학습

2.1 마르코프 결정 과정과 강화 학습

강화 학습은 마르코프 결정 과정(MDP)으로 정의되는 환경으로부터 기대 누적 보상을 최대화하는 최적 정책을 찾는 여러 가지 방법 중 하나이다. 여기서 환경에 해당하는 마르코프 결정 과정은 $\{S, A, P, R, \gamma\}$의 요소로 구성된다. S는 유한한 크기를 갖는 상태 집합, A는 유한한 크기를 갖는 행동 집합, $P(s'|s,a)$는 상태 전이 확률로 현재 상태 $s \in S$에서 행동 $a \in A$를 취했을 때 다음 상태가 $s' \in S$이 되는 확률 분포를 의미하며, R는 보상 함수, $\gamma \in (0,1)$은 할인 계수를 나타낸다. 또한 강화 학습 에이전트의 역할을 하는 정책 $\pi(a|s)$는 주어진 상태에 대한 행동의 확률 분포를 의미한다. 강화 학습 에이전트는 이산 시간 t시점에서의 상태 s_t에서 정책 π에 따라 행동 a_t를 취하고 상태 전이 확률 및 보상 함수에 따라 다음 상태 s_{t+1}과 보상 r_{t+1}을 얻는다. 이때 강화 학습은 초기 상태로부터 정책에 따라 연속적인 행동을 취했을 때의 기대 누적 보상 $E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\right]$을 최대화하는 정책을 찾는 것을 목적으로 하며 이때의 정책을 최적 정책 π^*로 표현한다.

2.2 Q-learning

강화 학습은 정책 기반 알고리즘과 가치 기반 알고리즘으로 분류 가능하며, Q-learning은 행동 가치 함수를 이용하여 최적 정책을 찾는 가치 기반 알고리즘에 속한다. 행동 가치 함수 Q_π는 식 (1)과 같이 정의되며 상태 s에서 행동 a를 취했을 때 정책 π에 따른 기대 누적 보상을 의미한다.

$$Q_\pi(s,a) = E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | s_t = s, a_t = a\right] \quad (1)$$

이러한 Q_π를 실제 알아내는 것은 어려운 일이므로 여러 가지 방법으로 추정하게 되는데 대표적으로 임의의 값을 가지는 함수 Q에 대해 식 (2)를 반복하면 학습비율 α에 대한 기본 조건하에서 Q_π로 수렴함이 알려져 있다[5,6]. Q-learning은 임의의 값으로 초기화된 함수 Q에 대한 ϵ-탐욕 정책을 통해 주어진 상태에 대한 행동을 취하고 환경으로부터 다음 상태와 보상을 얻고 식 (2)를 반복하는 과정으로 최적 정책 π^*를 찾는다.

$$Q(s_t,a_t) = Q(s_t,a_t) + \alpha(r_{t+1} + \gamma \max Q(s_{t+1}, \bullet) - Q(s_t,a_t)) \quad (2)$$

2.3 심층 Q 신경망 (Deep Q-Network)

고전적인 방식의 Q-learning은 상태 집합의 크기가 커지면 차원의 저주가 발생하기 때문에 실제 문제에 적용하는 것에 제약이 존재했다. 이에 DQN 알고리즘은 행동 가치 함수를 인공신경망을 통해 함수 근사 및 학습하는 방법을 사용한다. Q-신경망은 입력으로 상태가 인가되면 각 행동에 대한 행동 가치 함수를 출력한다. 신경망은 식 (3)으로 표현되는 시간차 오차의 제곱을 목적 함수로 정의하고 이에 대한 경사 하강법으로 학습한다.

$$L(\theta) = (r_{t+1} + \gamma \max Q_{\theta^-}(s_{t+1}, \bullet) - Q_\theta(s_t,a_t))^2 \quad (3)$$

이 때 [7]에서 제안한 double Q-learning 기반의 목적 함수로 확장 적용 가능하며 이는 식 (4)로 표현된다. 또한 DQN 알고리즘은 학습의 안정성을 위해 경험 재생 메모리 및 타겟 신경망을 사용한다[1].

$$L(\theta) = (r_{t+1} + \gamma Q_{\theta^-}(s_{t+1}, \max Q_\theta(s_{t+1}, \bullet)) - Q_\theta(s_t,a_t))^2 \quad (4)$$

2.4 우선순위 경험 재생(Prioritized Experience Replay)

우선순위 경험 재생은 DQN 알고리즘의 성능 향상 기법 중 한 가지로서, 기존 DQN 알고리즘에서는 경험 재생 메모리에 저장되어 있는 데이터를 무작위로 선택하여 학습하는 반면에 PER의 경우 데이터에 대한 우선순위 가중치를 부여하여 이 가중치를 반영한 확률로 데이터를 선택한다[8]. i번째 데이터의 우선순위 가중치 p_i는 시간차 오차의 크기에 비례하며 식 (4)로 정의된다. 이에 따라 i번째 데이터가 선택될 확률 $P(i)$는 식 (5)로 표현된다. 여기서 α는 우선순위 가중치를 얼마나 반영할지를 결정하는 변수이다. 우선순위 가중치에 따른 데이터 선택은 전체 데이터 분포를 반영하지 못하고 편향이 발생하기 때문에 이를 완화하는 방법으로 중요도 표집 기법(importance sampling)을 적용한다. i번째 데이터의 중요도 표집 가중치는

식 (6)로 정의되어 목적함수에 곱해진다. N은 메모리에 저장된 전체 데이터의 개수, β는 중요도 표집 기법을 얼마나 반영할지를 결정하는 변수이다.

$$p = |\delta_t| = |r_{t+1} + \gamma \max Q_{\theta^-}(s_{t+1}, \bullet) - Q_\theta(s_t, a_t)| \quad (5)$$

$$P(i) = \frac{p_i^\alpha}{\sum_j p_j^\alpha} \quad (6)$$

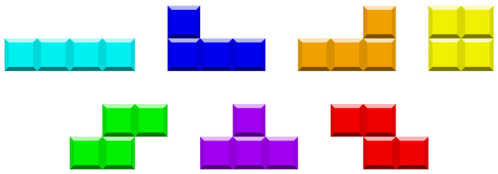
$$w_i = \frac{(N \bullet P(i))^{-\beta}}{\max w_j} \quad (7)$$

3. 테트리스 마르코프 결정 과정

고전적인 테트리스 비디오 게임은 20×10 크기의 보드를 사용하며 7가지 모양의 블록이 무작위 순서로 생성된다. 매 순간 게임 플레이어는 블록을 시계 방향으로 90° 회전, 좌로 한 칸 이동, 우로 한 칸 이동, 아래로 한 칸 이동, 아래로 바닥까지 이동의 5가지 행동을 선택할 수 있다. 또한 일정 시간마다 블록은 자동으로 아래로 한 칸 씩 이동하며 난이도가 올라갈수록 시간 간격은 줄어들어 블록은 더 빠르게 아래로 이동하게 된다. 본 논문에서는 문제를 단순화하고 학습 시간을 줄이기 위해 일반적인 테트리스 비디오 게임에서 몇 가지 규칙을 변경하였다. 첫째로 문제를 단순화하기 위해 일정 시간마다 블록이 자동으로 한 칸씩 아래로 이동하는 규칙을 배제하였다. 둘째로 다양한 실험을 진행하기 위해 보드의 크기를 14×7로 축소하여 한 번의 실험에 소요되는 학습 시간을 줄였다.

3.1 상태

기존 연구에서는 테트리스 보드의 상태를 잘 반영하면서 경우의 수를 줄일 수 있는 특징을 사람의 직관에 따라 정의하였다. 예를 들어 Bertsekas 특징의 경우 보드의 인접한 열 간의 쌓여있는 블록의 높이 차이, 가장 높게 쌓여있는 블록의 높이, 빈 공간의 개수로 정의된다. 이렇게 사람의 직관에 의존한 특징은 다양하게 정의될 수 있고, 어떤 특징을 사용하느냐에 따라 성능의 편차가 발생한다. 따라서 본 논문에서는 직관에 따른 특징 정의를 사용하지 않고 테트리스의 보드 형태 그대로를 상태로서 정의하고 이를 합성 곱 신경망(CNN, Convolution Neural Network)을 통해 특징 추출 과정 자체를 학습하도록 설계하였다. 합성 곱 신경망의 입력이자 테트리스 보드 형태 자체를 반영한 상태는 14×7 크기의 보드를 기준으로 같은 크기의 $14 \times 7 \times 1$ 행렬로 정의되며 각 행렬의 요소 값은 블록이 채워져 있는 경우에 1, 비워져 있는 경우에 0 값을 갖는다.



[Fig. 1] Tetromino : seven blocks used in Tetris games

3.2 행동

테트리스 비디오 게임에서 정의되어 있는 기본 행동을 그대로 학습에 사용한다면 강화 학습 에이전트의 학습 초기 무작위 탐험 과정에서 왼쪽으로 한 칸 이동하고 다시 오른쪽으로 한 칸 이동할 경우나 회전을 여러 번 하는 경우 등의 경험이 빈번하게 발생한다. 하지만 이런 경험은 줄 제거라는 목적을 고려했을 때 불필요한 탐험 과정이며 학습을 느리게 하는 원인이 된다. 본 논문에서는 현재 블록이 쌓일 수 있는 위치 및 형태의 가짓수에 따라 기본 행동을 조합하여 새로운 행동을 정의하였다. 예를 들어 [Fig. 1]의 T 모양의 블록은 회전을 통해 위로 볼록, 아래로 볼록, 좌로 볼록, 우로 볼록 으로 4가지 형태가 가능하다. 이때 각 형태에 대해 블록이 놓일 수 있는 위치는 14×7 크기의 보드를 기준으로 5가지, 5가지, 6가지, 6가지로 총 22가지 경우가 존재한다. 결국 해당 블록이 보드에 쌓일 수 있는 22가지 경우 그 자체가 새롭게 정의되는 행동이 된다. 이를 일반화하면 7가지 모양의 각 블록의 회전에 의한 형태 변화는 최대 4가지가 경우가 존재하며, 위치이동은 14×7 크기의 보드를 기준으로 최대 7가지가 경우가 존재한다. 따라서 새롭게 정의되는 행동의 집합 크기는 28이 된다.

3.3 보상

기본적으로 테트리스에서 점수를 얻는 방법은 보드의 행을 빈틈없이 채워서 줄을 제거하는 것이다. 이때 한 번에 제거하는 줄이 많을수록 추가점을 얻어 큰 점수를 얻는다. 따라서 보상 역시 줄을 제거할 경우에 발생하며 보상의 크기는 줄 제거 수에 비례하도록 정의하였다. 하지만 학습 초기 무작위 행동을 통해 블록을 빈 공간 없이 채워 줄을 제거하는 경우의 발생 빈도는 매우 적다. 따라서 줄 제거 보상만을 사용할 경우 보상을 얻는 경험의 빈도가 매우 드물기 때문에 학습에 어려움을 겪게 된다. 그러므로 학습을 보조하기 위한 용도로 몇 가지 추가 보상을 정의하였다. 줄 제거 보상 및 추가 보상을 모두 고려한 이산 시간 t 시점에서의 보상 r_t는 식 (8)로 표현된다.

$$r_t = r_{clear} + r_{hole} + r_{height} + r_{combo} \quad (8)$$

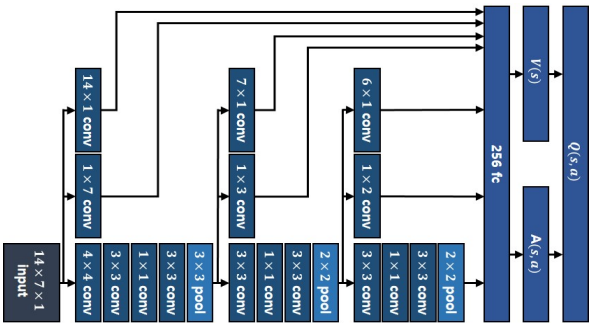
$$r_{clear} \in \{0, 1, 4, 10, 50\}, r_{hole} \in \{-0.01, 0.05\}$$
$$r_{height} \in \{-0.05, 0.05\}, r_{combo} \in \{0, 0.2\}$$

여기서 r_{clear}는 줄 제거 보상이며 순서대로 한 번에 0, 1, 2, 3, 4 줄을 제거 했을 때의 보상 크기를 나타낸다. r_{hole}은 현재 블록을 쌓는 순간에 빈 공간의 생성 여부에 따른 보상을 의미한다. 빈 공간이 발생할 경우 음의 보상, 그렇지 않을 경우 양의 보상을 받는다. r_{height}는 쌓여 있는 블록의 높이가 보드의 절반을 넘는 경우 음의 보상을 그렇지 않을 경우 양의 보상으로 정의된다. 마지막으로 r_{combo}의 경우 연속으로 줄을 제거했을 때 양의 보상을 제공한다.

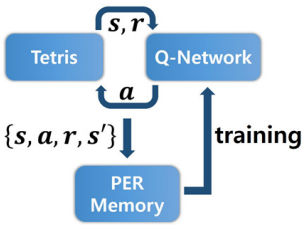
4. 학습 방법

4.1 인공 신경망 구조

[Fig. 2]는 본 논문에서 사용한 Q-신경망 구조를 나타낸다. 신경망은 테트리스 보드에서 특징 추출을 하는 합성 곱 신경망과 추출된 특징을 바탕으로 행동 가치 함수를 학습하는 완전 연결 신경망으로 이루어져 있다. 합성 곱 신경망의 경우 $7\times14\times1$ 크기의 입력에 대해 크게 3번의 합성 곱 블록 세트를 거치는 형태를 가진다. 첫 번째 블록 세트에서는 4×4 크기의 필터 32개로 구성되며 최종적으로 3×3 크기로 보폭 2의 max pooling을 거친다. 두 번째 블록 세트에서도 비슷한 형태가 반복된다. 또한 각 블록 세트를 거친 출력은 skip connection 구조로 완전 연결 신경망의 입력으로 연결된다. 추가적으로 테트리스는 행 또는 열의 모양이 중요한 정보를 담고 있을 것이라는 배경 지식을 활용하여 입력 및 각 pooling에 대해 $n_{height}\times1$, $1\times n_{width}$ 크기의 필터 64개를 갖는 합성 곱 신경망을 두었다. 이후 완전 연결 신경망은 256개의 뉴런으로 구성된다. 마지막으로 [9]에서 제안한 dueling Q-신경망 구조에 따라 상태 가치 함수와 이점 함수(advantage function)를 근사하는 완전 연결 신경망이 각각 256개의 뉴런을 구성하며 최종적으로 28개의 행동 가치 함수를 출력한다.



[Fig. 2] Q-network architecture: the Q-network consists of a convolution layer responsible for the process of feature extraction from a given input and a fully connected layer responsible for the approximation of the action value function



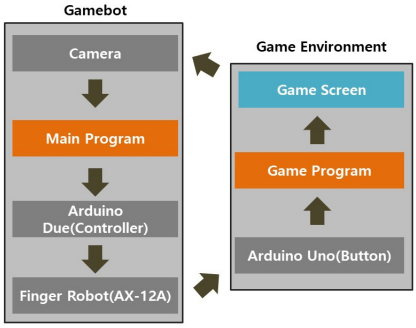
[Fig. 3] Agent training process : the Tetris agent performs training according to the DQN algorithm, and the experiences collected from the environment are stored and extracted by the PER method

4.2 학습 알고리즘

[Fig. 3]은 강화 학습 기반 테트리스 인공지능 에이전트의 학습 프로세스를 블록 다이어그램으로 표현한 그림이다. 먼저 초기 학습 데이터를 수집하는 과정에서는 주어진 상태 s에 대한 완전한 무작위 행동을 통해 행동 a를 취하고 이에 대한 보상 r과 다음 상태 s'을 얻는다. 이 과정을 한 번 진행할 때마다 상태 s, 행동 a, 보상 r, 다음 상태 s'을 하나의 학습 데이터 세트로서 메모리에 저장한다. 일정량의 학습 데이터가 메모리에 저장되면 본격적인 Q-신경망의 학습이 시작된다. Q-신경망 학습이 시작되면 매 순간 행동은 ϵ-탐욕 정책으로 취해지며 4번의 행동을 취할 때 마다 한 번의 Q-신경망 학습이 이루어진다. Q-신경망 학습의 경우 먼저 식 (5)와 식 (6)로 정의되는 우선순위 경험 재생 기법에 따라 메모리에서 32개의 데이터를 추출한다. 추출된 데이터로 식 (4)의 double Q-learning 기반 목적 함수에 식 (7)의 중요도 표집 기법을 적용하여 이에 대한 경사 하강법으로 Q-신경망의 가중치를 갱신한다. 타겟 신경망은 Q-신경망의 학습이 1000번 이루어질 때마다 한 번 갱신된다. 이러한 과정으로 총 48시간 동안 학습을 진행하였다.

5. 하드웨어 설계

이 장에서는 4장에서 설명한 학습 과정이 성공적으로 이루어진 후에 학습된 네트워크를 이용하여 테트리스 에이전트를 카메라와 손가락으로 이루어진 로봇으로 구현한다. 테트리스 에이전트와 게임 환경은 직접적인 데이터 교환을 하지 않고 카메라의 영상을 통해 수행할 행동을 계산한 후 로봇의 손가락으로 전달한다. 따라서 [Fig. 4]와 같이 로봇 에이전트와 테트리스 게임(MDP에 해당)과의 접점은 게임 화면을 보는 카메라와 버튼을 직접 누르는 로봇의 손가락 두 가지이다. 입력받은 게임 화면은 전 처리되어 미리 학습된 신경망으로 전달되고, 그 출력을 토대로 모터의 움직임을 결정한다. 게임 환경에서 가능한 입력은 블록의 좌측 이동, 우측 이동, 회전, 드롭과



[Fig. 4] Tetris AI Architecture: the left diagram is a procedure of the game robot, and the Game environment is nothing but the Tetris game screen with input buttons

[Fig. 5] Tetris agent robot: the left picture is the part of the finger robot with the arcade buttons, and the right one is the scene in which the camera recognizes the game screen[2)]

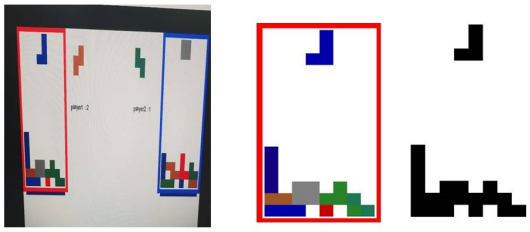
그들의 조합으로 정의했기에 로봇의 손가락은 네 개의 서보 모터(AX-12A[1)])와 3D 프린터로 제작한 손가락 모형으로 구성되었고 이들은 Arduino Due에 의해 제어된다. 이 손가락 들은 네 개의 입력 버튼을 눌러 게임을 진행한다. 신경망의 한 번의 출력은 묶음 행동으로 나오기에, 각 행동마다 네 개의 버튼을 몇 번씩 누를지 결정한 후 버튼을 순차적으로 누르도록 하였다.

[Fig. 5]는 최종적으로 완성된 테트리스 로봇을 보인다. 좌측 사진은 [Fig. 4]의 finger robot과 button의 역할을 하는 부분이며 각 버튼은 좌, 우, 드롭, 회전의 수행을 의미한다. 우측의 그림은 카메라를 통해 영상을 얻는 구도를 보여준다.

5.1 영상 인식 및 처리

시뮬레이션 환경에서 학습된 신경망을 실제 환경에 적용하기 위해 카메라에서 본 이미지와 이전에 학습한 이미지가 같도록 해야 한다. 또한 영상인식에 있어서 카메라의 기울어짐이나 영상의 휘어짐의 영향이 적도록 하고, 상대방의 게임화면을 자신의 화면으로 인식하는 것 또한 막아야한다. 그러기 위해서 먼저 게임 화면에 붉은색 테두리를 넣었고 이를 중심

1) http://robotis.com
2) 실제 로봇의 게임 실행 영상. https://goo.gl/XENRc6



[Fig. 6] Image data transforming: if the camera captures like the left image, it is transformed into a binary data shown in the right side figure

으로 인식하도록 하였다. [Fig. 6]의 좌측은 카메라 원본 이미지이고 우측의 흑백 이미지가 최종적으로 사용될 데이터이다. 인공지능 로봇이 자신의 화면만 인식할 수 있도록 윤곽선 검출(contour detection)을 통해 붉은색 사각형을 인식하고, 원근변환(perspective transform)을 통해 원본 게임 사이즈에 맞게 되돌린다. 이후 각 영역에 블록의 존재 유무에 따라 바이너리 이미지를 만들어낸다.

6. 성능 평가

DQN 알고리즘 기반의 테트리스 인공지능 에이전트의 성능 평가는 학습이 끝난 Q-신경망으로 1000번의 게임을 시뮬레이션 하여 평균 줄 제거수를 비교하는 방법으로 이루어졌다.

[Table 1]은 각각 테트리스의 기본 행동을 사용한 DQN, 본 논문에서 정의한 행동 묶음을 사용한 DQN, 행동 묶음 및 PER 기법을 적용한 DQN의 게임 당 줄 제거수를 나타낸다. 표에서 알 수 있듯이 기본 행동을 사용한 DQN 알고리즘의 성능은 매우 저조했다. 기본 행동을 사용할 경우 비효율적인 탐험에 따라 유용하지 않은 경험들이 메모리에 저장되기 때문에 학습이 거의 이루어지지 않았다고 해석할 수 있다. 행동 묶음을 사용한 DQN의 경우 불필요한 탐험이 줄어들기 때문에 기본 행동을 사용한 경우보다는 성능이 조금 향상되었다. 하지만 가장 큰 성능 향상은 행동 묶음 및 PER 기법을 적용한 DQN을 통해 이루어졌다. PER 기법은 시간차 오차가 큰 데이터를 더 높은

[Table 1] Performance of Tetris game agent based on DQN

Method	Episode	Time	Line clear (average)	Line clear (max)
DQN (step action)	10^5	16	4.19	11
DQN (group action)	10^5	9	8.06	29
DQN with PER (group action)	10^5	9	17.44	45
	5×10^5	45	87.44	713

확률로 샘플링한다. 또한 성능 평가의 기준이 되는 줄 제거에 대한 보상은 발생 빈도는 적지만 보상이 크기 때문에 시간차 오차가 클 가능성이 높다. 따라서 PER 기법을 적용할 경우의 성능 향상은 줄 제거시의 데이터 학습이 더 원활이 이루어지기 때문으로 추정할 수 있다.

7. 결 론

본 논문에서는 테트리스 게임에 대해 심층 강화 학습의 DQN 알고리즘을 적용하여 테트리스 인공지능 에이전트를 제작하는 과정을 소개하였다. 이를 위해 테트리스에 대한 마르코프 결정 과정을 정의하였으며, 보드 자체를 상태로 표현하고 행동 묶음의 사용 및 줄 제거 보상과 더불어 몇 가지 추가 보상을 제안하였다. 또한 테트리스 게임에 특화된 합성 곱 신경망 구조 및 PER 기법의 적용을 통해 성능 향상을 이끌었다. 향후 연구 주제로 정책 기반 알고리즘 개발을 통한 에이전트 성능 향상을 생각할 수 있다.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.

[2] E. Demaine, S. Hohenberger, and D. Liben-Nowell, "Tetris is hard, even to approximate," *In Proceedings of the Ninth International Computing and Combinatorics Conference*, pp. 351-363, 2003.

[3] I. Szita and A. Lörincz, "Learning Tetris using the noisy cross-entropy method," *Neural Computation*, vol. 18, no. 12, pp. 2936-2941, 2006.

[4] V. Gabillon, M. Ghavamzadeh, and B. Scherrer, "Approximate dynamic programming finally performs well in the game of Tetris," *In Advances in Neural Information Processing Systems*, pp. 1754-1762, 2013.

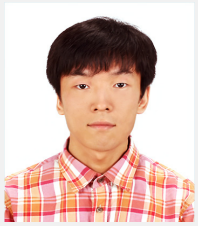
[5] C. J. C. H. Watkins and P. Dayan, "Technical note Q-learning," *Journal of Machine Learning*, vol. 8, pp. 279-292, 1992.

[6] T. Mitchell, "Machine Learning," McGraw Hill, 1997.

[7] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *In Proceedings of the Association for the Advancement of Artificial Intelligence*, pp. 2094-2100, 2016.

[8] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *In Proceedings of the International Conference on Learning Representations*, 2016 (arXiv:1511.05952).

[9] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," *In Proceedings of the International conference on Machine Learning*, vol. 48, pp. 1995-2003, 2016.



한 동 기

2018 서울과학기술대학교 전기정보공학과 (학사)

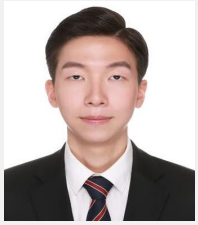
2018 8월~현재 서울과학기술대학교 전기정보공학과(석사과정)

관심분야: 딥러닝, 강화학습, 인공 지능

김 명 섭

2012~현재 서울과학기술대학교 전기정보공학과(학사과정)

관심분야: 딥러닝, 강화학습, 인공 지능



김 재 윤

2019 서울과학기술대학교 전기정보공학과 (학사)

관심분야: 기계학습, 로보틱스, 임베디드 시스템

김 정 수

2005 고려대학교 전기공학과, 박사

2006 서울대학교 박사 후 연구원

2007 슈튜트가르트대학 박사 후 연구원

2008 레스터대학 박사 후 연구원

2009~현재 서울과학기술대학교 교수

관심분야: MPC, 분산 제어, 최적화, 에너지 시스템, 인공 지능