DBpia

논문 19-44-10-24 The Journal of Korean Institute of Communications and Information Sciences '19-10 Vol.44 No.10
https://doi.org/10.7840/kics.2019.44.10.1961

N-DQN: 계층화된 병렬 강화학습 모델의 구현 및 연구

정 택 현◆, 김 상 원*, 김 기 천°

N-DQN: Study on the Implementation and Research of Hierarchical Parallel Reinforcement Learning Model

Tack-hyun Jung◆, Sang-won Kim*, Keecheon Kim°

요 약

본 논문은 강화학습(Reinforcement Learning)의 성능에 상관성을 갖는 여러 요인 및 조건들을 고찰하고, 이를 개선하기 위한 N-DQN 모델을 정의한다. N-DQN은 HDQN의 구조를 응용 및 확장하는 개념으로 여러 actor를 계층화하며 Policy-Base의 행동관리를 통해 작업이 동시/병렬적으로 수행되는 구조를 갖는다. 각각의 actor가 행동을 수행하면서 획득한 Episode는 Replay Buffer에 공유성 있게 저장되며, 이러한 과정에서 우선적 경험재생(Prioritized Experience Replay), 보상 획득구간의 세분화 등과 같은 다양한 강화학습의 개선요인을 적용하였다. 구현된 N-DQN은 성능평가에서 Reward-Sparse 환경에서 Q-Learning 알고리즘과 대비하여 약 3.5배의 높은 학습 성능을 보였으며, DQN과 대비해서 약 1.1배의 빠른 목표 달성 속도를 보였다. 또한, 우선적 경험재생의 구현과 보상획득 구간 세분화 정책의 구현을 통해 기존의 강화학습 모델들이 갖던 Positive-Bias 등의 문제점이 거의 발생하지 않았다. 하지만, 많은 수의 Actor를 병렬로 활용하는 아키텍처의 특성에 따라, 추후 경량화에 관한 추가적인 연구의 필요성이 제기된다. 본 논문은 추후 수행될 경량화 및 성능 개선 연구의 초석으로, 제안하는 아키텍처의 구조와 활용된 다양한 알고리즘의 내용 그리고 이를 구현하기 위한 명세를 상세히 기술한다.

Key Words : 강화학습, Reinforcement Learning, DQN

ABSTRACT

This paper considers various factors and conditions that correlate with the performance of the Reinforcement Learning, and defines the N-DQN model to improve them. N-DQN is a concept of applying and extending the architecture of HDQN, which layers multiple actors and has a structure in which operations are performed simultaneously/parallel through policy-based behavior management. Episodes acquired by each actor with action are stored in shared Replay Buffer, and various reinforcement learning enhancements such as Prioritized Experience Replay and segmentation of the reward acquisition period are applied to it. The implemented N-DQN showed about 3.5 times higher learning performance than the Q-Learning algorithm in the Reward-Sparse environment and about 1.1 times faster than DQN in attaining goal. Additionally, through the implementation of preferential experience regeneration and segmentation of reward acquisition period, problems of existing reinforcement learning models such as Positive-Bias have hardly occurred. However, due to architecture feature of

※ 본 연구는 2019년도 과학기술정보통신부의 재원으로 한국연구재단의 지원(No. 2017R1A2B4008860)과 정보통신기획평가원의 지원(No.2017-0-00279)을 받아 수행되었습니다.
◆ First Author : Konkuk University Department of IT Convergence Information Security, tackhyun12@konkuk.ac.kr, 학생회원
° Corresponding Author : Konkuk University Department of Computer Engineering, kckim@konkuk.ac.kr, 종신회원
* Konkuk University Department of Computer Information & Communications Engineering, llllssss94@konkuk.ac.kr
논문번호 : 201906-101-D-RN, Received June 6, 2019; Revised June 21, 2019; Accepted September 3, 2019

www.dbpia.co.kr

using lot of actors in parallel, working on improving performance through light-weightening in the future is needed. As a cornerstone of our future work for light-weightening and improving proposed architecture, this paper describe specific detail of proposed architecture structure, various algorithm used and ways to implement it.

Ⅰ. Introduction

강화학습(Reinforcement Learning)은 적용하고자 하는 환경(environment)을 상태와 행동 그리고 기대되는 보상으로 정의하여 미래에 기대되는 가치를 극대화하는 방향으로 자율적인 의사결정을 하는 알고리즘이다. 이러한 강화학습은 다른 지도학습 등의 기술과는 달리 데이터의 상태를 인식하고 의사결정을 수행하기 때문에 매 순간 최적의 정책을 선택해야 하는 문제를 해결하는 데 효과적이다. 또한, 복잡한 문제에 대하여 사전 지식의 필요 없이 스스로 환경과 상황을 판단하여 학습을 진행할 수 있다는 강점이 있다.[1]

하지만 이러한 강점을 갖는 강화학습이 모든 분야와 문제에 적용되어, 이를 해결할 수 있다는 의미는 아니다. 특정 분야와 산업에서는 예측할 수 없는 돌발적인 변수가 다양하게 발생하고, 예측 불가능한 경우의 수를 갖는 고차원의 데이터를 사용한다. 이와 같은 특정 환경에서는 강화학습의 명확한 한계점이 드러나고 있다.[2] 이러한 예시로, 강화학습에서 가장 범용적으로 사용되는 Q-Learning 알고리즘은 MDP(Markov Decision Process)[3] 이론을 바탕으로 해결하고자 하는 환경에서 학습을 수행하면서 행동의 Q-value를 평가하고 가장 높은 가치의 행동을 선택하는 의사결정 구조를 갖는다.[3] 이러한 의사결정 구조는 강화학습의 기본 철학과 가장 유사하지만, 실제 규칙성이 떨어지는 환경에 이러한 구조를 적용하게 되면 학습 능력이 현저히 떨어지는 문제점이 있는 것으로 밝혀졌다.[2]

이처럼 강화학습의 성능에 직접적인 영향을 미치는 여러 요인이 제시되면서, 이를 개선하기 위한 다양한 연구 및 시도가 진행되고 있다. 본 논문 또한 마찬가지로 강화학습의 성능 개선을 위한 연구를 목적으로, 성능에 영향을 끼치는 여러 요인 및 조건들을 중심으로 다양한 실험을 통해 이를 고찰하여 개선점을 통찰하고자 한다. 구체적으로는 기존의 Q-Learning, DQN(Deep Q-Networks) 알고리즘을 응용 및 개선한 N-DQN 모델을 제시하여, 활용된 기술과 성능 개선요인들을 설명한다. 또한, 이를 증명하기 위한 작업으로 N-DQN, Q-Learning, DQN을 각기 다른 특성을 가진 두 가지 환경에서 구현하여 성능평가를 수행하여, 그 결과를 비교/분석한다. 이러한 실험결과는 N-DQN 모델이 강화학습이 가진 한계점을 해결하는데 어떠한 가능성을 가지는지를 살피고자 한다.

Ⅱ. Related Works

2.1 Q-Learning

Q-Learning은 MDP[3]를 기반으로 하는 강화학습의 가장 대표적인 알고리즘이다. Q-Learning은 학습을 진행하면서 행동의 Q-value를 평가하고 가장 높은 가치의 행동을 선택한다. 수식(1)은 이러한 Q-Learning 알고리즘을 의미한다. 각 시간 t에서 Actor는 상태(S_t)에서 행동(A_t)을 수행하고 새로운 상태(S_{t+1})로 전이하는데 이러한 결과로 보상(R_t)을 획득하게 되는데, 이전의 값과 새 정보의 가중 합을 이용하여 반복함을 의미한다.[3]

$$Q(S_t,A_t)\leftarrow Q(S_t,A_t)+\alpha(R_{t+1}+\gamma\max_{a} Q(S_{t+1},a')-Q(S_t,A_t))' \quad (1)$$

하지만 Q-Learning은 각 행동을 평가할 때와 선택할 때 모두 같은 값을 활용하기 때문에, 현재 상황에서 가장 높은 가치로 평가된 행동의 미래 가치가 과대평가 되는 문제점이 있다. 즉, 과도하게 좋은 Q-value를 계속 선택하게 되어서 학습에 positive bias가 형성되고 학습이 진행될수록 악순환이 발생한다.[4] 이러한 현상은 딥러닝 방법을 도입하기 이전부터 일반적인 함수 추정에서도 발생하던 고질적인 문제이다.

Google의 DeepMind와 MIT 소속의 연구자들은 이러한 문제를 해결하기 위해 여러 강화학습 모델을 병렬처리하는 HDQN(hierarchical-DQN)[5]모델을 고안하였다. 이는 [Fig 1]과 같이, 학습을 top-level과 lower-level로 계층화하여 수행하는 구조를 의미한다.[5]

HDQN 모델에서 top-level과 lower-level은 각기 다른 목적성을 갖는다. lower-level은 최종적인 목표를 이루기 위한 것이 아니고, 현시점에서 유효한 고유 목적을 가진다. 반대로 top-level은 lower-level에서 생성한 결과를 취합하고 policy에 기반을 두어 학습을 수행한다.[5]

이러한 계층적 구조의 메커니즘은 복잡하고 불규칙

www.dbpia.co.kr

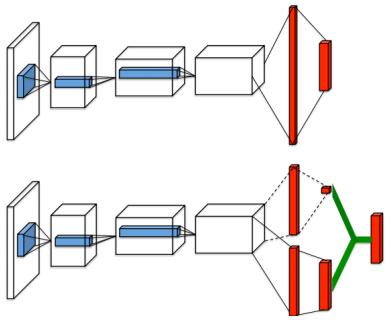

그림 1. HDQN의 구조
Fig. 1. Structure of HDQN

한 환경에서도 효율적인 학습의 가능성이 증가하는 것으로 밝혀졌다.[5] 따라서 본 논문은 이러한 HDQN의 강점을 더욱 활용하기 위해 계층구조를 추가로 확장하며, 여러 신경망을 계층화하고 정책을 기반으로 행동을 동시 병렬적으로 수행하는 N-DQN 모델을 정의하고자 한다. 또한, 이러한 모델을 미로찾기와 Ping-Pong 게임환경을 대상으로 구현하여 성능평가를 수행한다.

2.2 Find a Maze

미로찾기는 [Fig 2]와 같이 actor(●)를 미로의 출발점에서부터 도착지점까지 이동시키는 게임이다. actor는 기본적으로는 4가지(상/하/좌/우) 방향으로만 움직일 수 있으며, 벽은 통과할 수 없다는 직관적인 규칙을 갖는다. 이러한 과정에서 도착지점에 도달하기까지 소요된 시간 혹은 단계가 가장 적은 순으로 actor의 성능이 뛰어난 것으로 간주한다.

강화학습이 대두되어 연구되기 이전에도 미로찾기를 자동으로 수행하기 위한 다양한 연구가 있었다.[1] 일반적으로는 재귀함수 등에 휴리스틱(Heuristic)을 접목하는 방식으로 구현되었으며, 한 번에 목적지를 찾는 메커니즘을 구현하기보다는 반복 수행되는 경험(Experience) 속에서 최적의 경로와 가중치를 찾는 방식을 통해 문제를 해결하였다.[1]

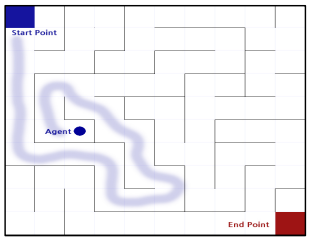

그림 2. 파이썬을 통해 구현된 미로 찾기
Fig. 2. Implemented Find a Maze with Python

표 1. 미로 찾기 기본 규칙
Table 1. Basic Rules of Find a Maze

No.	Rule
1	Actor는 상/하/좌/우 4방향으로만 움직일 수 있다
2	Actor는 벽을 통과할 수 없다
3	최단 시간 내 도착지에 도달해야 한다.

따라서 미로찾기에 강화학습을 적용함에서도 이러한 문제 해결방식을 응용한다. 즉, Actor의 반복 수행(Training) 중 발생하는 학습 데이터(Episode)에서 최적의 가중치를 찾아서 학습하는 것이다.

본 연구에서는 이러한 방향을 갖는 강화학습 모델을 구현하여 직관적인 실험을 수행하기 위해 미로찾기 환경을 Python을 통해 구현하였으며, 게임 내 적용되는 규칙은 [Table 1]로 정의한다.

2.3 Ping-Pong Game

Ping-Pong 게임은 [Fig 3]와 같이 actor paddle(▯)의 움직임을 통해 공(■)을 추적하여 쳐내고, 상대방의 paddle이 공을 받지 못하는 공간으로 보내는 게임이다. actor는 기본적으로 5가지(상/하/좌/우/無) 중 하나의 행동을 수행할 수 있으며, 움직임은 상/하 움직임으로 제한된다. 좌/우의 행동은 공이 paddle에 닿는 순간 발동할 수 있으며, 공을 상대의 방향으로 강하게 밀어낸다. 공은 타격 시 물리법칙을 통해 나아가며, 게임 화면의 상/하면에 닿으면 굴절된다. 따라서 actor는 이러한 물리법칙을 고려하면서 상대방이 받을 수 없는 위치로 공을 쳐 내는 것이 중요하다. [Table 2]는 Ping-Pong 게임의 규칙을 의미한다.

Ping-Pong 게임에 강화학습을 적용함에서는 actor paddle의 위치와 공의 위치 그리고 상대 paddle의 위치를 모두 고려해야 한다. 이처럼 다양한 객체가 움직이는 경우 객체의 움직임에 따라 기존에 수행한 학습 데이터의 가치가 낮아질 수 있다는 점이 예측된다. 이러한 방향성을 갖는 강화학습 모델을 구현 및 실험하

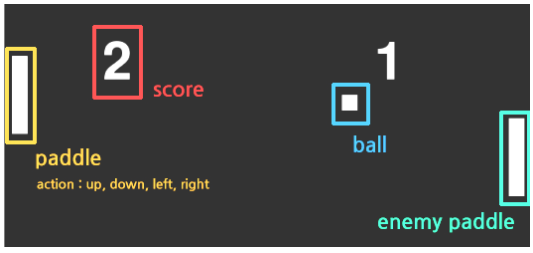

그림 3. 파이썬을 통해 구현된 Ping-Pong 게임
Fig. 3. Implemented Ping-Pong Game by Python

www.dbpia.co.kr

표 2. 핑퐁 게임 기본 규칙
Table 2. Basic Rules of Ping-Pong Game

No.	Rule
1	Actor는 상/하/좌/우/無 5가지 행동을 수행할 수 있다
2	상/하는 움직임과 관련되며, 좌/우는 공을 밀어낸다
3	공이 게임 화면의 상/하 면적에 닿으면 굴절된다

기 위해 Python을 통해 구현하였으며, 상대 Paddle은 휴리스틱이 접목된 Rule-Based 알고리즘에 의해 동작하도록 한다. 따라서, 성능평가의 내용은 강화학습과 Heuristic Rule-Based의 대결 구도를 의미한다.

Ⅲ. Proposed N-DQN

3.1 Proposed Architecture

N-DQN은 [Fig 4]와 같이 여러 개의 신경망(Sub Conv)을 동시 병렬적으로 활용하면서 일종의 HDQN과 같은 구조를 갖는다. 또한, 행동과 학습을 제어하는 정책을 수립하여, 여러 개의 신경망의 각각의 행동(action)을 통제한다. 행동에 따른 보상과 경험은 하나의 메모리에 공유되며 가장 좋은 데이터를 선정하여 우선하여 학습한다.

3.1.1 Hierarchization and Policy

N-DQN은 크게 main 계층과 sub 계층으로 구분되어 정의된다. 먼저, sub 계층은 행동을 실제로 수행하고 보상과 경험을 획득하여 메모리에 저장하는 actor로 정의하며, main 계층은 학습 데이터(Episode) 중에서 더 좋은 데이터를 추출하고, 실제 학습을 관장하면서 정책을 통해 sub 계층의 행동을 통제한다. [Fig 4]의 예시를 살펴보면, 4개의 sub 계층(Sub Conv)은 각각 4가지 종류의 행동을 수행할 수 있다. 이때, sub 계층은 main 계층(Main Conv)이 수립한 정책에 의해서 서로 중복되지 않으면서도, 수행한 적이 없는 행동을 우선 수행하도록 구현한다. 이러한 구현 정책은 기존의 강화학습 알고리즘과 대비하여 효율적으로 Episode를 생성하여 학습 효율을 높이기 위함이며, 이는 [Algorithm 1]과 같이 구현할 수 있다.

[Algorithm 1]은 main 계층(Main Conv)에 의해 만들어진 히스토리 큐 H를 통해 각 sub 계층(Sub Conv)이 중복되지 않는 행동을 취하게 되는 과정을 의미한다. 히스토리 큐 H는 기존 replay buffer에 저장되었던 모든 Episode에 대한 state와 action의 기록을 저장하고 있다. 따라서, 각 sub 계층은 히스토리 큐 H를 확인함으로써 현재 자신의 target 행동의 중복성을 검증할 수 있다.

기존의 Q-Learning, DQN 알고리즘은 수행하지 않은 학습 데이터를 획득하기 위해 무작위적인 행동을 수행하게 된다.[3] 이는, 난수를 생성하여 threshold보다 낮으면 행동을 수행하고 threshold의 값을 점차 낮추어가는 과정을 반복 수행하는 과정(e-greedy)으로 구현된다. 이러한 절차는 무작위 확률에 의존하기 때문에 명확성과 효율성이 떨어진다.[6] 하지만 N-DQN은 [Algorithm 1]을 통해 무작위적인 확률에만 의존하

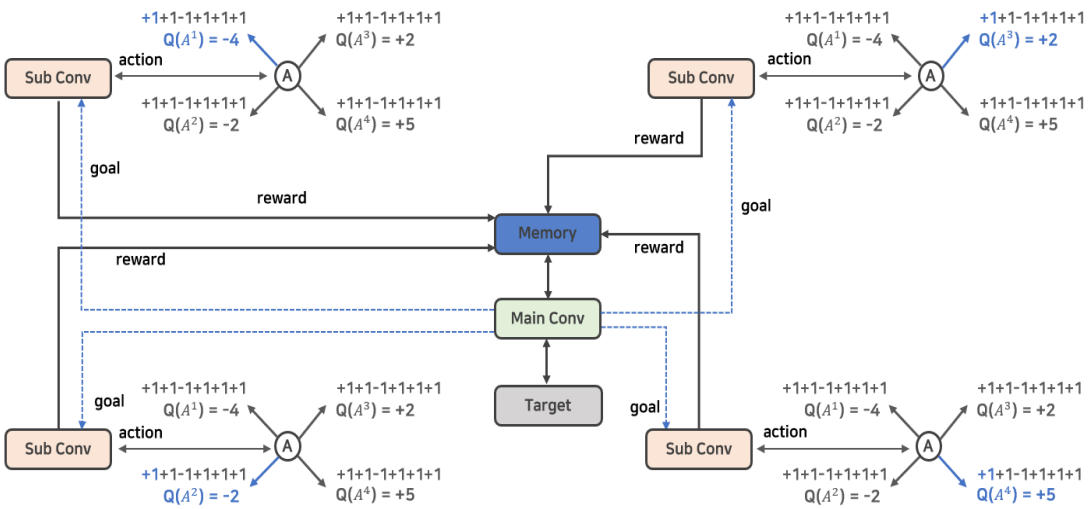

그림 4. 제안하는 N-DQN의 아키텍처
Fig. 4. Proposed Architecture of N-DQN

www.dbpia.co.kr

Algorithm 1 : Action Control Policy

```
1. Input: Number of Sub Layer N, Current State s, target action a
2. Initialize: shared replay memory D to size N * n
3. Initialize: N action-value function Q with random weights
4. Initialize: episode history queue H to size N * n
   Copy Data to H from D
      For 1, N do
         For state in H do
            if state = s then
               For action in state.action do
5.                if action != a then
                     do action
                  else
                     break
            else
               do action
```

지 않고, 검증을 수행하며 정책을 통해 학습 데이터 생성에 적극적으로 개입하므로 학습의 효율성을 높일 수 있다.

main 계층은 보상(reward)과 관련된 정책 또한 생성하고 수행하여 학습의 효율성을 강화한다. [Fig 4]에서 main 계층은 sub 계층에게 목표(goal)를 제시하는 것을 살펴볼 수 있다. 이는 강화학습의 성능에 직접적인 영향을 미치는 요인 중 하나가 보상의 발생빈도에서 비롯된다는 사실[7]에서 기인한 결과이다.

강화학습은 행동에 따른 보상의 발생빈도와 확률에 따라 학습 난이도가 유의하게 달라지는 특징이 있다.[7] 보상이 자주 발생하는 환경이란, 학습되지 않은 행동을 통해서도 목표를 달성할 확률이 높은 환경을 의미하며, 반대로 보상이 자주 발생하지 않는 환경은 불규칙하고 변칙적인 환경에서 정의된 보상의 획득 확률이 적은 것을 의미한다. 이처럼 보상이 적게 발생하는 환경에서는 학습을 수행하는 데 많은 문제가 발생한다.

N-DQN은 이러한 문제를 해결하기 위해 보상획득 구간을 나누어 세분화하는 정책을 수립한다. 구체적으로는 학습을 일정 기간(time, step) 수행하고, 보상의 획득이 적게 발생하는 환경에서는 행동과 보상 그리고 상태 등의 정보가 모두 포함된 Episode를 기반으로 보상획득의 구간을 세분화시켜 점진적인 학습이 가능토록 하는 원리이다. 이러한 일련의 과정은 [Algorithm 2]와 같다.

[Algorithm 2]는 세분된 보상획득 구간을 확인하여

Algorithm 2 : Reward Policy

```
1. Input: Current State s, immediate reward r
2. Initialize: List of lethal state L, Dict of reward P
3. Initialize: the number of reward point K
   After do action
   Before return immediate reward r
   For k in range(0, K)
4.    if s = L[k] then
         return r += P[L[k]]
      else
         return r
```

immediate reward가 주어지는 과정을 의미한다. 미리 정의된 보상획득 구간에 대한 정보를 리스트의 형태로 가지고 있으며, 각 구간에 해당하는 reward 값 또한 유지한다. 이 값들을 활용하여 매 step에서 임의의 action을 취한 후 그에 대한 reward를 계산하면서, 최종적으로 보상획득에 해당하는지 확인 후 immediate reward에 추가적인 reward를 부여하게 된다.

3.1.2 Parallel Processing and Memory Sharing

행동을 직접 수행하는 주체인 actor(sub 계층)는 동시 병렬적으로 행동을 수행하고 그 결과(Episode)의 정보를 하나의 Replay Buffer에 공유하도록 구현한다. 이러한 병렬처리 및 공유구조를 구현하기 위해서 Apache Spark를 활용한 Parallel Processing 구조의 선행연구를 참조한다.[8-10]

[Fig 5]는 N-DQN에 적용된 Parallel Processing 구조를 의미한다. 그림에 활용된 예시는 Apache Spark 라이브러리를 통해 2개의 Actor(Sub Conv)를 병렬처리하게 되는 구조를 의미하는데, 이는 실제로 단일 처리보다 성능이 더욱 향상된다는 연구결과에 근거한다.[9]

Actor가 병렬처리를 수행하여 획득한 각각의 Episode 결과는 Replay Buffer에 공유성 있게 저장한

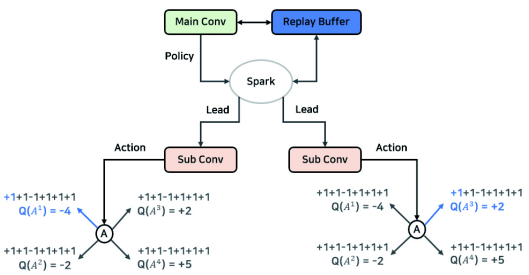

그림 5. Spark를 통한 Parallel Processing 구조
Fig. 5. Structure of Parallel Processing through Spark

www.dbpia.co.kr

다. 이로 인해, 버퍼에 데이터가 누적되는 속도가 Actor의 개수에 비례하여 향상되는 효과가 있다. 이후, Main 계층은 빠르게 누적된 Replay Buffer에서 효율적으로 데이터를 선정하여 학습하게 된다.

[Algorithm 3]은 N-DQN의 특성 중 하나인 경험의 공유구조를 구현하기 위한 명세이다. 각각의 actor (Q_n)는 n개의 action-value function Q의 값을 구한 후 공유된 Replay Buffer에 결과를 저장한다. 이처럼 여러 개의 Actor에 의해 공유성 있게 수집된 학습 데이터는 일반적인 강화학습 알고리즘보다 더욱 빠르게 데이터를 모을 수 있다는 강점을 이용하는 원리이다.

Algorithm 3 : N-DQN Experience Sharing

Procedure: Training
Initialize: shared replay memory D to size N * n
Initialize: N action-value function Q with random weights
Loop:
episode = 1, M do
Initialize state s_1
For t = 1, T do
For each Q_n do
With probability ϵ select random action a_t
otherwise select a_t = argmax$_a$ $Q_n(s_t, a; \Theta_i)$
Execute action a_t in emulator and observe r_t and s_{t+1}
Store transition (s_t, a_t, r_t, s_{t+1}) in D
End For
Sample a minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
Set y_j :=
r_j **For** terminal s_{j+1}
r_j + γmax_(a^') $Q_n(s_{j+1}, a'; \Theta_i)$ **For** non-terminal s_{j+1}
Perform a gradient step on (y_j - $Q_n(s_j, a_j; \Theta_i)$)^2 with respect to Θ
end For
End Loop

3.1.3 Training Policy

강화학습 알고리즘은 보편적으로 데이터 간의 의존성을 제거하기 위해 데이터를 Replay Buffer에 저장 후 무작위로 선정하여 학습을 수행한다.[6]

하지만 Replay Buffer에 저장된 모든 데이터가 같은 가치를 가지는 것은 아니다. 환경에 따라 예외가 발생할 수는 있으나, 일반적으로는 특정 경험이 더 중요한 가치를 가지게 된다.[11]

이러한 특성에 따라 학습 데이터의 우선순위를 선정하거나 중요도를 구분하는 다양한 연구가 진행되었고, 그중 가장 대표적인 기술이 우선적 경험재생(Prioritized Experience Replay)[11]이다. N-DQN은 아키텍처의 특성에 따라 많은 actor를 활용하기 때문에 경험의 누적속도가 일반적인 DQN 보다 빠르다. 따라서, 본 아키텍처를 기준으로 우선적 경험재생을 구현하여 경험의 가치를 판별하는 메커니즘은 학습 성능에 큰 영향을 주는 요소이다.

경험의 중요성은 해당 경험을 통해서 현재 상태에서 학습할 수 있는 양에 비례한다. 이러한 정량적인 수치는 TD-Error로 정의되는 TD Target과 실제 V(S)와의 차이를 통해 구할 수 있다. 이는 보편적으로 수식(2)과 같이 정의된다.[12]

$$\delta_j = R_j + \gamma_j Q_{target}(S_j, \max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1}) \quad (2)$$

수식(2)을 통해 구해진 정량적인 수치는 [Algorithm 4]와 같이 TD-Error의 크기를 기준으로 확률분포를 만들고 그 분포에서 값을 추출하는 방식

Algorithm 4 : Prioritized Experience Replay

Input: minibatch k, step-size n, replay period K and size N, exponents α and β, budget T and N action-value function Q
Initialize replay memory $H = \theta, \triangle = 0, p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for t = 1 **to** T **do**
Observe S_t, R_t, γ_t
for $p = 1$ to N **do**
Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ in H with maximal priority $p_t = \max_{i<t} p_i$
end for
if $t \equiv 1 \bmod K$ **then**
for $j=1$ **to** k **do**
Sample transition $j \sim P(j) = p_j^\alpha / \sum_i p_i^\alpha$
Compute sampling weight $w_j = (N \cdot P(j))^{-\beta} / \max_i w_i$
Compute TD-error
$\delta_j = R_j + \gamma_j Q_{target}(S_j, \arg\max_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
Update transition priority $p_j \leftarrow |\delta_j|$
Gather weight-change $\triangle \leftarrow \triangle + w_j \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
end for
Update weights $\theta \leftarrow \theta + n \cdot \triangle, reset \triangle = 0$
From time to time copy weights into target NN $\theta_{target} \leftarrow \theta$
end if
for $p = 1$ to N **do**
Choose action $A_t \sim \pi_\theta(S_t)$
end for
end for

www.dbpia.co.kr

으로 구현한다.[14] 단순히 TD-Error 값의 크기를 기준으로 정렬하여 활용할 수도 있으나, 확률분포를 사용하는 것이 더 좋은 결과를 갖는다는 연구결과를 근거로 한다.[4,12-14]

Ⅳ. Evaluation Results and Analysis

4.1 Environment

[Fig 6]은 미로찾기 환경을 대상으로 강화학습을 수행하는 구조를 의미한다. 각 State별 Action이 수행되는 원리를 도식화하였다.

4.2 Training Features

State는 게임 내에서 Actor의 현재 위치 좌표(x/y)로 정의한다. Actor는 행동에 따라 위치가 실시간으로 변동되기 때문에 한 번 행동이 수행될 때마다 다시 정보를 획득하도록 구현한다.

Action은 Actor의 직접적인 움직임으로 정의된다. Actor는 (상/하/좌/우) 중 하나의 행동을 취할 수 있으며, 막혀있는 벽은 통과할 수 없다.

Reward는 State의 위치 정보가 셀의 숫자에 비례하여 목표에 얼마나 가까워졌는지 확인하는 것으로 정의한다. 또한, 목표에 도달 시에는 높은 가중치의 보상을 획득하도록 구현한다.

표 3. 미로 게임 학습 대상
Table 3. Training Features

Category	Contents
State	Actor의 위치 좌표(x/y)
Action	(상/하/좌/우) 중 1개의 행동
Rewards	Goal에 도착 시 + 1 Step 이동 시마다 $-\frac{0.1}{Number\ of\ cell}$

4.3 Implement and Performance Evaluation

4.3.1 N-DQN Based Implementation

N-DQN을 통한 미로찾기 환경에서의 강화학습 구현 간에는 4개의 sub 계층과 1개의 main 계층을 활용하였다. 학습 알고리즘은 DQN을 기반으로 하였으며, 3장에 서술된 아키텍처의 구조와 기능들을 포함한다. [Fig 7]은 미로찾기에 적용된 Reward Policy를 의미한다. 학습 및 탐색을 일정 기간 수행하면서, 각 State에서 소요된 시간과 목표에 도달하기까지의 Step 수를 기반으로 보상획득 구간을 세분화하는 정책을 수립한다.

보상에는 게임이 완전히 종료될 때까지의 보상의 합계를 구하여, 각각의 단계마다 y의 인수로 discount하는 수식(3)을 추가로 구현하여 가중부여한다. 이러한 과정과 더불어 State를 기준으로 Reward가 급격히 나빠지는 구간을 식별한다. 이처럼 식별된 구간은 가중치에 Negative 하게 보상을 부여한다.

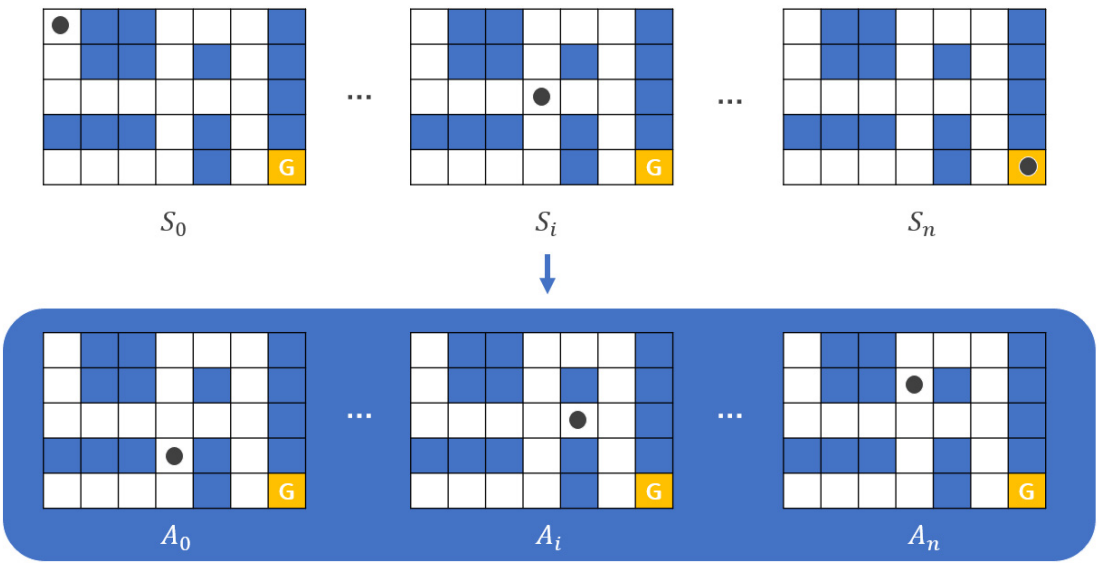

그림 6. 미로찾기의 강화학습을 시각화한 그림
Fig. 6. Visualization of RL in Find a Maze

www.dbpia.co.kr

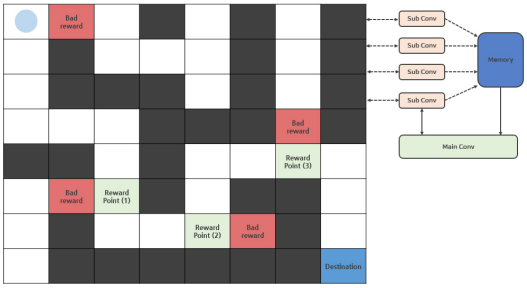

그림 7. N-DQN의 보상획득 정책
Fig. 7. N-DQN's Reward Acquisition Policy

$$R_t = \sum_{t'=1}^{T} Y^{t'-t} r_t \tag{3}$$

4.3.2 Performance Evaluation and Discussion

본 절에서는 제안된 N-DQN 모델의 성능을 평가하기 위해 동등한 환경과 조건에서 Q-Learning과 DQN 알고리즘에 강화학습을 적용한 결과와 비교 및 고찰하여 개선점을 통찰하고자 한다.

[Fig 8]은 5x5 크기의 미로 환경에 Q-Learning을 적용한 결과이다. 결론적으로 Q-Learning은 목표한 400단계 안에서 문제를 해결하였으며, 50단계도 걸리지 않아 좋은 학습 효율을 보였다. 작은 크기의 미로에서는 학습되지 않은 행위에 의해서도 목표를 달성할 가능성이 크다. 이로 인해, Q-table을 통한 단순한 저장 방식이 오히려 Q-network를 사용하는 다른 모델에 비해 높은 효율성을 보인다.

[Fig 9]는 5x5 크기의 미로 환경에 DQN을 적용한 결과이다. 결론적으로 DQN 알고리즘은 목표한 400단계 내에서 문제를 해결하는 것에 실패하였다. 구현 간에 발생한 논리 오류일 가능성 확인하기 위해 목표 단계의 수를 기존보다 25%(100단계)를 추가로 수행한 결과에서는 목표 해결에 성공하였다.

이러한 실험결과는 실패의 원인이 Replay Buffer에서 학습 표본을 선정하는 과정이 무작위성을 가지고 있어, 학습의 효율성이 좋지 못함을 시사한다.[6] 학습 효율을 개선하는 알고리즘의 추가적인 구현이나 Replay Buffer에서 가장 오래된 데이터를 차례로 제거하여도 성능이 크게 개선된다.

[Fig 10]은 5x5 크기의 미로 환경에 N-DQN을 적용한 결과이다. 결론적으로 N-DQN은 목표한 400단계 안에서 문제를 해결하였으며, 약 100단계 이후부터 안정적인 성능을 보인다. DQN과는 달리 다양한 성능 개선요소들이 구현되어 있으므로 학습 효율이 월등히 뛰어다. 하지만, sub 계층의 병렬화에서 발생하는 높은 H/W나 자원 점유율을 보이는 문제점이 식별된다.

[Table 4]는 실험결과의 요약을 의미한다.

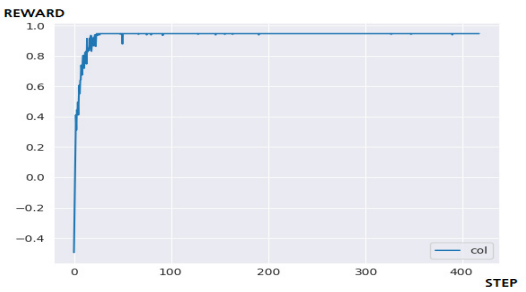

그림 8. Q-Learning을 통한 5x5 미로찾기의 결과
Fig. 8. Result of 5x5 Maze through Q-Learning

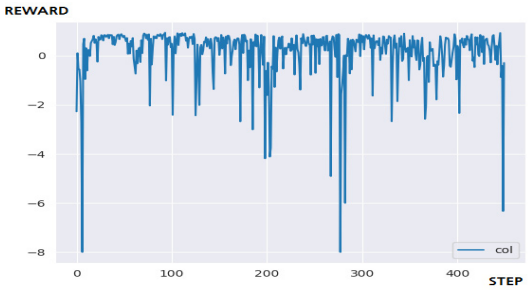

그림 9. DQN을 통한 5x5 미로찾기의 결과
Fig. 9. Result of 5x5 Maze through DQN

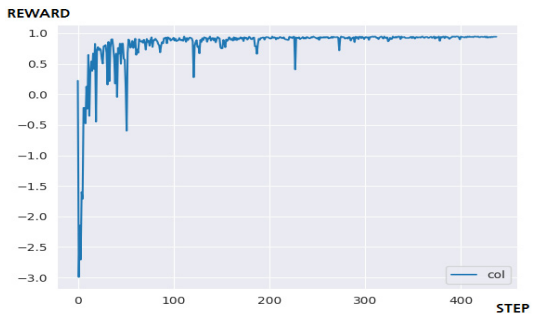

그림 10. N-DQN을 통한 5x5 미로찾기의 결과
Fig. 10. Result of 5x5 Maze through N-DQN

표 4. 5x5 학습 결과 요약
Table 4. Results of Summary

Model	Goal	Need step	Time required (400 step)
Q-Learning	Success	50 step	46.7918 second
DQN	Fail	500 step	57.8764 second
N-DQN	Success	100 step	66.1712 second

www.dbpia.co.kr

Q-Learning이 가장 적은 단계와 시간으로 문제를 해결하였다. 5X5 크기의 미로는 Episode의 수가 많지 않으며, 학습되지 않은 행동에서도 보상을 획득할 확률이 높다. 즉, 학습의 난이도가 결론을 내기에는 너무 쉽다고 판단되어 미로의 크기를 더욱 확장하여 추가 실험을 진행하였다.

[Fig 11]은 10x10 크기의 미로 환경에 Q-Learning을 적용한 결과이다. Q-Learning은 목표한 700단계 안에서 문제를 해결하는 것에 실패하였다. 하지만 학습 단계의 수를 늘리면 충분히 목표치에 도달 가능할 정도의 상승 곡선을 보였다. 실제로 목표치를 1.7배 늘려서 약 1,200단계의 학습을 수행한 결과, 목표를 달성하는 것을 확인하였다.

[Fig 12]는 10x10 크기의 미로 환경에 DQN을 적용한 결과이다. DQN은 5x5 크기의 미로 때와 마찬가지로 목표한 단계 안에서 문제를 해결하는 데 실패하였다. 이는 여러 번의 실험에서 가끔 성능곡선이 좋게 나올 때가 있었지만, 그것은 운에 의한 요소로 판단된다. 학습 단계의 수를 2배로 늘린 1,400단계의 학습을 수행하였을 때부터 학습 효율이 나타나기 시작하였다.

성능저하의 원인을 고찰하기 위해 렌더링을 통해 actor의 움직임을 관찰한 결과, 미로의 잘못된 구간에 진입하였을 때 빠져나오지 못하고 비효율적인 action을 수행하는 것이 확인되었다. 따라서, Episode에 대량의 state와 action이 발생하여 학습의 성능이 크게 저하된다.

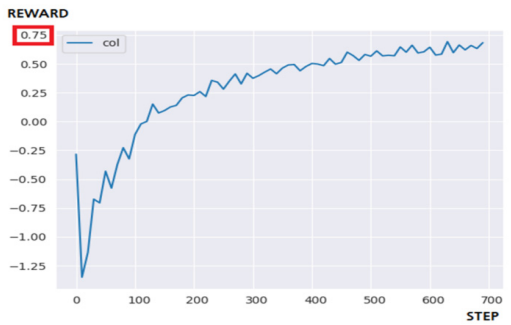

그림 11. Q-Learning을 통한 10x10 미로찾기의 결과
Fig. 11. Result of 10x10 Maze through Q-Learning

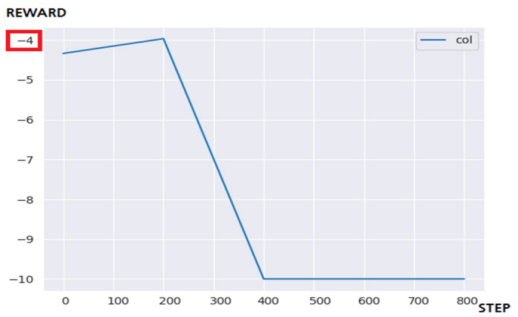

그림 12. DQN을 통한 10x10 미로찾기의 결과
Fig. 12. Result of 10x10 Maze through DQN

[Fig 13]은 10x10 크기의 미로 환경에 N-DQN을 적용한 결과이다. N-DQN은 목표한 700단계 안에서 문제해결에 성공하였으며, 꾸준하며 안정적인 학습 효율성을 보였다. 약 400단계가 지난 시점에서 이미 일정 수준 이상의 학습이 완료된다. 이러한 결과는 학습과 행동을 통제하는 정책에서 가장 큰 영향을 받는다.[15]

우선적 경험재생의 구현과 보상획득 구간 세분화 정책에 의해 기존의 강화학습이 가지던 Positive-Bias 문제점이 거의 발생하지 않았으며, Reward-Sparse 환경에서의 학습 효율저하 문제점 또한 개선된 것으로 추정된다. 하지만 이전과 마찬가지로 많은 수의 actor를 병렬로 활용하는 아키텍처 구조의 특성상 매우 높은 하드웨어 자원의 점유율을 보인다. 이는 게임 화면의 시각화를 담당하는 렌더링을 끔으로써 크게 개선시킬 수 있으나, 더 근본적인 개선책의 필요성을 제기한다.

[Table 5]는 실험결과의 요약을 의미한다. N-DQN만이 목표한 단계 내에서 문제를 해결하였으며, DQN과 대비해서 약 10% 더 빠른 수행 속도를 보였다. 학

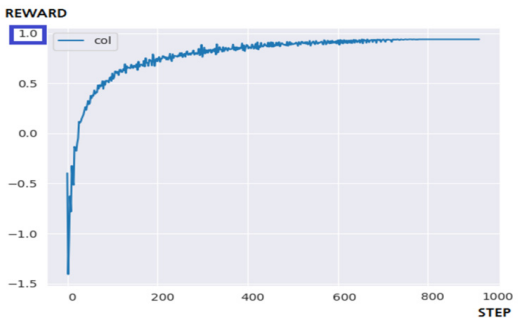

그림 13. N-DQN을 통한 10x10 미로찾기의 결과
Fig. 13. Result of 10x10 Maze through N-DQN

표 5. 10x10 학습 결과 요약
Table 5. Results of Summary

Model	Goal	Need step	Time required (700 step)
Q-Learning	Fail	1,200 step	379.1281 second
DQN	Fail	- step	427.3794 second
N-DQN	Success	700 step	395.9281 second

www.dbpia.co.kr

습의 효율성 측면에서도 Q-Learning이 700단계를 통해 얻은 0.75라는 Reward 수치를 N-DQN은 약 200단계로 획득하였다. 이는, 약 3.5배의 성능 차이를 의미한다. 이처럼 Reward가 Sparse 한 경우에도 N-DQN 모델은 안정적인 학습 성능을 보였다.

V. Ping-Pong with RL

5.1 Environment

[Fig 14]는 Ping-Pong 환경을 대상으로 강화학습을 수행하는 구조를 의미한다.

5.2 Training Features

State는 게임 내에서 Actor 패들의 현재 위치 좌표와 볼의 현재 위치 좌표로 정의한다. Actor 패들과 볼은 행동에 따라 위치가 실시간으로 변동되므로 행동이 수행될 때마다 다시 정보를 획득하도록 구현한다.

Action은 Actor 패들의 직접적인 움직임으로 정의된다. Actor는 (상/하/좌/우) 중 하나의 행동을 취한다. Rewards는 상대방의 패들이 공을 받아내지 못한다면 플레이어가 라운드에서 승리하기 때문에 높은 가중치의 보상을 부여한다. 반면, actor의 패들이 공을 놓치게 되면 플레이가 라운드에서 패배하기 때문에 Negative 하게 보상을 부여한다.

표 6. 핑퐁 게임 학습 대상
Table 6. Training Features

Category	Contents
State	패들의 현재 위치의 좌표(x/y) 볼의 현재 위치의 좌표(x/y)
Action	(상/하/좌/우) 중 1개의 행동
Rewards	공을 상대방의 패들이 놓치면 +1 공을 actor의 패들이 놓치면 -1 그렇지 않으면 0

5.3 Implement and performance evaluation

5.3.1 N-DQN Based Implementation

N-DQN을 통한 Ping-Pong 게임환경에서의 강화학습 구현 간에는 4개의 sub 계층과 1개의 main 계층을 활용하였다. [Fig 14]는 이러한 계층구조 및 아키텍처를 의미한다. 학습 알고리즘은 DQN을 기반으로 하였으며, 3장에 서술된 아키텍처의 구조와 기능들을 포함한다.

강화학습의 사전준비 과정으로 게임을 프레임 단위로 나누어 그레이 스케일로[16] 처리한다. 이러한 과정이 인식률에 영향을 준다는 연구결과를 근거로 한다.[16,17] 이후, 이미지 프레임을 80x80의 크기로 조정한 뒤 4개의 프레임을 쌓아 80x80x4의 배열을 구성한다. 신경망은 [Fig 15]와 같이 총 3개의 layer의 구조를 가지며, 마지막 hidden layer에서는 완전히 연결된 256개의 ReLU로 구성하였다.

학습 알고리즘의 구현 간에는 DQN 알고리즘을 기반으로 가장 보편적인 구조를 활용한다. 아래의 수식 (4)과 같이 Q-value로 정의된 함수를 통해 주어진 state에서 행동을 수행할 때 예상되는 최대의 값을 근사치로 구하여 활용한다.[6]

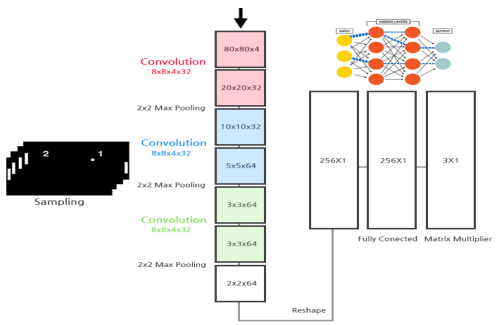

그림 15. 구현된 인공 신경망의 구조
Fig. 15. Implemented Neural Network Structure

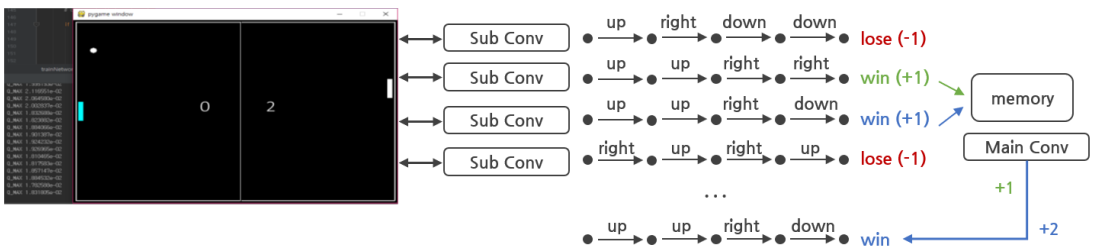

그림 14. 강화학습을 통한 Ping-Pong 게임의 시각화
Fig. 14. Visualization of RL in Ping-Pong Game

www.dbpia.co.kr

$$Q^*(s,a) = E[R_t|s_t = s, a_t = a] \quad (4)$$

수식(3)으로 정의된 학습구조는 [Algorithm 5]와 같은 방식으로 구현할 수 있다. 해당 알고리즘의 코드는 DQN을 기반으로 작성되었다.

공과 상대 패들이 유기적으로 움직임에 따라, actor 패들의 각 행동 a_t 가 가져올 미래의 가치 $Q(a_t)$에 대한 가치가 낮아질 수 있다. 이러한 문제점을 해결하기 위해서는 객체를 개별화하는 centralized critic 정책의 구현이 필요함을 시사한다.[18,19]

학습의 효율을 최대한 높이기 위해 보상획득 구조의 세분화 또한 적용한다. 학습 및 탐색을 일정 기간 수행하고, 각 State에서 소요된 시간과 목표에 도달하기까지의 Step 수를 기반으로 보상획득 구간을 세분화하는 정책을 수립한다.

보상에는 게임이 완전히 종료될 때까지의 보상의 합계를 구하여, 각각의 단계마다 y의 인수로 discount 하는 수식(5)을 추가로 구현하여 가중부여한다. 이는 보상획득 구조의 세분화와 상통하는 의미로 최단기간에 목표를 수행하도록 하기 위한 목적성을 갖는다.

$$R_t = \sum_{t'=1}^{T} Y^{t'-t} r_t \quad (5)$$

Algorithm 5 : Ping-Pong Game's DQN Training

1. **Procedure:** Training
2. **Initialize:** replay memory D to size N
3. **Initialize:** action-value function Q with random weights
4. **Loop:**
episode = 1, M do
Initialize state s_1
for t = 1, T do
With probability ϵ select random action a_t
otherwise select a_t=argmax_a Q(s_t,a; Θ_i)
Execute action a_t in emulator and observe r_t and s_(t+1)
Store transition (s_t,a_t,r_t,s_(t+1)) in D
Sample a minibatch of transitions (s_j,a_j,r_j,s_(j+1)) from D
Set y_j:=
r_j for terminal s_(j+1)
r_j+γ*max_(a') Q(s_(j+1),a'; Θ_i) for non-terminal s_(j+1)
Perform a gradient step on (y_j-Q(s_j,a_j; Θ_i))^2 with respect to Θ
end for
5. **End Loop**

5.3.2 Performance Evaluation and Discussion

본 절에서는 제안된 N-DQN 아키텍처의 성능을 평가하기 위해 동등한 환경과 조건에서 DQN 알고리즘에 강화학습을 적용한 결과와 비교 및 고찰하여 개선점을 통찰하고자 한다.

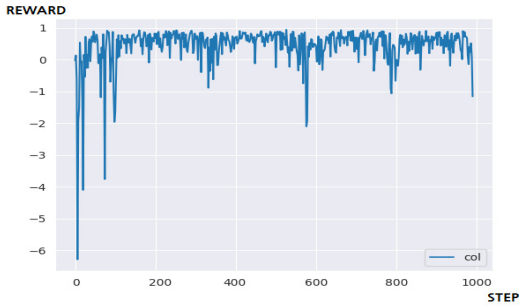

그림 16. DQN을 통한 Ping-Pong 게임의 결과
Fig. 16. Result of Ping-Pong Game through DQN

[Fig 16]은 Ping-Pong 게임환경에 DQN 알고리즘을 적용한 결과이다. 결론적으로 많은 훈련단계를 통해 일정 수준의 성능을 보장하는 강화학습의 적용에는 성공하였으나, 성능이 나빠지는 구간이 종종 발생한다. 새로운 Episode를 탐색하고 만드는 과정을 무작위성에 의존하기 때문에 비효율적인 성능을 보인다.

[Fig 17]은 Ping-Pong 게임환경에 N-DQN을 적용한 결과이다. 결론적으로 일정 수준을 보장하는 강화학습의 적용에는 성공하였고, DQN 알고리즘의 수행결과와 대비하였을 때, 새로운 Episode 패턴에 더욱 유연하게 대처한다. 또한, positive-bias 문제점이 거의 발생하지 않는다. 하지만 예상치보다 학습에 필요한 단계의 수를 크게 줄이지는 못했으며, 성능이 나빠지는 구간을 완전히 배제하는 데는 실패하였다. 추가적인 성능 개선을 위해서는 보상 구조에 휴리스틱을 접목하는 것이 방법이 될 수 있다.

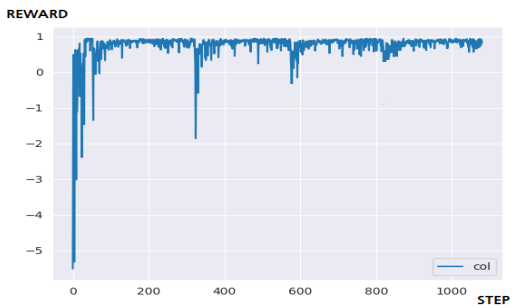

그림 17. N-DQN을 통한 Ping-Pong 게임의 결과
Fig. 17. Result of Ping-Pong Game through N-DQN

www.dbpia.co.kr

Ⅵ. Conclusion

본 논문은 강화학습의 성능에 영향을 끼치는 여러 요인 및 조건들을 중심으로, 이를 개선하기 위한 N-DQN 모델을 구현하였다. N-DQN은 HDQN의 구조를 응용 및 확장하여 여러 신경망을 계층화하고 정책을 기반으로 행동을 동시 병렬적으로 수행하는 구조를 갖는다. 또한, 성능을 개선하기 위해 우선적 경험재생, 보상 정책의 세분화 등 여러 추가적인 아이디어와 기술들이 접목되었다. 이처럼 많은 성능 개선의 요인을 갖는 N-DQN은 실제 성능평가에서 일반적인 강화학습 알고리즘들과 같은 환경에서 더 좋은 결과를 보였다.

N-DQN은 Reward-Sparse 환경에서 Q-Learning 알고리즘과 대비하여 약 3.5배의 높은 학습 성능을 보이며, DQN과 대비해서 약 1.1배의 빠른 목표 달성 속도를 보였다. 또한, 우선적 경험재생의 구현과 보상획득 구간 세분화 정책의 구현을 통해 기존의 강화학습이 갖던 Positive-Bias가 거의 발생하지 않았으며, 여러 학습 효율저하 문제점 또한 개선되었다. 하지만 N-DQN은 여러 개의 actor를 Parallel Processing 하는 구조의 특성상 일반적인 강화학습 모델보다 높은 H/W 점유율을 보인다. 이러한 특징에 따라, 추후 Parallel Processing과 신경망 구조의 경량화에 관한 추가적인 연구의 필요성이 제기된다.

References

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artificial Intell. Res.*, vol. 4, pp. 237-285, 1996.

[2] J. A. Boyan and A. W. Moore, "Generalization in reinforcement learning: Safely approximating the value function," in *Advances in Neural Inf. Process. Syst.*, pp. 369-376, 1995.

[3] C. J. Watkins and P. Dayan, "*Q-learning. Machine learning*," vol. 8, no. 3-4, pp. 279-292, 1992.

[4] H. V. Hasselt, "Double Q-learning," in *Advances in Neural Inf. Process. Syst.*, pp. 2613-2621, 2010.

[5] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in Neural Inf. Process. Syst.*, pp. 3675-3683, 2016.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[7] R. S. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding," in *Advances in Neural Inf. Process. Syst.*, pp. 1038-1044, 1996.

[8] Maxence Queyrel, *Multithreading to Construct Neural Networks*(2018), Retrieved Jan. 14, 2019, from https://www.slideshare.net/altoros/multithreading-to-construct-neural-networks

[9] N. Purkaitl, *How to train your Neural Networks in parallel with Keras and Apache Spark*(2018), Retrieved Jan. 15, 2019, from https://towardsdatascience.com/how-to-train-your-neural-networks-in-parallel-with-keras-and-apache-spark-ea8a3f48cae6

[10] lifeomic, *Sparkflow*(2018), Retrieved Jan. 15, 2019, from https://github.com/lifeomic/sparkflow

[11] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas, "Dueling network architectures for deep reinforcement learning," arXiv preprint arXiv:1511.06581, 2015.

[12] G. Tesauro, "Temporal difference learning and TD-Gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58-68, 1995.

[13] J. P. O'Doherty, P. Dayan, K. Friston, H. Critchley, and R. J. Dolan, "Temporal difference models and reward-related learning in the human brain," *Neuron*, vol. 38, no. 2, 329-337, 2003.

[14] S. Lyu, *Prioritized Experience Replay*(2018), Retrieved Jan. 11, 2019, from https://lyusungwon.github.io/reinforcement-learning/2018/03/20/preplay.html

[15] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for

www.dbpia.co.kr

reinforcement learning with function approximation," in *Advances in Neural Inf. Process. Syst.*, pp. 1057-1063, 2000.

[16] C. Saravanan, "Color image to grayscale image conversion," *ICCEA 2010*, vol. 2, Bali Island, Indonesia, 2010.

[17] C. Kanan and G. W. Cottrell, "Color-to-Grayscale: Does the method matter in image recognition?," doi:10.1371/journal.pone.0029740, 2011.

[18] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Thirty-Second AAAI Conf. Artificial Intell.*, Apr. 2018.

[19] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Advances in Neural Inf. Process. Syst.*, pp. 6379-6390, 2017.

[20] A. D. Tijsma, M. M. Drugan, and M. A. Wiering, "Comparing exploration strategies for Q-learning in random stochastic mazes," in *2016 IEEE SSCI*, pp. 1-8, Dec. 2016.

[21] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI Conf. Artificial Intell.*, Mar. 2016.

[22] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *Int. Conf. Mach. Learning*, pp. 1928-1937, Jun. 2016.

[23] I. Osband, C. Blundell, A. Pritzel, & B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Inf. Process. Syst.*, pp. 4026-4034, 2016.

[24] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," in *Advances in Neural Inf. Process. Syst.*, pp. 2863-2871, 2015.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Inf. Process. Syst.*, pp. 1097-1105, 2012.

[26] M. A. Wiering and H. Van Hasselt, "Ensemble algorithms in reinforcement learning," *IEEE Trans. Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 930-936, 2008.

[27] E. T. Rolls, C. McCabe, and J. Redoute, "Expected value, reward outcome, and temporal difference error representations in a probabilistic decision task," *Cerebral cortex*, vol. 18, no. 3, pp. 652-663, 2007.

[28] A. W. Moore and C. G. Atkeson, "Prioritized sweeping: Reinforcement learning with less data and less time," *Machine learning*, vol. 13, no. 1, pp. 103-130, 1993.

[29] P. Dayan, and G. E. Hinton, "Feudal reinforcement learning," in *Advances in Neural Inf. Process. Syst.*, pp. 271-278, 1993.

[30] G. Tesauro, "Practical issues in temporal difference learning," in *Advances in Neural Inf. Process. Syst.*, pp. 259-266, 1992.

www.dbpia.co.kr

정 택 현 (Tack-hyun Jung)

2018년~현재 : 건국대학교 IT융합정보보호학과 석사과정

<관심분야> 사이버 보안, 인공지능, 컴퓨터비전

[ORCID:0000-0002-9172-0817]

김 기 천 (Keecheon Kim)

1992년 : Northwestern Univ. 공학박사

1998년~현재 : 건국대학교 컴퓨터공학과 교수

<관심분야> 통신공학, 사이버 보안, 미래인터넷, IoT

[ORCID:0000-0003-3445-3334]

김 상 원 (Sang-won Kim)

2019년~현재 : 건국대학교 컴퓨터정보통신공학과 석사과정

<관심분야> 미래인터넷, IoT, 인공지능, 네트워크 추론

[ORCID:0000-0002-6146-4103]

www.dbpia.co.kr