Journal of Korea Game Society 2021 Oct; 21(5): 17-28
http://dx.doi.org/10.7583/JKGS.2021.21.5.17

게임 프로그래밍

강화학습 에이전트 시야 정보 차이에 의한 학습 성능 비교

김찬섭*, 장시환**, 양성일**, 강신진*
홍익대학교 게임학부*, 한국전자통신연구원 콘텐츠연구본부**
{kimcold, jjangshan, siyang}@etri.re.kr, directx@korea.ac.kr

Comparison of Learning Performance by Reinforcement Learning Agent Visibility Information Difference

Chan Sub Kim*, Si-Hwan Jang**, Seong-Il Yang**, Shin Jin Kang*
School of Games, Hongik University*,
Contents Research Division, ETRI**

요 약

인공지능 스스로가 자신을 발전시켜 최적의 문제 해결 방법을 찾는 강화학습은 여러 분야에서 활용 가치가 높은 기술이다. 특히 게임 분야는 강화학습 인공지능에 문제 해결을 위한 가상 환경을 제공할 수 있다는 장점이 있으며 강화학습 에이전트는 주어진 환경에 대한 정보인 관측 변수를 사용하여 자신의 상황과 환경에 대한 정보를 파악하여 환경에 대한 문제를 해결한다. 본 실험에서는 롤플레잉 게임의 인스턴트 던전 환경을 간략화하여 제작하고 에이전트에게 관측 변수 중 시야에 관련된 관측변수를 다양하게 설정하였다. 실험 결과 각 설정된 변수들이 학습 속도에 얼마나 영향을 주는지를 파악할 수 있었고, 이러한 결과는 롤플레잉 게임 강화학습 연구에 참고할 수 있다.

ABSTRACT

Reinforcement learning, in which artificial intelligence develops itself to find the best solution to problems, is a technology that is highly valuable in many fields. In particular, the game field has the advantage of providing a virtual environment for problem-solving to reinforcement learning artificial intelligence, and reinforcement learning agents solve problems about their environment by identifying information about their situation and environment using observations. In this experiment, the instant dungeon environment of the RPG game was simplified and produced and various observation variables related to the field of view were set to the agent. As a result of the experiment, it was possible to figure out how much each set variable affects the learning speed, and these results can be referred to in the study of game RPG reinforcement learning.

Keywords : Reinforcement learning(강화 학습), Proximal policy optimization(PPO), Game agent(게임 에이전트)

Received: Jul. 20. 2021 Revised: Aug. 18. 2021
Accepted: Sep. 27. 2021
Corresponding Author: Shin Jin Kang(Hongik University)
E-mail: directx@korea.ac..kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

사람과 대결하는 인공지능은 이전부터 여러 가지가 있었으나 대부분은 유한 상태 기계[1]에 기반한 모델에 불과했기에 불특정한 상황이 되면 인공지능이 논리적이지 못한 선택을 하는 경우가 잦았다. 그러나 2016년 구글의 딥 마인드가 스스로 생각하고 발전시켜 바둑을 두는 인공지능 알파고를 개발하였다[2].

이를 기점으로 대중들에게 인공지능에 대한 열풍이 불었고, 특히 게임 업계에서는 인공지능에게 가상 환경을 제공하고, 가상 환경에서 최적의 결과를 낼 수 있는 인공지능 개발을 활발히 진행하고 있다. 2D 러닝 게임인 쿠키런을 학습한 AI 알파런[3], 스타크래프트에서 프로게이머를 압승해버린 알파스타[4], 블레이드 앤 소울에서는 플레이어와 PVP를 하는 수준급의 AI를 만들어서 대중들에게 경기하는 내용을 공개했다[5].

강화학습의 인공신경망 모델이나 학습 완료한 인공지능을 실제 적용하여 결과를 공개한 경우는 많지만, 기대하는 성능의 인공지능 모델을 하기 위해 여러 가지 파라미터 조절해나가며 많은 시행착오를 거쳐야 한다. 또한, 그 뒤에도 해당 환경을 학습하기까지 시간이 걸린다는 단점이 있다. 또한, 학습 대상의 환경과 조건이 복잡할수록 학습 효율성은 더욱 낮아진다[6].

강화학습에서의 인공지능은 학습 환경의 상황을 이해하기 위해 변수로 된 배열을 입력받고 계산된 결과값을 에이전트에게 전달한다. 이때 학습 환경을 이해하기 위해 제공되는 변수를 관측변수라 한다. 본 연구에서는 관측변수 중 에이전트의 시야를 담당하는 부분을 시야 범위와 시야의 밀집도, 시야 거리를 다양하게 설정하고, 목표 달성에까지 걸리는 시간과 그 결과를 분석하여 학습 성능에 미치는 영향을 비교하였다.

이는 게임 내 범용적으로 활용되는 시야 파라미터의 강화학습 내에서의 영향력을 보여줌으로써 후속 연구자들이 관련 연구를 참조하여 강화학습 파라미터를 최적화하는 데 도움을 줄 수 있다.

2. 관련 연구

강화학습은 [Fig. 1] 처럼 에이전트가 어떠한 환경에 대해 Action을 가하면 환경은 변화한 상태와 행동 결과에 따른 보상을 에이전트에게 제시한다. 인공신경망은 해당 상태와 얻은 보상을 확인하고 다음 행동에 대한 환경의 상태 변화와 보상을 계산하여 에이전트가 다음 Action에서 더 좋은 보상을 얻을 수 있는 방향으로 정책을 발전해 나아간다.

[Fig. 1] Reinforcement Learning Diagram

본 연구에서는 강화학습 알고리즘으로 Policy Gradient(PG)[7]의 파생형인 Proximal Policy Optimization(PPO)[8]를 이용하였다. PG는 학습을 진행하여 최대한의 보상을 얻기 위한 방법을 찾아나가는 정책이다. PG는 주어진 상황에서 기대 보상을 최대로 하는 값을 찾는데, 이때 목표 함수가 최대화가 되도록 값을 업데이트한다. 하지만 PG에서 신경망의 가중치와 편향 값의 변화가 정책의 급변화를 일으킬 수 있다[9].

이러한 정책의 급변화 문제를 해결하기 위해 TRPO(Trust region policy optimization)[10]가 등장하였다. TRPO는 목표 함수인 surrogate objective를 제안하고 파라미터 안정화를 위해 이전 Step의 파라미터와 현재 Step의 파라미터 간의 차이를 구하고 이 차이가 너무 크게 되지 않도록 정책 갱신의 전후에 KL(Kullback-Leibler) divergence를 사용하여 정책에 제약을 건다. 그러나 TRPO는 신경망 구조도 복잡하거니와 계산과정

도 복잡하기에 많은 연산량을 요구한다.

이러한 TRPO의 성능을 유지하면서 단점을 보완한 PPO가 등장하였다. PPO는 KL divergenct대신 목적 손실함수에 clipping을 사용하여 이전 정책에서 크게 변화하지 않도록 최적화를 시도하였다. 또한 일정 데이터 샘플을 모아 mini batch를 형성하고, 이를 여러 번에 걸쳐 경사 하강법을 사용하여 정책을 안정적으로 업데이트 한다.

3. 실험 환경

3.1 강화학습 환경 구조

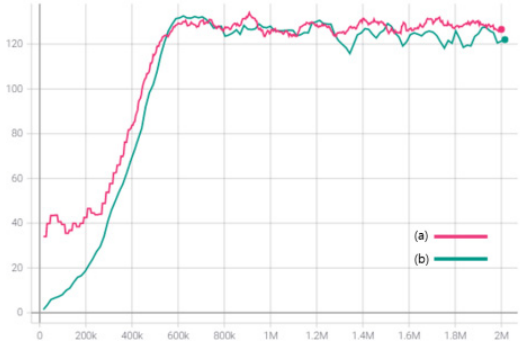
강화학습 라이브러리는 Pytorch 라이브러리[11] 기반인 Stable baselines3[12] 를 사용하였고 학습하기 위한 가상 환경은 게임 제작 엔진인 Unity[13] 를 사용하여 제작하였다. Unity에서 가상 환경의 상태를 알려주는 관측변수, 에이전트의 행동 결과에 따른 보상, 환경이 재시작되었을 때 제공되는 에피소드 초기화 정보를 Stable baselines3에 전달한다. Stable baselines3의 인공신경망은 Unity로부터 제공받은 정보를 토대로 어떠한 Action을 취할지 계산하고, 해당 결과를 Unity에 전달하여 Unity 내부의 에이전트에게 Action대로 움직이게 한다. Python 기반의 Stable baselines3은 Unity와 상호 작용하기 위해 변수를 주고받을 수 있도록 소켓 통신 라이브러리인 ZeroMQ[14] 를 사용하였다. 해당 과정을 이해하기 쉽도록 [Fig. 2]로 도식화시켰다.

[Fig. 2] Interaction Flow between Stable Baseline3 and Unity

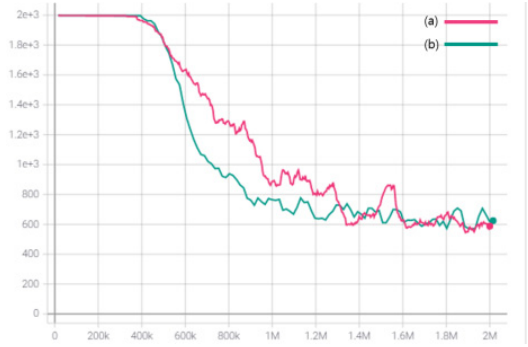
3.2 인공신경망 설정

본 연구는 기준이 되는 모델과 대조군 전부 인공신경망 구조는 전부 같은 것을 사용하였다. 인공신경망에 관측변수를 입력받으면 해당 변수들은 64개의 인공신경망을 통과하고 다시 64개의 인공신경망을 통과한 뒤, 세 번째 신경망에서는 256개를 통과하여 마지막엔 0~9 사이의 변수 중 하나를 출력한다. 에이전트의 훈련은 2만 Step까지 진행하였다.

강화학습에 사용된 알고리즘은 동일한 시야 정보 환경에서 [Fig. 3], [Fig. 4]와 같이 Advantage Actor Critic(A2C) 모델과 PPO 모델을 이용한 사전 실험을 진행하였다.



[Fig. 3] Reward Value Comparison Graph for A2C(a) and PPO(b)



[Fig. 4] Episode Length Graph Comparison for A2C(a) and PPO(b)

결과 비교를 통해 학습 속도와 수렴 성능이 좋은 PPO를 학습 모델로 채택하였으며, 인공신경망

이 다음 행동을 결정하기 위한 세부 조절 내용인 하이퍼 파라미터는 [Table 1]에 표시하였다.

[Table 1] Hyperparameters in Reinforcement Learning

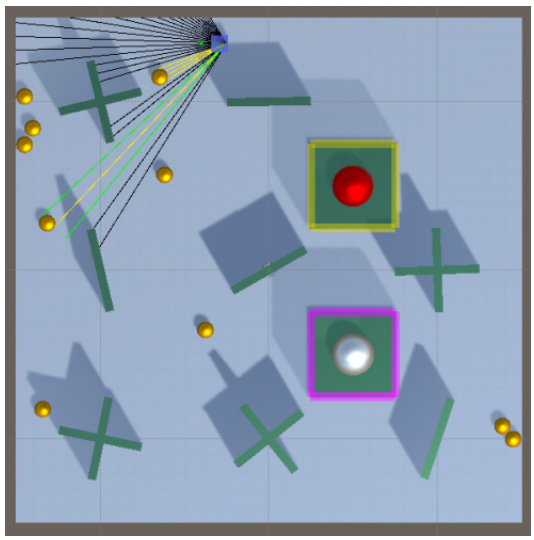
Name	Value	Description
learning_rate	3e-4 ~ 1e-10	Learning intensity control
n_steps	2048	Number of steps for policy renewal
batch_size	64	Divided data size
n_epochs	10	Number of learning episode
gamma	0.99	Discount factor for future rewards
gae_lambda	0.95	Present value estimate coefficient
clip_range	0.2	Range of policy updates
ent_coef	0.0	Entropy coefficient
vf_coef	0.5	Factors for loss calculations
max_grad_norm	0.5	Maximum gradient of policy updates

기본적인 하이퍼 파라미터는 Stable baseline3에서 기본 제공하는 변수로 설정하였으나 경사 하강법에 따른 학습 강도 조절 변수인 learning_rate는 초깃값을 0.0003으로 시작하여 학습이 끝나는 지점인 2만 Step에서는 1e-10에 도달하도록 감소시켰다.

3.3 학습 환경

학습 환경은 RPG 게임에서의 대표적으로 사용되는 던전 형태를 간략하게 압축시켜 가상 환경화 시켰다. 가상 학습 환경의 목적은 다음과 같다. 먼저 생성된 일반 몬스터를 40% 이상 잡으면 중간 보스를 막는 벽이 해제된다. 중간 보스를 잡고 난 뒤, 일반 몬스터를 80% 이상 잡으면 최종 보스를 막는 벽이 해제된다. 마지막으로 최종 보스를 공격하여 hp를 0으로 만들면 에피소드가 종료된다.

가상 학습 환경에서는 에이전트가 2000 Step에 도달하게 되면 초기화가 진행되며, 에피소드가 초기화가 될 때마다 10에서 20 사이의 값으로 무작위 개수로 생성된 일반 몬스터들과 중간 보스 1마리, 최종 보스 1마리를 무작위의 위치에 배치한다. 가상 환경의 지형에도 무작위성을 주기 위해 벽을 배치하였는데, 벽은 지정된 위치 9개의 위치마다 세 가지 타입 중 하나가 무작위로 배치된다. 벽의 종류는 아예 없거나, 일자형 벽, 십자형 벽이 있으며 각 벽은 랜덤하게 회전되어 배치된다. 가상 환경의 크기는 가로, 세로 30이다. [Fig. 5]은 위에서 설명한 학습 환경이 실제로 실행되는 이미지를 보여준다.



[Fig. 5] Training Environment Screen

에이전트는 10가지의 행동을 선택할 수 있다. 8가지는 8방향으로의 이동을 할 수 있고 1가지는 주변 거리 1 이내에 적군이 있을 시 공격, 마지막 1가지는 아무것도 하지 않는 행동으로 두었다.

Stable baselines3에 관측변수를 전달하기 위해서 에이전트는 주변 환경 정보를 수집하여 float 배열의 형태로 채운 뒤 전달한다. 에이전트가 수집

하는 관측변수는 [Table 2]와 같다.

[Table 2] Type of Observation Variable

Name	Type
Current step	Current step/ Maximum number of steps
Current location	Current x, z coordinate
Direction of sight	Two-dimensional direction vector
Agent status	Stop, Move, Attack, Death
Ray information	Type, Distance, Health power
Mid boss wall	Open, Closed
Mid boss status	Alive, Death
Final boss wall	Open, Closed
Final boss status	Alive, Death
Target in range	True, False

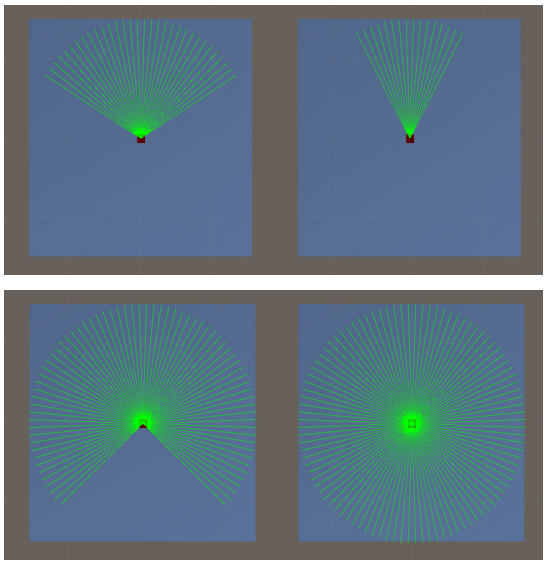
[Table 2]에 있는 Ray 정보의 Ray는 에이전트로부터 뻗어 나가는 직선을 뜻하는데, Ray는 벽이나 적군에 닿게 되면 닿게 된 오브젝트의 종류, 거리, 체력을 반환하게 된다. Ray는 실험 대조군마다 각기 다른 각도, 다른 Ray의 개수, 다른 거리로 뻗어 나간다. 기준점이 되는 에이전트는 120도에 거리 15, 32개의 Ray를 쏘는 에이전트이며 각 Ray당 간격은 3.75도 간격이며 뻗어 나가는 거리는 15로 설정하였다.

첫 번째 실험은 고정된 Ray의 사이 각에서 각도를 조절하여 에이전트의 학습 속도와 성능을 비교하였다. 기준점인 모델 (a)를 기준으로 삼아 다른 모델들은 Ray 간의 사이 각을 고정한 채 각도를 달리하여 학습 진행하였다. 각각의 대조군은 60도, 270도, 360도로 설정하였으며 모델 이름은 (b), (c), (d)로 명명하였다.

[Table 3]으로 해당 모델들의 정보를 정리하였고 [Fig. 6]는 학습 환경 내 Ray가 학습 환경 크기 대비 에이전트의 시야를 시각화한 이미지이며 왼쪽 위부터 (a), (b), (c), (d) 순으로 배치하였다.

[Table 3] Control Group for Range of View

ID	Range	Ray Number	Ray Angle	Distance
(a)	120	32	3.75	15
(b)	60	16	3.75	15
(c)	270	72	3.75	15
(d)	360	96	3.75	15



[Fig. 6] Visualization of Control Group for Range of View (Top Left (a), Top Right (b), Bottom Left (c), Bottom Right (d))

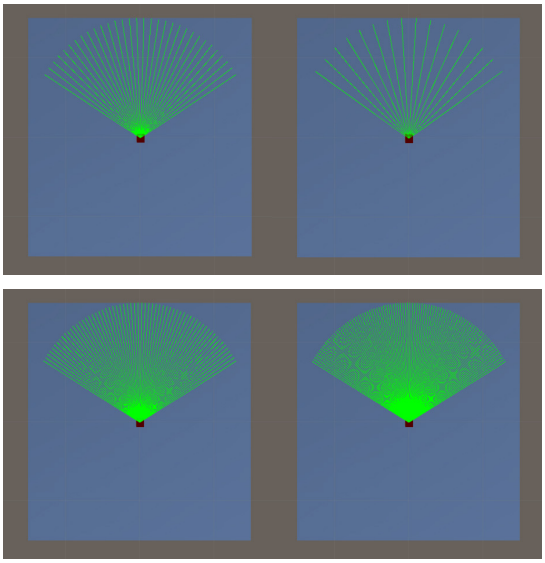
두 번째 실험은 고정된 각도에서 Ray의 수를 달리하여 시야 사이 각을 조절해 실험 진행하였다. 기준점은 (a)이며 대조군은 Ray의 개수를 기본형의 절반인 16개, 2배인 64개, 3배인 96개로 설정하였고 모델 이름은 (e), (f), (g)로 명명하여 학습 진행하였다. [Table 4]는 해당 사항을 정리한 표이며 [Fig. 7]는 해당 모델들의 시야를 시각화한 이미지이다. 순서는 (a), (e), (f), (g) 순으로 표현하였다.

[Table 4] Control Group for Ray

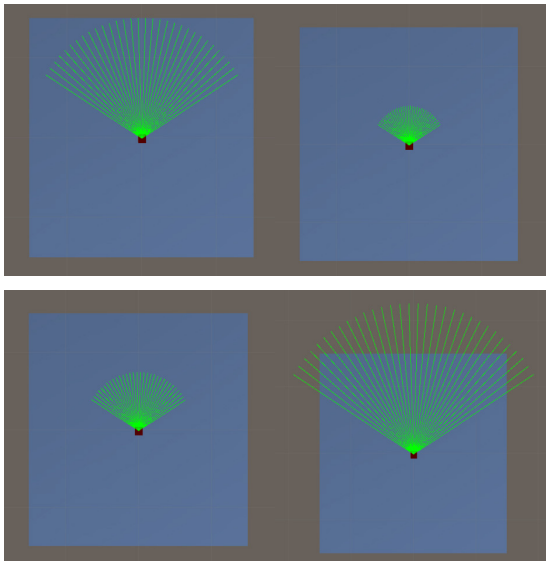
ID	Range	Ray Number	Ray Angle	Distance
(a)	120	32	3.75	15
(e)	120	16	7.5	15
(f)	120	64	1.875	15
(g)	120	96	1.25	15

[Table 5] Control Group for Distance of View

ID	Range	Ray Number	Ray Angle	Distance
(a)	120	32	3.75	15
(h)	120	32	3.75	5
(i)	120	32	3.75	7.5
(j)	120	32	3.75	22.5



[Fig. 7] Visualization of Control group for Number of Ray (Top Left (a), Top Right (e), Bottom Left (f), Bottom Right (g))



[Fig. 8] Visualization of Control Group for Distance of View (Top Left (a), Top Right (h), Bottom Left (i), Bottom Right (j))

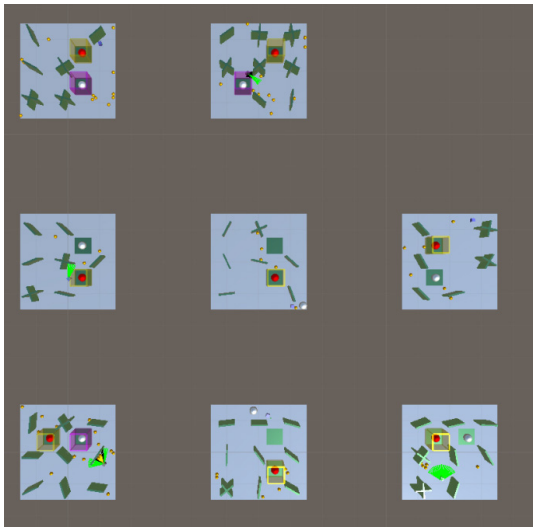
세 번째 실험은 고정된 각도, 고정된 Ray의 수에서 Ray가 뻗어 나가는 길이를 조절하여 실험 진행하였다. 기준점과 대조되는 대조군은 Ray의 거리를 각각 5, 7.5, 22.5로 설정하여 학습 진행하였으며 대조군의 모델 번호는 (h), (i), (j)로 명명하였다. [Table 5]는 해당 사항을 간략하게 정리하여 표현하였으며 [Fig. 8]은 해당 모델들의 Ray를 이미지로 표현한 것이다.

[Table 6]과 같이 보상함수 설계는 에이전트가 빠르게 에피소드를 클리어할 시에 대한 보상을 주기 위해 매 Step 마다 보상을 -0.005(1/환경 최대 Step 수) 씩 제공하였고 적군을 공격하면 0.5씩, 적군의 hp를 0으로 만들었다면 5의 보상, 중간 보스는 10, 최종 보스를 잡았다면 15의 보상을 주도록 설계하였다. 플레이어는 1의 공격력을 가지며 일반 몬스터의 hp는 5, 중간 보스와 최종 보스의 hp는 10으로 설정하였다.

[Table 6] Design of Reward Functions

Description	Reward
Every step	-0.005
Attack enemy	0.5
Kill normal monster	5
Kill mid boss	10
Kill final boss	15

마지막으로 학습 속도 향상을 위해 [Fig. 9]과 같이 분산 에이전트를 사용하였다. 학습 시작 시, Stable baseline3에서 Unity에 생성되어있는 강화학습 환경의 수 만큼 강화학습 환경을 생성 후, 각각 환경을 Unity의 환경에 연결한다. 실험에서는 8개의 학습 환경을 생성하였으며 각각의 환경은 Stable baseline3의 강화학습 환경마다 일대일 대응으로 연결되어 있다.

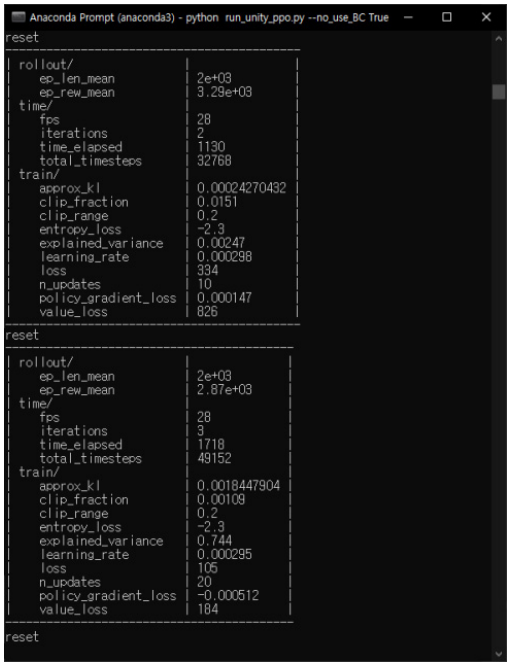


[Fig. 9] Visualization of Eight Distributed Training Environment

4. 실험 결과

4.1 실험 결과 측정 방법

본 실험에서는 RPG게임의 인스턴트 던전을 가상 환경으로 간략화하여 강화학습을 위한 환경으로 제공하였으며, 에이전트가 학습되는 진행 과정은 [Fig. 10]의 콘솔창을 통해 실시간으로 확인할 수 있다.

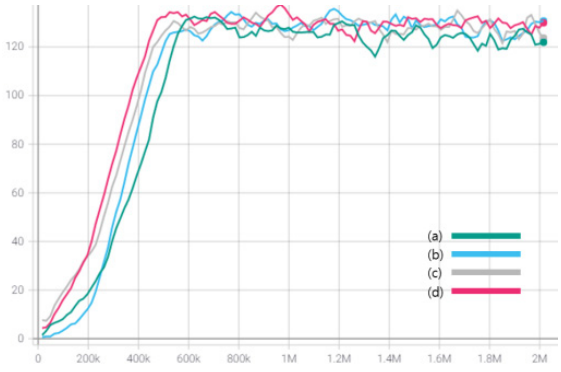


[Fig. 10] Process Screen for Agent Training

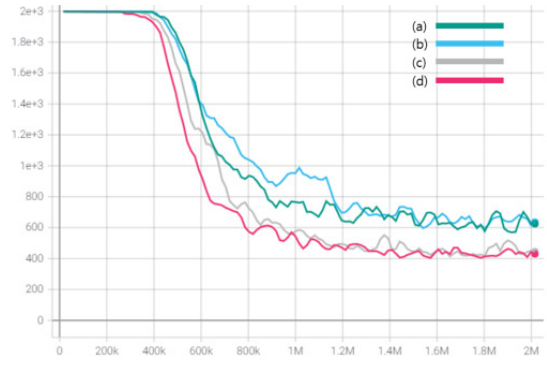
해당 환경에서 실행되는 강화학습에서 에이전트의 시야에 해당하는 Ray에 서로 다른 관측변수를 주어 보상을 얼마나 많이 얻고 빠르게 에피소드를 클리어하였는가에 대한 실험을 진행하였다. 해당 실험의 결과를 확인하기 위해 TensorBoard [15]를 사용하여 로그를 기록하고 그래프로 시각화하였다. 학습에 관한 데이터를 확인하기 위한 그래프는 에피소드당 보상 그래프와 에피소드의 길이를 나타낸 그래프 두 가지로 나누었다. 그래프의 기울기가 크다면 그만큼 학습 속도가 더 빠르다는 것을 뜻하며, 그래프에서 보상이 크고 에피소드 길이가 짧을수록 학습 성능이 높다는 것을 뜻한다.

4.2 시야 범위 조절 비교 실험

시야 범위를 조절한 모델들의 결과는 [Fig. 11]과 [Fig. 12]로 나타내었다. 실험 진행한 학습 환경에서 학습 결과는 (b), (a), (d), (c) 순으로 학습 결과가 좋았다. 학습 도중 (b)가 (a)보다 학습 속도가 더 빨랐던 이유는 (b)는 (a)에 비해 관측변수가 적었기에 환경에 대한 인공신경망의 계산과정이 적었기 때문이지만, 결과적으로는 주변 환경에 대한 관측변수가 (a)보다 적었기에 최종적인 학습 속도와 결과는 낮게 나타났다. 또한 (d)가 (c)보다 학습 속도와 결과가 더 좋게 나타났는데, 이는 시야 정보 내 움직이고 있는 방향의 반대 방향에 대한 정보는 비교적 중요하지 않았다는 것을 보여준다.



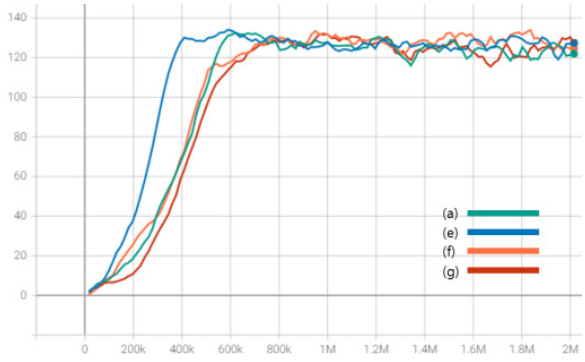
[Fig. 11] Reward Value Graph for Range of View



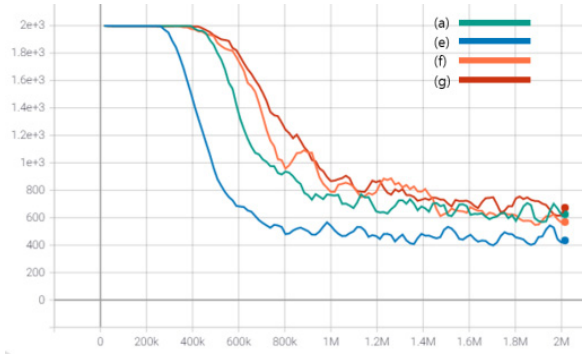
[Fig. 12] Episode Length Graph for Range of View

4.3 시야 Ray 수 조절 비교 실험

관측변수 중 Ray의 개수를 조절한 모델들의 결과는 [Fig. 13]과 [Fig. 14]로 나타내었다. 학습의 결과는 Ray가 적은 순서대로인 (e), (a), (f), (g) 순으로 나타났다. Ray가 많을수록 불필요한 정보나 중복되는 정보가 많았기에 해당 환경을 학습하는 데 시간이 오래 걸렸음을 보여주며 Ray의 사이 사이가 비어있어 사각이 있더라도 학습에는 문제가 없다는 것을 의미한다.



[Fig. 13] Reward Value Graph for Number of Ray

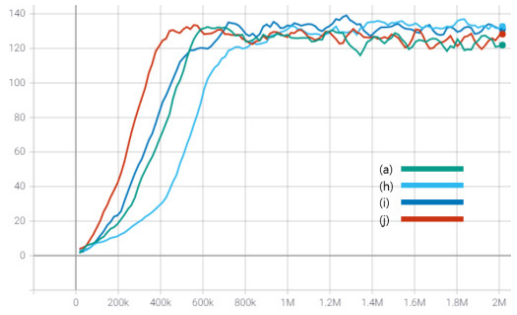


[Fig. 14] Episode Length Graph for Number of Ray

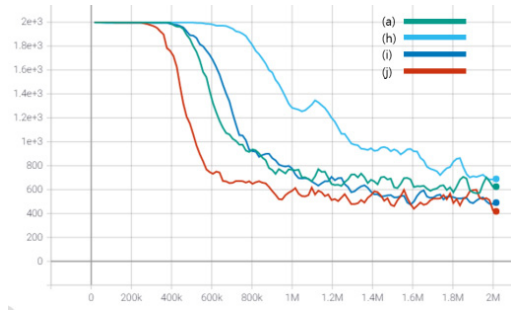
4.4 시야 거리 조절 비교 실험

시야 거리를 조절한 모델들의 결과는 [Fig. 15], [Fig. 16]로 나타내었다. 학습 속도는 (j), (a), (i), (h) 순으로 나타났지만 수렴한 가장 높은 Reward의 결과는 (j), (i), (a), (h) 순으로 나타났다. 학습 환경에서 시야가 길수록 주변을 더 멀리 탐지할 수 있으므로 거리가 더 긴 편이 학습 진행에 유리

하였다. 다만 (a)와 (i)는 학습 속도에 유의미한 차이가 나지 않았는데, 이는 시야가 어느 이상으로 길지 않다면 학습 속도 면에서 많은 차이가 나지 않다는 것을 의미한다.



[Fig. 15] Reward Value Graph for Distance of View



[Fig. 16] Episode Length Graph for Distance of View

마지막으로 [Table 7]로 실험 진행한 결과를 표로 간단히 정리하였다. 이때 정리한 값은 그래프의 Smoothing을 0.8로 두고 기록하였다. 수렴된 Step의 지점은 보상을 기준으로 작성하였다.

첫 번째 실험인 시야 범위 조절 실험에서는 시야가 넓을수록, 의미 없는 변수가 적을수록 학습 속도가 빠르다는 결론을 (d)모델을 통해 알 수 있었고, 둘째로 시야 Ray 수 조절 실험에서는 Ray의 수가 과다하기보다 적절히 적을 때가 학습 성능이 더 좋다는 것을 모델 (e)를 통해 알 수 있었다. 세 번째 실험인 시야 거리 조절 실험에서는 시야 거리가 긴 모델이었던 (j)를 통해 에이전트의 시야 거리가 길수록 주변을 탐지하기 용이하므로 성능이 더 좋다는 결론을 내릴 수 있었다.

[Table 7] Experimental Results for Visibility Information Difference

ID	Episode Length	Converged Reward	Converged Step
(a)	634	123	737k
(b)	645	128	802k
(c)	433	128	622k
(d)	453	127	802k
(e)	458	125	655k
(f)	585	125	1.0m
(g)	664	127	1.0m
(h)	709	132	1.1m
(i)	501	131	802k
(j)	492	128	622k

5. 결론 및 향후 계획

본 연구에서는 일반적인 롤플레잉 게임 상황에서 시야를 정의하는 시야 범위, 시야 Ray 수, 시야 거리 파라미터들이 강화학습에 어떠한 영향을 미치는지를 확인해 보고자 하였다. 향후 연구에서는 각각의 학습된 모델을 다양한 환경을 제공하고 모델의 행동 양상을 기록하여 모델별로 어느 환경이 더 좋은 성적을 내는지에 대해 실험을 진행할 예정이다.

강화학습은 인공지능이 주어진 환경에서 스스로 발전해나가 환경을 효율적으로 풀어나가는 기술이며 이는 가상 환경 제공이 유리한 게임 분야에서 요긴하게 사용될 수 있는 기술이다. 본 논문에서 제안하는 강화학습 프레임워크를 응용하면 사용자의 수준에 알맞게 난이도를 조절하는 게임 밸런스 업무, 혹은 사용자의 데이터를 학습하여 이를 모방하는 비동기 플레이 시스템 구축이 가능하다. 또한 기존에 기획팀에서 수작업으로 이루어지던 다양한 파라미터 최적화 작업에도 응용될 수 있을 것이다.

ACKNOWLEDGMENTS

This research is supported by the Ministry of Culture, Sports and Tourism and Korea Creative Content Agency (Project Number: R2019020067). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. NRF-2019R1A2C1002525).

REFERENCES

[1] Marzian, F., & Qamal, M. (2017). Game RPG "The Royal Sword" Berbasis Desktop Dengan Menggunakan Metode Finite State Machine (FSM). Jurnal Sistem Informasi, 1(2).

[2] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... & Hassabis, D. (2017). Mastering the game of go without human knowledge. nature, 550(7676), 354-359.

[3] Teahoon Kim, "Implementing Cookie Run AI that is better than me with deep learning and reinforcement learning", slideshare, last modified Oct 25, 2016, accessed May 24, 2021, https://www.slideshare.net/carpedm20/ai-67616630.

[4] Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., ... & Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature, 575(7782), 350-354.

[5] Sangbin Moon, "Generation of progamer level Bimu AI using reinforcement learning in Blade and Soul", NDC, last modified Jul 24, 2019, accessed May 26, 2021,

[6] Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009, June). Curriculum learning. In Proceedings of the 26th annual international conference on machine learning (pp. 41-48).

[7] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

[8] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347.

[9] Soo Yeong Jang, et al. Deep reinforcement learning technology trends, ETRI Electronics and Telecommunications Trends, 34.4 (2019): 1-14.

[10] Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, June). Trust region policy optimization. In International conference on machine learning (pp. 1889-1897). PMLR.

[11] Pytorch Library, https://pytorch.org/

[12] Stable Baselines 3, https://github.com/DLR-RM/stable-baselines3

[13] Unity Engine, https://www.unity.com/

[14] ZeroMQ library, https://zeromq.org/

[15] Tensorboard, https://www.tensorflow.org/tensorboard

김 찬 섭 (Kim, Chan Sub)

2019년-현재 한국전자통신연구원, 위촉연구원
2019년-현재 홍익대학교 게임학부 소프트웨어전공 석사과정
2016년-2016년 Xiness 인턴연수생
2011년-2018년 홍익대학교 게임학부 소프트웨어전공 공학사

관심분야 : 게임인공지능, 컴퓨터 그래픽스, 게임 개발

장 시 환 (Jang, Si-Hwan)

2013년-현재 한국전자통신연구원, 선임연구원
2010년-2013년 고려대학교 공학석사
2004년-2010년 강원대학교 공학사

관심분야 : 게임 인공지능, 인공지능 표준화, 메타버스

양 성 일 (Yang, Seong-Il)

2000년-현재 한국전자통신연구원, 책임연구원
1998년-2000년 아시아나항공 SW연구소 주임연구원
1996년-1998년 연세대학교 박사수료

관심분야 : 기계학습, 게임 인공지능, 자연어처리

강 신 진 (Kang, Shin Jin)

2008년-현재 홍익대학교 게임학부, 부교수
2011년 고려대학교 정보통신대학 컴퓨터학과 이학박사
2003년-2006년 소니 컴퓨터 엔터테인먼트 코리아
2006년-2008년 엔씨소프트

관심분야 : 게임 인공지능, 게임 기획, 컴퓨터 그래픽스