2023년도

학사학위논문

PYSC2를 이용한 강화학습 에이전트 개발

Development of a Reinforce Learning Agent using PYSC2

2023년 11월 20일

순천향대학교 공과대학  
컴퓨터공학과

한지훈

PYSC2를 이용한 강화학습 에이전트 개발

Development of a Reinforce Learning Agent using PYSC2

지도교수 이상정

이 논문을 공학사학위 논문으로 제출함

2023년 11월 20일

순천향대학교 공과대학

컴퓨터 공학과

한지훈

의 공학사학위논문을 인준함

2023년 11월 20일

심 사 위 원 홍 인 식 인

지 도 교 수 이 상 정 인

순천향대학교 공과대학

컴퓨터공학과

**초 록**

강화 학습이라는 알고리즘이 등장하고 발전하면서 간단한 문제들에 적용되어 문제를 해결하는 것에 그치지 않고 더 다양하고 복잡한 문제들을 해결하고 있다.

강화 학습이 해결하고 있는 문제들 중에는 게임도 존재하는데 강화 학습이라는 알고리즘이 사용되기 시작한 초기 아타리 게임 위주로 학습이 진행되었던 것과 다르게 요즘은 바둑이나 더 복잡한 게임에도 적용되고 있다.

구글의 딥 마인드는 알파고와 이세돌의 대국 이후 강화학습을 턴제 보드 게임이 아닌 StarCraft II에 적용하기 위해 블리자드와 협력하여 pysc2라는 파이썬 라이브러리를 개발했다. 본 논문은 이 pysc2 라이브러리를 이용하여 StarCraft II를 플레이하는 강화 학습 에이전트를 학습시켜 더 향상된 에이전트를 통해 알파고와 이세돌의 대국처럼 인간과 AI 간의 대결에서 좋은 결과를 내기 위한 목표로 에이전트의 승률을 기존의 에이전트보다 향상 시키는 것이 목적이다. 그 결과 기존 에이전트의 승률보다 높은 80%의 성능을 내는 결과를 얻었고 향후 개선방안에 대해서 분석한다.

키워드: StarCraft II,pysc2,강화학습

**차례**

**제1장 서론**

**제2장 PYSC2 소개**

**2.1 pysc2**

**제3장 환경과 에이전트 설계 및 학습**

**3.1 게임 소개**

**3.1.1스타크래프트**

**3.2 게임 환경**

**3.2.1 Simple64**

**3.2.2 게임 설정**

**3.2.3 상태**

**3.3 에이전트 개발**

**3.3.1 Build Order**

**3.3.2 에이전트 설계**

**3.4 학습**

**3.4.1 Q-Learning**

**3.4.2 DQN**

**제4장 결과분석**

**제5장 한계 및 향후 개선 방안**

**감사의 글**

**참고문헌**

**그림 차례**

**[그림 1] 스타크래프트2**

**[그림 2] Pysc2와 스타크래프트2 사이의 관계**

**[그림 3] 미니게임화면**

**[그림 4] 미니게임 “DefeatRoaches”를 pysc2를 이용하여 학습시킨 결과**

**[그림 5] full-game 학습(Steven Brown, https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255)**

**[그림 6] 스타크래프트2 게임화면**

**[그림 7] 스타크래프트2 블록도(테란)**

**[그림 8] 스타크래프트2 게임 순서**

**[그림 9] Simple64 (초록색이 아군 진영 파란색이 적군 진영이다.)**

**[그림 10] 게임환경과 에이전트의 관계**

**[그림 11] 에이전트의 설계**

**[그림 12] 게임 종료 후 점수 화면**

**[그림 13] score그래프**

**[그림 14] score 그래프 마지막 200번 결과 그래프**

**[그림 15] 학습결과(초록색: 승률, 빨간색: 패배 비율, 파란색: 무승부 비율)**

**[그림 16] full-game 학습(Steven Brown, 2017,** [**https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255**](https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255)**)**

**제 1 장 서 론**

구글 알파고의 등장으로 많은 사람들이 AI 개발에 주목하게 되었고 이는 매우 급격한 속도로 발전하기 시작했다. 아타리 게임 같은 단순한 게임을 넘어서 자율 주행이나 로봇을 조종하고 드론을 조종하는 등 복잡한 문제도 강화 학습을 통해서 해결하고 있다.

뿐만 아니라 바둑이나 체스 같은 턴제 형식의 게임이 아닌 다양한 게임에도 강화학습이 적용되어 인간과 대결을 펼치고 있는데 이는 스타크래프트2 역시 마찬가지이다. 본 논문에서는 스타크래프트2를 강화 학습을 통해 학습시키는 것이 목표이다.

강화 학습[1]이란 기계학습의 한 영역으로 어떤 환경 내에서 정의된 에이전트가 행동을 선택하고 보상과 처벌을 받으면서 보상을 최대화하는 행동 혹은 순서를 선택하는 것을 말한다. 본 논문에서는 Pysc2 라이브러리를 이용하여 에이전트를 학습시키고 성능을 분석하는 것에 초점을 맞추었다. 특히 미니게임이 아닌 full-game을 학습시켜 스타크래프트2 봇을 상대로 어떤 결과를 내는지 분석하는 것이 목표이다. 또한 한계의 뿐만 아니라 더 나은 성능을 내기 위해 어떤 점을 개선해야 하는지 분석하는 것도 포함한다. 궁극적으로 더 나은 성능의 에이전트를 개발하여 더 높은 승률을 내는 것이 목표이다.

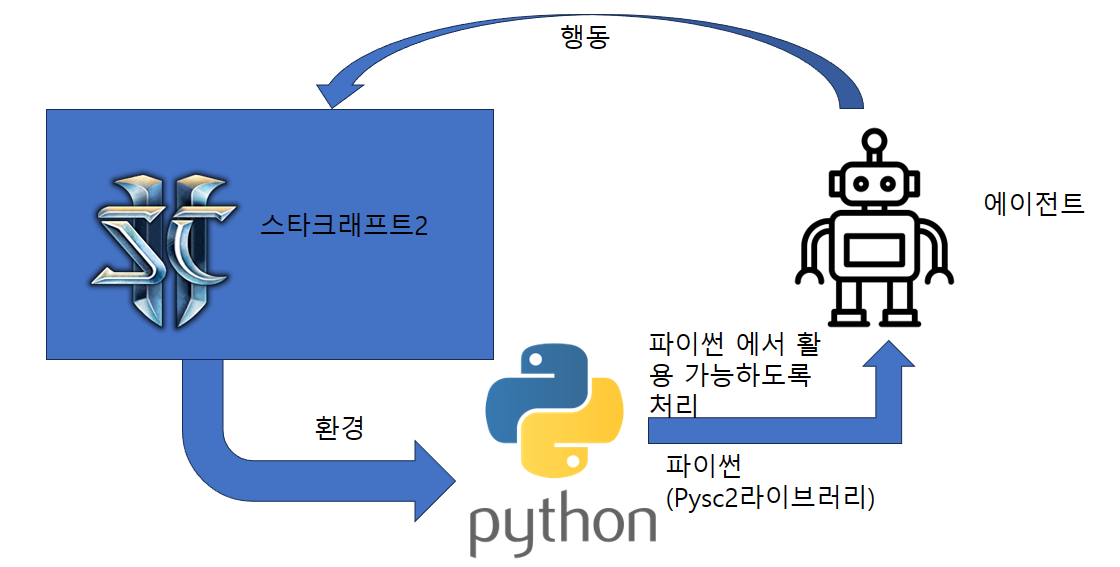
**제 2 장 PYSC2 소개**

**2.1 pysc2**



**[그림 1] 스타크래프트2**

Pysc2는 블리자드와 구글 딥마인드가 협력하여 만든 파이썬 라이브러리로 강화학습을 통해 스타크래프트2를 플레이하는 에이전트를 개발하기위해 널리 쓰이고 있다. 강화학습을 위해 중요한 것 중 하나가 환경인데 아타리 게임같은 단순한 게임과는 달리 스타크래프트2처럼 복잡하고 여러 상황이 존재하는 게임은 환경을 정의하기가 어렵다. 하지만 Pysc2 라이브러리를 이용하면 에이전트를 설계하기도 쉽고 학습을 시키기에도 용이하다. Pysc2를 이용하면 BaseAgent를 비롯한 다양한 모듈을 이용하여 에이전트를 처음부터 설계할 필요없이 BaseAgent를 상속받아서 학습시키려는 주제에 맞게 에이전트를 설계하면 된다. 뿐만 아니라 유닛을 선택, 공격, 건설까지 모두 메서드로 정의되어 있기 때문에 이를 잘 조합하여 설계한 뒤 학습시키면 뛰어난 스타크래프트 AI를 만들 수 있다.



**[그림 2] Pysc2와 스타크래프트2 사이의 관계**

Pysc2라이브러리[2]를 이용하여 만들어진 에이전트가 현재 상태를 이해할 수 있도록 pysc2라이브러리가 환경을 상태값으로 처리하고 에이전트는 상태값을 통해서 다음행동을 선택할 수 있다.

Pysc2라이브러리에 있는 BaseAgent는 다음과 같이 구성되어있다.

def \_\_init\_\_(self):

애이전트를 초기화하는 메서드

def setup(self, obs\_spec, action\_spec):

환경을 설정하는 메서드

def reset(self):

에피소드 처음에 호출되어 일부 상태 변수를 재설정하는 메서드

def step(self, obs):

매 에피소드의 타임스탭마다 호출되는 메서드

주로 이 메서드 안에서 행동이 결정된다.

환경은 Rendered, Feature Layer, Raw 이렇게 3가지 방식으로 전달되는데

‘Rendered’ 방식은 게임 화면을 나타낸다. 에이전트가 화면을 직접 보고 행동한다면 이 방식을 통해서 환경을 시각적으로 전달한다.

‘Feature Layer’ 방식은 화면 또는 미니맵 상의 특정 위치에 대한 게임 상태 정보로 위치,지형,자원 분포 등과 저수준 정보를 전달하는 방식으로 에이전트가 시각적 정보를 얻을 수 있도록 한다.

‘Raw’ 방식은 게임 내 유닛 및 빌딩과 같은 객체에 대한 낮은 수준의 상세 정보를 전달하는 방식으로 유닛의 현재 위치, 체력 등과 같은 정보를 전달한다. Raw 방법을 사용할 경우 화면을 굳이 이동하지 않고 에이전트가 플레이가 가능하도록 도와준다.



**[그림 3] 미니게임화면**

그리고 obs.observation 객체를 이용하여 다양한 시각적 정보에 접근이 가능하다.

미니게임의 경우 화면좌측 상단에 나타나는 현재까지 누적된 점수에 접근할 방법이 필요한데 이 경우에도 obs.observation객체가 유용하게 쓰인다.

미니게임뿐만 아니라 full-game에서도 cumulative score가 게임 종료후 볼 수 있는데 이 데이터에 접근할때도 obs.observation객체를 통해 접근이 가능하다.

게임 화면과 미니맵에도 접근이 가능하기 때문에 raw방법이 아닌 직접 화면에 접근해야 할 경우에는 obs.observation['feature\_screen'] 및obs.observation['feature\_minimap'] 메서드를 이용하여 접근이 가능하다.

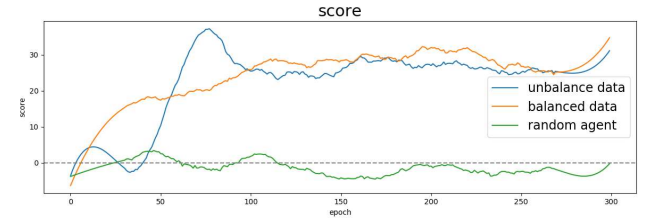
obs.observation['player'] 를 이용하여 플레이어에 대한 정보에 접근이 가능하다.

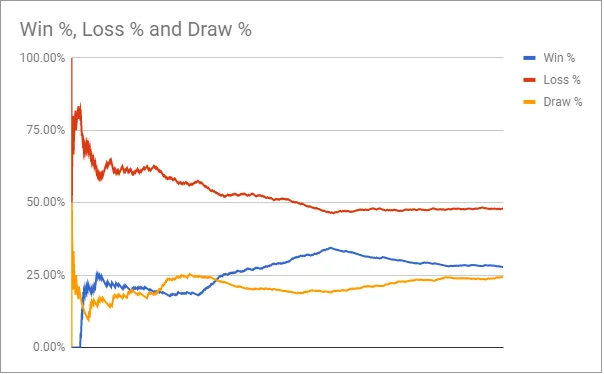
자원에 해당하는 미네랄, 가스, 인구 수와 같은 플레이어 정보가 이에 해당한다.

obs.observation['available\_actions']과 obs.observation['last\_actions']을 통해 현재 가능한 행동과 직전 행동에 대한 정보를 알 수 있다.

이 외에도 더 많은 정보가 obs.observation 객체에 포함되어 있기 때문에 게임 환경에 필요한 정보를 편하게 접근하게 해주어 강화 학습을 하는데 수고를 덜어준다.

이러한 이유때문에 많은 사람들이 pysc2라이브러리를 이용하여 스타크래프트2 full-game을 학습시킨 연구도 있고 미니게임을 통해 full-game을 학습시키려는 방법도 수행중이다. [3]



**[그림 4] 미니게임 “DefeatRoaches”를 pysc2를 이용하여 학습시킨 결과**   
  
CNN을 이용하여 미니게임 “DefeatRoaches”를 학습시켜 35점의 성능을 보인 경우도 있고 (정의진, 김경중. 2019. CNN구조를 이용한 Starcraft II 미니게임 모방학습. 2019년 한국소프트웨어종합학술대회 논문집.1518)   


**[그림 5] full-game 학습 결과(Steven Brown, https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255)**  
Q-Learning을 이용하여 full-game을 학습시킨 경우도 있다.[4] 마린을 생산하고 공격하기까지의 행동만 정의된 에이전트로 학습결과 약 50% 정도의 승률을 나타냈다.

**제 3 장 환경과 에이전트 설계 및 학습**

**3.1 게임 소개**

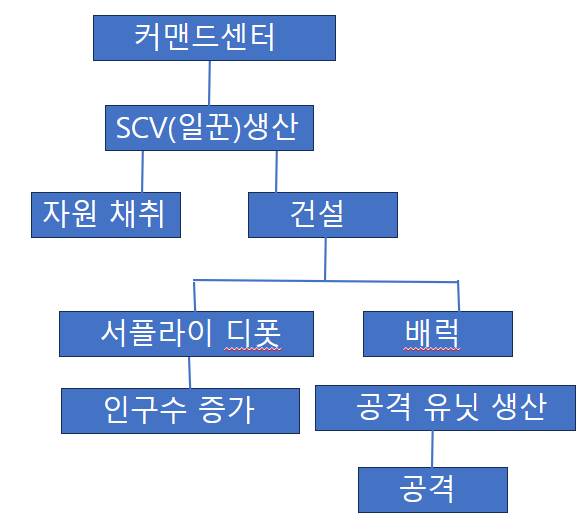
**3.1.1스타크래프트**

블리자드 엔터테인먼트에서 제작한 실시간 전략 게임(Real-Time Strategy, RTS) 스타크래프트의 후속작으로 턴제로 게임이 진행되는 바둑이나 체스와 달리 게임상에 주어진 모든 요소를 활용하여 적을 없애고 목적을 달성하는 게임을 말한다.[5] 때문에 여러 종류의 수많은 유닛을 계속해서 컨트롤 해야한다. 때문에 키보드와 마우스를 통해 컨트롤 하는 것이 일반적이다.

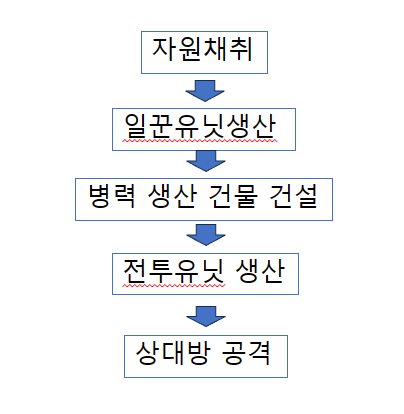


**[그림 6] 스타크래프트2 게임화면**

스타크래프트2는 일반적으로 일련의 과정들을 순서대로 진행하면서 진행된다. 그리고 건물마다 생산이 가능한 유닛의 종류도 다르고 유닛마다 가능한 행동이 모두 다르다. 공격 유닛이라고 하더라도 근접에서 공격하는 유닛도 있으며 굉장히 먼 거리에서 공격이 가능한 유닛도 있고 공격을 못하고 스킬만 사용이 가능한 유닛도 존재한다.



**[그림 7] 스타크래프트2 블록도(테란)**



**[그림8] 스타크래프트2 게임 순서**

일반적으로 스타크래프트2는 위와 같은 방식으로 게임이 진행된다. 스타크래프트2 종족 중 테란을 기준으로 설명하면 가장 중심이 되는 건물은 커맨드 센터로 SCV를 생산하는 것이 주된 목표이다. SCV가 없을경우 자원을 채취할 수 없고 그렇게 되면 다른 건물을 건설할 수도 다른 유닛을 생산할 수도 없기 때문에 커맨드센터를 보호하면서 SCV의 수를 계속 유지하여 자원을 안정적으로 채취하는 것이 중요하다. 그렇게 모은 자원으로 다른 건물들을 건설하여 생산가능한 최대 유닛의 수를 늘리는 서플라이 디폿과 공격 유닛을 생성할 수 있는 건물을 건설하여 공격 유닛을 모을 수 있다. 간단하게 설명하면 처음엔 SCV를 통해서 자원을 채취하고 그 후에 병력 생산을 위한 건물을 건설하고 그 건물에서 전투 유닛을 생산하여 병력을 모아 상대방을 공격하여 무력화시키는 것이 목표이다.

**3.2 게임 환경**

**3.2.1 Simple64**



**[그림 9] Simple64 (초록색이 아군 진영 파란색이 적군 진영이다.)**

Pysc2에 존재하는 미니맵 중 하나로 10시와 4시 방향 둘 중 한 곳으로 시작 진영이 정해지며 바로 옆에 앞마당(본진 바로 앞에 자원을 채취할 수 있는 지형)이 존재하는 맵이다. 두 시작 진영 사이의 거리가 매우 가깝고 맵이 작은 것이 특징이다.

이 맵으로 강화 학습을 할 경우 많은 상태 값을 에이전트에게 넘겨줄 수 있으나 상태 값이 너무 많고 복잡할 경우 학습하는데 시간도 오래 걸리고 학습이 성공적으로 이루어지지 않을 가능성이 있기 때문에 꼭 필요한 상태 값만 에이전트에 넘겨주는 것으로 설계했다.

**3.2.2 게임 설정**

상대방 봇의 난이도는 쉬운 난이도로 설정했고 상대방 종족도 아군과 같은 테란으로 설정했다. 게임이 끝나지 않고 일정 시간이 지날 경우 무승부로 게임을 끝내도록 하여 게임이 승부가 나지 않고 영원히 이어지는 문제를 방지했다. 전장의 안개(아군 유닛이 없는 곳은 어둡게 표현되어 확인할 수 없는 장치)로 인해 상태 값을 파악하지 못하는 문제를 방지하기 위해 전장의 안개 기능을 끄고 게임을 진행하도록 설정했다. 이 문제는 치트키로 해결하였다.

**3.2.3 상태**

에이전트에게 전달 되는 상태 값은 다음과 같다.

return (len(command\_centers), 아군 커맨드 센터의 수

len(scvs), 아군 scv의 수

len(idle\_scvs), 아무것도 하지 않고 대기중인 scv의 수

len(supply\_depots), 아군 서플라이 디폿의 수

len(completed\_supply\_depots),아군의 완성된 서플라이 디폿의 수

len(barrackses), 아군의 배럭 수

len(completed\_barrackses), 아군의 완성된 배럭 수

len(marines), 아군의 마린 수

queued\_marines, 아군의 대기 중인 마린의 수

free\_supply, 남은 인원

can\_afford\_supply\_depot, 서플라이 디폿 건설 가능 여부

can\_afford\_barracks, 배럭 건설 가능 여부

can\_afford\_marine, 마린 생산 가능 여부

len(enemy\_command\_centers), 적군 커맨드 센터 수

len(enemy\_scvs), 적군 scv 수

len(enemy\_idle\_scvs), 적군 scv중 아무것도 하지 않고 대기중인 scv 수

len(enemy\_supply\_depots), 적군 서플라이 디폿의 수

len(enemy\_completed\_supply\_depots), 적군 완성된 서플라이 디폿 수

len(enemy\_barrackses), 적군 배럭 수

len(enemy\_completed\_barrackses), 적군 완성된 배럭의 수

len(enemy\_marines)) 적군 마린 수

이중 적군 커맨드 센터의 경우 상대 봇이 커맨드 센터를 Orbital Command나 Planetary Fortress로 업그레이드 하는 경우 커맨트 센터로 인식하지 못하는 상황이 발생하기 때문에 업그레이드가 완료된 커맨드 센터도 인식하도록 설계했다. 적군 커맨드 센터의 수는 상대방의 앞마당 멀티 유무를 파악하기 위한 수단이므로 적군 커맨드 센터의 수를 정확하게 파악하여 에이전트에게 넘겨주는 것이 중요하다.

**3.3 에이전트 개발**

**3.3.1 Build Order**

빌드 오더 흔히 빌드라고 하는데 이는 기본적인 전략 구사를 위해 건물들을 짓고, 병력을 생산하는 순서를 말한다.[6] 스타크래프트2같은 RTS 게임에서는 굉장히 중요한 요소이다. 빌드에 따라 초반에 승부수를 띄우고 상대방을 쉴 틈 없이 압박하여 빠르게 게임을 끝낼 수도 있고 상대의 공격을 버텨내며 후반에 승부를 보는 방법 등 게임의 역사가 긴 만큼 빌드도 다양하다. 수많은 빌드 중 Simple 64 맵에서 효과적인 빌드는 투배럭 빌드이다. 본 논문의 목표는 에이전트가 게임을 통해 투배럭 빌드를 학습하여 빠르게 마린을 생성한 뒤 상대방을 공격하여 빠른 시간안에 상대방을 무력화하는 것이 목표이다.

**3.3.2 에이전트 설계**

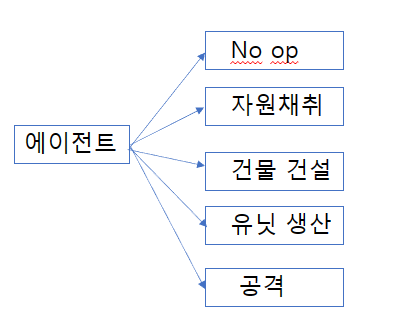


**[그림 10] 게임환경과 에이전트의 관계**

에이전트란 강화 학습에서 환경과 상호 작용하며 행동을 결정하고 그에 따른 보상과 처벌을 받으며 의사결정을 하는 주체를 말한다.[7] 본 논문의 에이전트의 경우 스타크래프트2 게임환경에서 존재하는 여러 상태 값들을 입력 받아 가장 높은 가치 함수값을 갖는 행동을 선택하는 것이라고 할 수 있다.

먼저 에이전트가 할 수 있는 행동을 정의하면

NO-OP(아무 것도 안하기), 미네랄(자원) 채취, 건물(Supply depot, Barrack) 건설, 마린(공격 유닛) 생산, 공격 이렇게 총 5가지이다.



**[그림 11] 에이전트의 설계**

미네랄 채취하는 행동은 pysc2에 내장되어 있는 Harvest\_Gather\_unit 메서드를 이용하여 정의하였다. 이 때 모든 종류의 미네랄을 채취할 수 있도록 하기 위해 필드에 존재하는 각각의 미네랄 필드를 모두 인식할 수 있도록 조건문에 포함 시켰다

건물을 건설하는 행동은 Build\_SupplyDepot\_pt과 Build\_Barracks\_를 이용하여 정의하였다. 초기에는 서플라이 디폿과 배럭을 하나씩만 건설하였으나 투배럭 빌드를 구현할 수 있도록 하기 위하여 배럭을 2개 이상 서플라이 디폿은 제한없이 만들 수 있도록 설계했다.

마린을 생산하는 행동은 Train\_Marine\_quick메서드를 사용하여 정의하였다.

마지막 행동인 공격은 Attack\_pt 메서드를 이용하여 정의하였다. 하지만 이 과정에서 상대 본진의 위치를 공격 위치로 설정할 경우 상대방의 앞마당 멀티를 공격하지 못하기 때문에 상태값에서 상대방의 커맨드 센터의 개수를 이용하여 2개 이상일 경우에는 앞마당을 먼저 공격하도록 정의했다.

**3.4 학습**

**3.4.1 Q-Learning**

먼저 Q 학습(Q-Learning)[8]을 이용하여 학습을 시도했다.

Q 함수를 이용하는 Q 학습은 Q 함수 값을 최대로 하는 행동값(argmax Q(s,a))을 찾아내는 방법으로 학습한다.

Q 학습은 행동가치함수 Q(s,a)를 이용한 강화학습 방법으로 타겟값을 이용하여 학습한다. 이 타겟값을 바탕으로 행동 가치함수  [**수식1]**을 갱신한다.

타겟 정책은 그리디 방식을 적용하고 동작 정책은 입실론 그리디 정책을 이용하는 것이 특징이다.

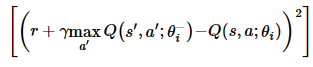
**[수식2]**

이를 통해 Q-Table을 작성하고 정책을 통해서 행동을 선택하게 된다.

그러나 Q-Table로 모든 상태를 표현하고 Q 함수 만으로 행동을 선택하기에는 상태가 너무 많고 복잡하기 때문에 DQN을 이용하여 학습하는 것으로 결정했다.

**3.4.2 DQN**

DQN(Deep Q-Network) 은 지도 학습과 같이 강화 학습을 학습하도록 개발된 알고리즘으로 구글 딥 마인드에서 개발했다.[9] 특히 알파고에 적용되어 유명해진 학습 알고리즘이다. DQN은 Q-Table대신 신경망을 이용하여 Q 함수를 근사화하고 이를 통해 Q 값을 추정하는 것이 Q-Learning과의 큰 차이점이다. 신경망을 통해서 학습을 하기 위해서는 손실함수가 필요한데 본 논문의 경우 손실함수로 MSE를 사용했다.

 **수식[3]**

그리고 신경망을 업데이트하기 위한 방법으로 경사 하강법을 사용했다.

경사 하강법(Gradient descent)은 1차 근삿값 발견용 최적화 알고리즘이다. 기본 개념은 함수의 기울기(경사)를 구하고 경사의 반대 방향으로 계속 이동시켜 극값에 이를 때까지 반복시키는 것이다.[10]

이를 통해 기울기가 0이 되는 점을 발견하여 손실함수의 값을 최소화 하는 것이 목표이다.

DQN은 환경을 탐색한 뒤 경험 튜플을 메모리에 저장하고 저장된 샘플을 랜덤하게 선택하여 학습한다. 이 방법을 통해 데이터 샘플들 간의 독립성을 보장할 수 있다.

또한 Target-network를 구성하여 타겟값이 계속 움직이면서 발생하는 불안정성을 해결할 수 있다.

신경망은 입력층을 총 21개 출력층을 6개로 설정했다. 에이전트가 전달 받는 상태값이 총 21개이고 에이전트가 선택할 수 있는 행동이 총 6개이기 때문이다. 학습률은 일반적으로 많이 사용하는 1e-4로 설정했다. 또한 탐색을 위해 입실론 그리디 정책을 사용하여 행동을 결정하기 때문에 초기 입실론 값은 0.9로 설정하고 150번의 에피소드 이후부터 입실론을 0.001의 감가율로 감소시켜 학습이 진행됨에 따라 탐색의 중요도가 낮아진다는 문제를 해결했다. 보상에 대한 할인율은 1로 설정했다. 에피소드가 종료된 후 경기의 승패 결과가 중요하기 때문에 1로 설정했다. 뉴런의 수는 128개로 설정했고 은닉층의 활성화 함수는 ‘Relu’를 사용했다. 출력층의 활성화 함수는 Q-값을 직접적으로 사용해야하므로 입력을 그대로 출력하는 ‘Identity’함수를 사용했다. 학습 횟수는 1000에피소드로 설정했다.

개발 환경은 window 10 CPU 12th Gen Intel(R) Core(TM) i5-12600K Ram 32GB

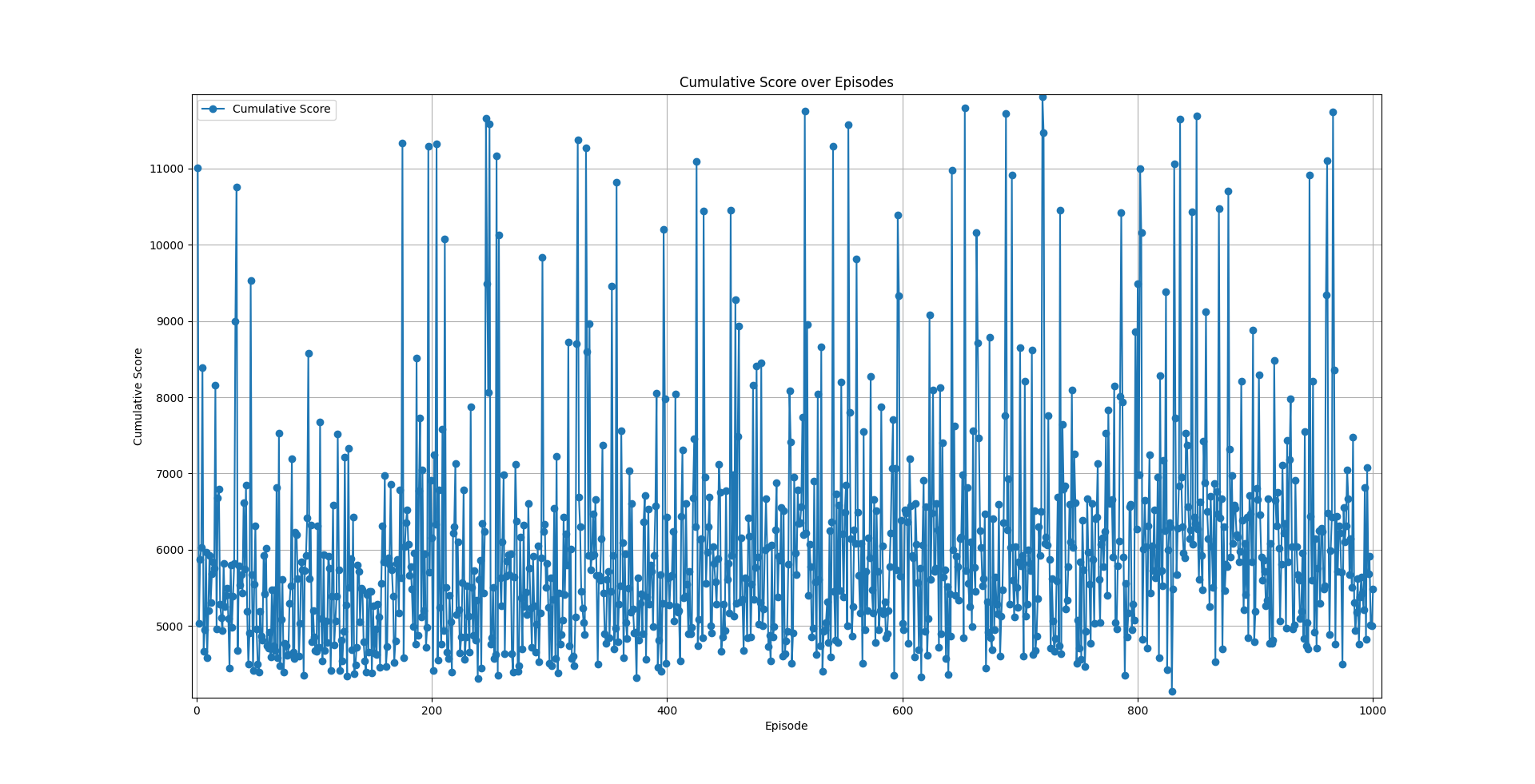
GPU NVIDIA GeForce GTX 1660 Ti ANANCONDA 가상환경에서 VS CODE를 이용하여 개발했다.

**제 4 장 결과분석**

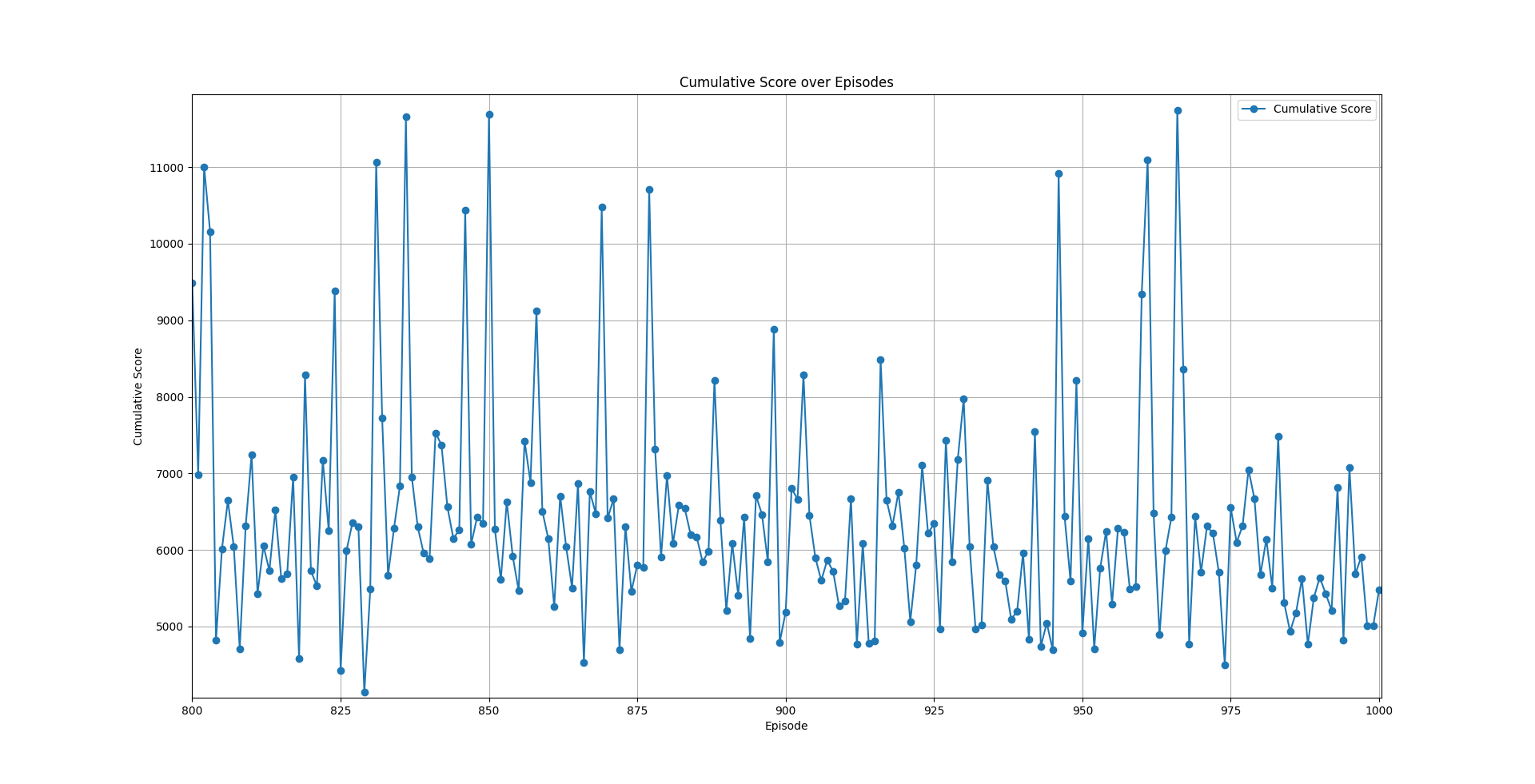


**[그림 12] 게임종료 후 점수 화면**

스타크래프트2는 게임 종료 후 화면에서 그래프로 생산한 유닛, 건설한 건물, 채취한 자원을 점수로 변환해서 보여주는데 이를 그래프로 나타내면 다음과 같다.



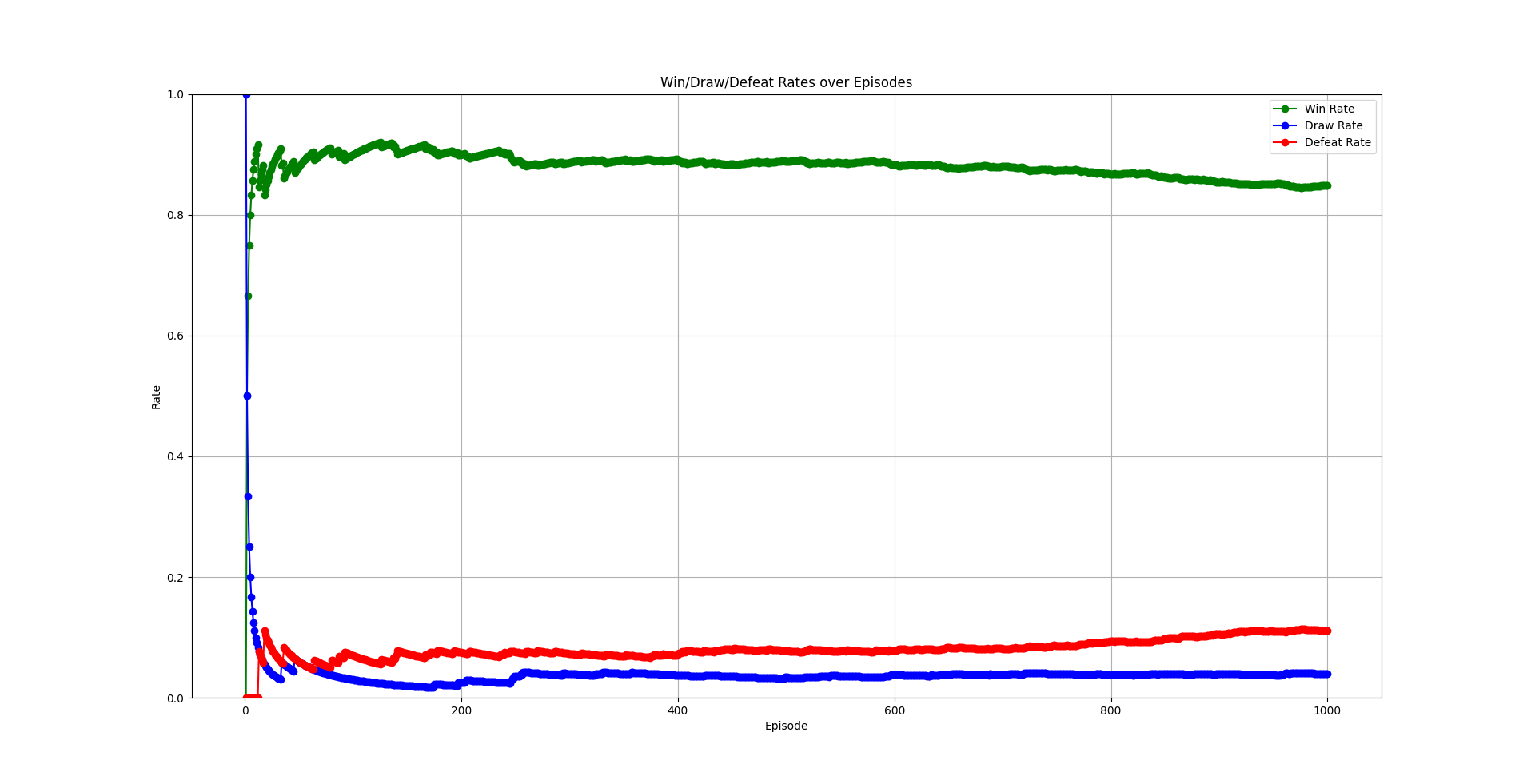
**[그림 13] score그래프**



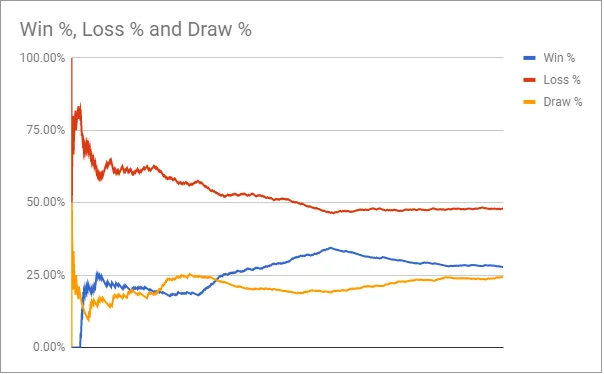
**[그림 14] score 그래프 마지막 200번 결과 그래프**

에이전트가 투배럭 빌드를 학습하는 것이 목표인데 투배럭 빌드의 경우 상대를 빠르게 제압하는 것이 특징이다. 즉 빠르게 끝내야 하기 때문에 점수가 일정 수준을 넘어가면 게임이 길어져 상대가 공격을 막아내고 다른 강한 유닛들로 공격한다는 것을 의미하므로 이 점수는 10000점을 넘지않는 편이 좋다. 그러나 만약 10000점이 넘을경우 상대가 대응하는것을 의미하기도 하지만 게임이 끝난것이나 마찬가지인 상황(상대방이 항복했거나 상대방의 모든 공격 유닛과 svc를 처치하고 건물만 남아있는 경우)을 인식하지못하고 무승부 판정이 날 때까지 게임을 지속하기 때문에 점수가 높아지는 경우도 있기 때문에 score그래프로 에이전트의 성능을 평가하기에는 무리가 있다.

그래서 승률 그래프로 평가하는 것이 에이전트의 성능을 평가하기에 가장 적합하다.

**[그림 15] 학습결과(초록색: 승률, 빨간색: 패배 비율, 파란색: 무승부 비율)**

본 논문의 에이전트의 결과이다. 학습결과 80%이상의 승률을 나타내는 것을 확인할 수 있었다. 패배 비율이 회를 거듭할 수록 증가한 값으로 수렴하긴 했지만 10%대로 낮다는 것을 확인 할 수 있다.



**[그림 16] full-game 학습(Steven Brown,** [**https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255**](https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255)**)**

2장에서 나왔던 기존의 학습결과의 경우 50%정도의 승률을 나타내는데 본 논문의 경우 80% 이상의 높은 승률을 나타내는 것을 알 수 있다. 이를 통해 스타크래프트2같은 상태 값이 다양하고 복잡한 경우에는 Q-Learning보다 DQN으로 학습을 진행시켜야 더 높은 승률을 얻을 수 있다는 것을 확인할 수 있다.

**제 5 장 한계 및 향후 개선 방안**

현재 에이전트의 가장 큰 한계는 상대방의 항복을 인식하지 못한다는 점이다. 현재 pysc2라이브러리에서는 기본적으로 승리 판정을 게임이 종료되고 나서 나오는 결과로 판정한다. 즉 게임이 일단 종료되어야 한다는 것이다. 게임이 종료되기 위한 조건은 게임이 강제 종료되거나 상대방 혹은 아군이 항복하고 그 항복이 승인되거나 한쪽이 엘리미네이션(건물이 모두 파괴되어 패배한 것으로 판정되는 상황[11]. 흔히 ‘엘리’라고 함)일 경우 게임이 끝난다. 하지만 pysc2라이브러리로 설계된 에이전트의 경우 항복을 인식하지 못하기 때문에 게임이 종료되는 조건은 무승부이거나 둘 중 하나가 엘리인 경우밖에 존재하지 않는다. 그렇기 때문에 상대방의 모든 유닛을 파괴하고 상대방이 더 이상 어떤 행동을 할 수 없는 상황이 되어 항복을 요청하더라도 이를 승인할 수 없기 때문에 에이전트는 게임을 계속 이어 나가게 된다. 그렇게 게임을 진행하다 상대방의 마지막 남은 건물을 파괴하지 못하면 시간 초과로 인해 무승부로 게임이 종료되는 경우도 종종 있고 초반에 강하게 상대를 몰아붙여 항복을 받아내더라도 승인을 하지 않고 게임을 계속 하다가 역전을 당해 패배하는 경우도 종종 있다. 물론 항복 요청을 수락하는 버튼이 나타나는 곳은 고정되어 있으나 항복 승인 버튼이 나타나지 않았을 경우엔 다른 유닛이나 건물을 선택하게 되어 게임을 방해하게 되고 에이전트가 행동을 하는데 방해하게 된다. 이런 이유로 일정 위치를 주기적으로 클릭하게 하는 방법은 바람직하지 못하기 때문에 pysc2라이브러리에서 업데이트가 진행되지 않는 이상 개선하기가 어려운 상황이다. 만약 항복을 승인할 수 있도록 개선될 경우 지금보다 승률이 더 올라갈 것이라고 예측이 가능하다. 그리고 더 다양한 건물 및 유닛을 활용 가능하도록 행동을 정의하고 학습시킬 경우 지금의 쉬운 난이도의 봇이 아닌 더 높은 수준의 봇 이를 넘어서 인간을 상대로도 높은 승률을 얻을 수 있을 것으로 기대된다.

**감사의 글**

먼저 본 논문을 작성하기까지 4년 동안 대학생활을 하면서 컴퓨터 공학에 대해 아무것도 모르던 저에게 많은 가르침을 주신 교수님들께 감사인사를 드립니다.

처음에 논문을 써야한다는 사실을 알게되었을때 너무 막막하고 할 수 있을까 하는 두려움이 컸지만 할 수 있도록 교수님들이 많은 가르침을 주시고 격려해 주셔서 논문을 완성할 수 있었습니다.

이번 논문을 쓰면서 제가 아직 많이 부족하고 배워야 할 내용이 많다는 것을 다시 한번 느끼게 되었고 이를 통해 더욱 성장하는 경험을 할 수 있었습니다. 비록 많이 부족한 논문이지만 지도해주신 이상정 교수님과 심사위원 홍인식 교수님께 다시한번 깊은 감사 인사를 드립니다.

**참 고 문 헌**

[1] 강화학습 나무위키 2023-05-28

<https://namu.wiki/w/%EA%B0%95%ED%99%94%ED%95%99%EC%8A%B5>

[2] pysc2 2022-7-13

<https://deepmind.google/discover/blog/deepmind-and-blizzard-open-starcraft-ii-as-an-ai-research-environment/>

[3] CNN구조를 이용한 Starcraft II 미니게임 모방학습 2019-12

<https://www-dbpia-co-kr-ssl.eproxy.sch.ac.kr:8443/journal/articleDetail?nodeId=NODE09301985>

[4] Build a Sparse Reward PySC2 Agent 2017-12-26

<https://itnext.io/build-a-sparse-reward-pysc2-agent-a44e94ba5255>

[5] Starcraft II 2023-11-19

<https://namu.wiki/w/%EC%8A%A4%ED%83%80%ED%81%AC%EB%9E%98%ED%94%84%ED%8A%B8%202>

[6] 빌드 오더 2023-10-18

<https://namu.wiki/w/%EB%B9%8C%EB%93%9C%20%EC%98%A4%EB%8D%94>

[7] 강화학습 에이전트 2023-07-28

<https://www.thedatahunt.com/trend-insight/reinforcement-learning>

[8] Q 학습 2022-02-27

<https://ko.wikipedia.org/wiki/Q_%EB%9F%AC%EB%8B%9D>

[9] DQN 2023-07-02

<https://namu.wiki/w/DQN#s-2>

[10] 경사 하강법 2022-11-24

<https://ko.wikipedia.org/wiki/%EA%B2%BD%EC%82%AC_%ED%95%98%EA%B0%95%EB%B2%95>

[11] 엘리전 2023-09-21

<https://namu.wiki/w/%EC%97%98%EB%A6%AC%EC%A0%84>