

|  |
| --- |
|  |
| 게임 인공지능 – Report #2  2018112128 박준서  정보통신공학과 |
|  |

<목차>

**I. 게임 주제 및 환경 설명** 3

1. 게임 주제 3

2. 환경 설명 3

**II. 학습 알고리즘 설명 및 실험 내용** 4

1. 알고리즘 4

2. 실험 내용 6

**III. 실험 결과** 13

1. DQN 13
2. A2C 21

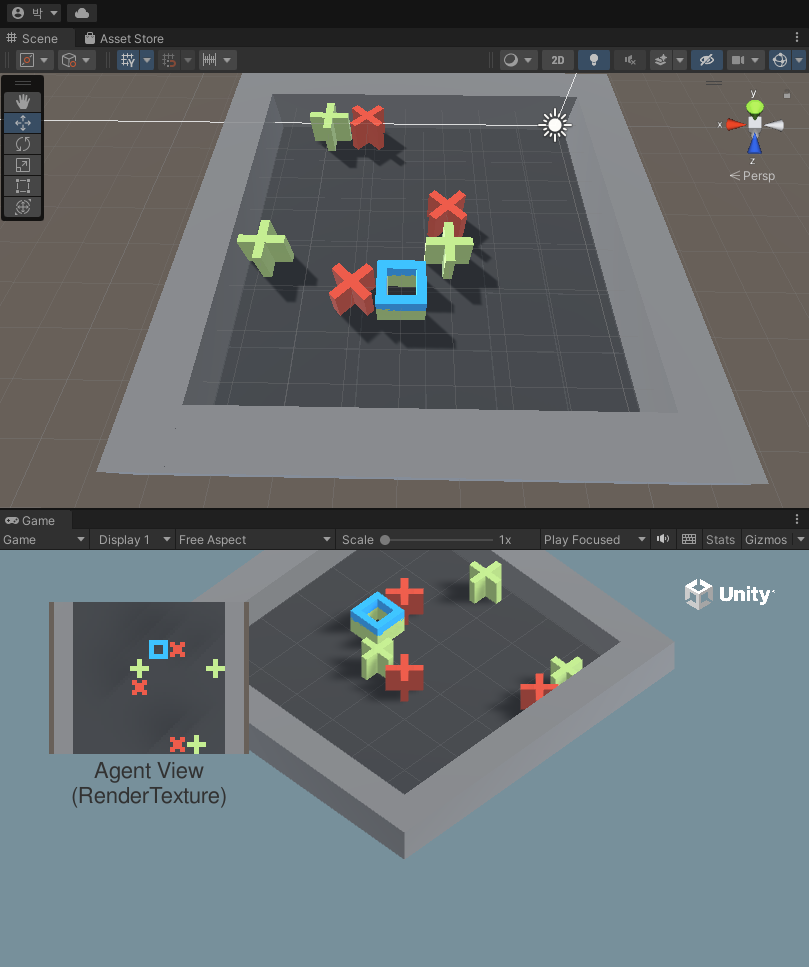
**IV. 종합결과 분석 및 결론** 28

1. **게임 주제 및 환경 설명**
2. 게임 주제

게임인공지능-Report #1에서 진행했던 Grid World 15 by 15를 변경하여 8 by 8로 진행한다. Plus & Ex는 각각 3개씩 생성한다. 게임의 목적은 Agent가 특정 목적지에 도달하는 것이다. 이때, 강화학습의 알고리즘인 DQN과 A2C를 통하여 Agent를 학습시키되, 다양한 hyperparameter를 조절함으로써 실험 결과를 확인한다.

1. 환경 설명

화면으로 보는 구성은 다음과 같다.



Gird의 크기가 8 by 8이고, 목적지인 Plus와 Ex가 각각 3개씩 존재하는 걸 확인할 수 있다.

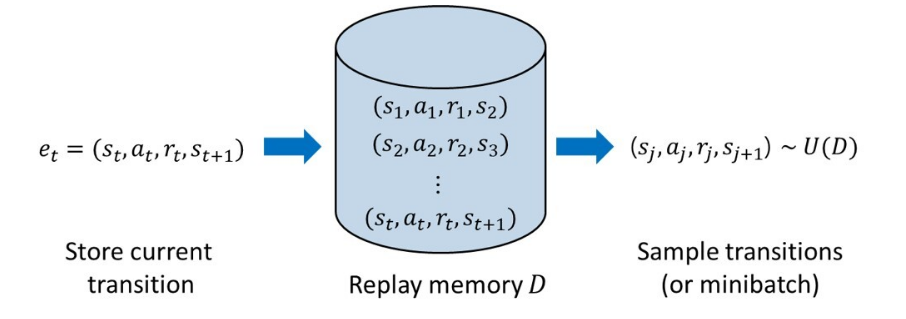
1. **학습 알고리즘 설명 및 실험 내용**
2. 알고리즘
3. DQN

가치 기반 강화학습, 정책 기반 강화학습 중 가치 기반 강화학습에 속한다. Q 함수가 최적의 Q 함수 값을 도출할 수 있도록 학습하고 해당 Q 함수 값을 통해 의사결정을 수행한다. 기존의 Q-learning은 Q 함수 값을 table 형식으로 저장하여 학습한다. 이러한 방식은 state space & action space가 커지게 되면, 모든 Q 함수 값을 저장하기에는 많은 memory와 긴 exploration time이 필요하다. 이를 deep-learning으로 해결하기 위해 DQN이 등장했다.

하지만 deep-learning을 통해 Q 함수 값을 근사하는 방법들은 학습의 불안정성이 존재했다. 따라서 두 가지의 기법을 추가한다.

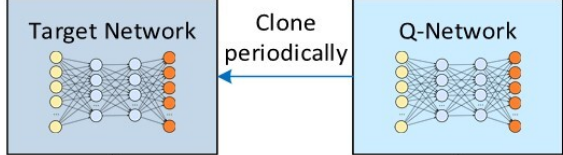
1. Experience replay

Deep-learning에서는 학습 샘플들이 independent하게 추출되었다고 가정하고 모델을 학습한다. 하지만 강화학습은 sequential한 data이기 때문에 이들 사이에 dependency가 존재한다. 즉, correlation이 강한 샘플들로 학습하면 올바른 data distribution을 바탕으로 학습할 수 없게 된다. 이를 해결하기 위해 등장한 것이 experience replay이고 방법은 매우 간단하다. Replay buffer라는 버퍼를 하나 만들어줘서 샘플들을 저장시켜주는 것이다. 학습 시에는 단순하게 버퍼에서 mini-batch만큼 샘플들을 뽑아 학습시켜주면 샘플들 간의 correlation을 해결시켜 줄 수 있을 뿐 더러, 학습의 안정성을 더해줄 수 있다.



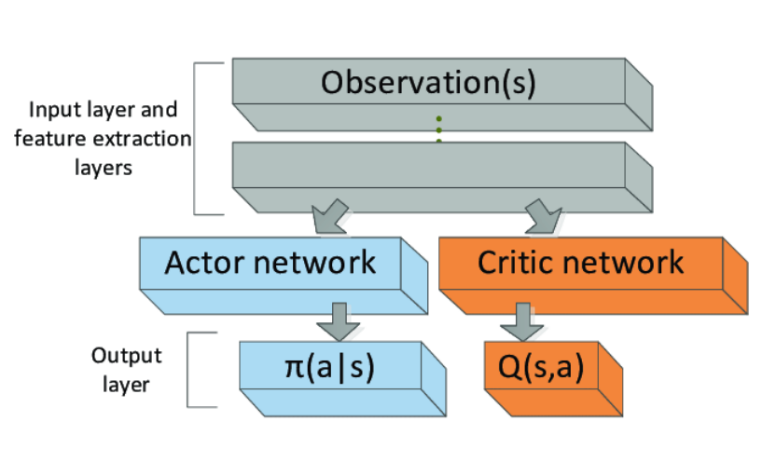
1. Target network

Target network는 기존의 Q-network를 그대로 복제하여 target value를 고정시켜 놓는 형태이다. 기존의 학습 방식 그대로 진행하면 target value의 움직임이 크기 때문에 학습이 진행되지 않았는데, target value를 고정시킴으로써 학습의 안정성을 높일 수 있었다. 이때, network의 복제 시기는 hyperparameter이다.



본 프로젝트에서는 합성곱 신경망(CNN)을 활용하여 각 행동의 Q 함수 값들은 근사한다.

1. A2C

가치 기반 강화학습, 정책 기반 강화학습 중 정책 기반 강화학습에 속한다. 정책 기반 강화학습 알고리즘 중에는 REINFORCE 알고리즘이 존재한다. 해당 알고리즘은 return G라는 value를 구하기 위해 한 episode가 끝나길 기다렸다가 학습을 진행해야 하는 반면, 본 프로젝트에서 소개하는 A2C 알고리즘은 return G가 아닌 Q 함수 값을 또 다른 network로 근사하기 때문에, episode가 끝나지 않아도 학습이 가능하다. 따라서 Actor network와 Critic network라는 두 개의 network를 사용한다.

1. 실험 내용

본 프로젝트의 실험은 8 by 8의 Grid World에 DQN과 A2C 알고리즘을 적용시킨 뒤, hyperparameter를 다양하게 조절함으로써 최대한의 성능을 이끌어내는 model을 찾는 것이다.

실험 내용은 다음과 같다.

1. DQN
2. Original

* 교재에서 제공하는 python script의 hyperparameter를 바꾸지 않은 실험.   
  즉 기본값

|  |  |
| --- | --- |
| Batch\_size | * 32 |
| Mem\_maxlen | * **10000** |
| Discount\_factor | * **0.9** |
| Learning\_rate | * **0.00025** |
| Run\_step | * **50000** |
| Train\_start\_step | * **5000** |
| Target\_update\_step | * **500** |
| Epsilon\_eval | * **0.05** |
| Epsilon\_init | * **1.0** |
| Epsilon\_min | * **0.1** |
| Explore\_step | * **Run\_step \* 0.8** |
| Eplsilon\_delta | * **(epsilon\_init\_epilon\_min)/explore\_step** |
| 학습 시간 | * **8.523min** |

1. Experiment 1

기본값에서 Run\_step만 50000 -> 150000으로 올렸다.

|  |  |
| --- | --- |
| Run\_step | * 150000 |
| 학습 시간 | * **18.52min** |

1. Experiment 2

Experiment 1에서 mem\_maxlen을 20000으로 늘렸다.

|  |  |
| --- | --- |
| Mem\_maxlen | * 20000 |
| Run\_step | * **150000** |
| 학습 시간 | * **38.63min** |

1. Experiment 3

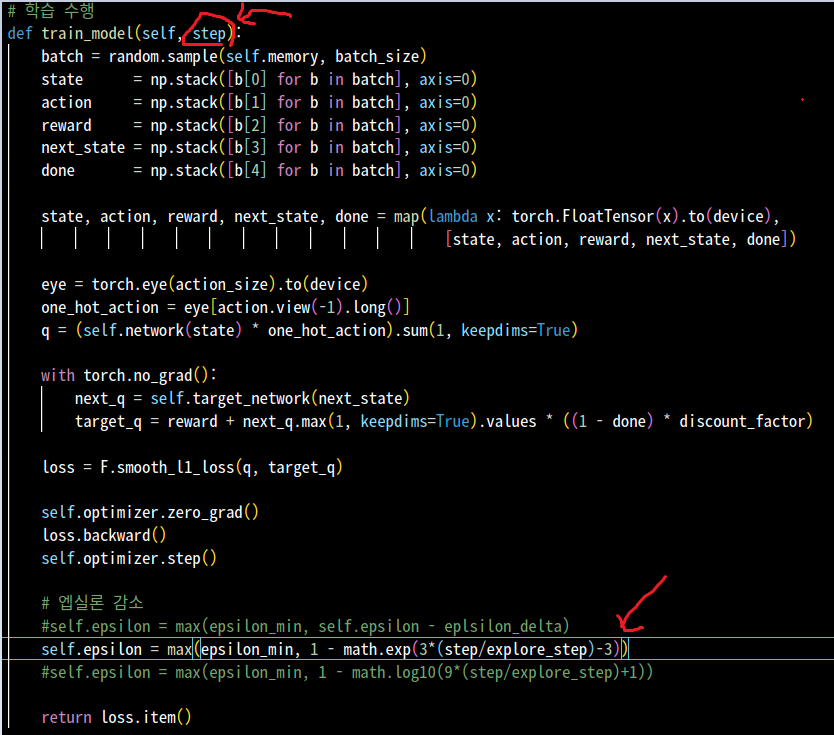
|  |  |
| --- | --- |
| Mem\_maxlen | * 20000 |
| Run\_step | * **150000** |
| Train\_start\_step | * **10000** |
| Target\_update\_step | * **1000** |
| Epsilon\_min | * **0.05** |
| 학습 시간 | * **18.34min** |

1. Experiment 4

|  |  |
| --- | --- |
| Run\_step | * 300000 |
| 학습 시간 | * **33.51min** |

1. Experiment 5

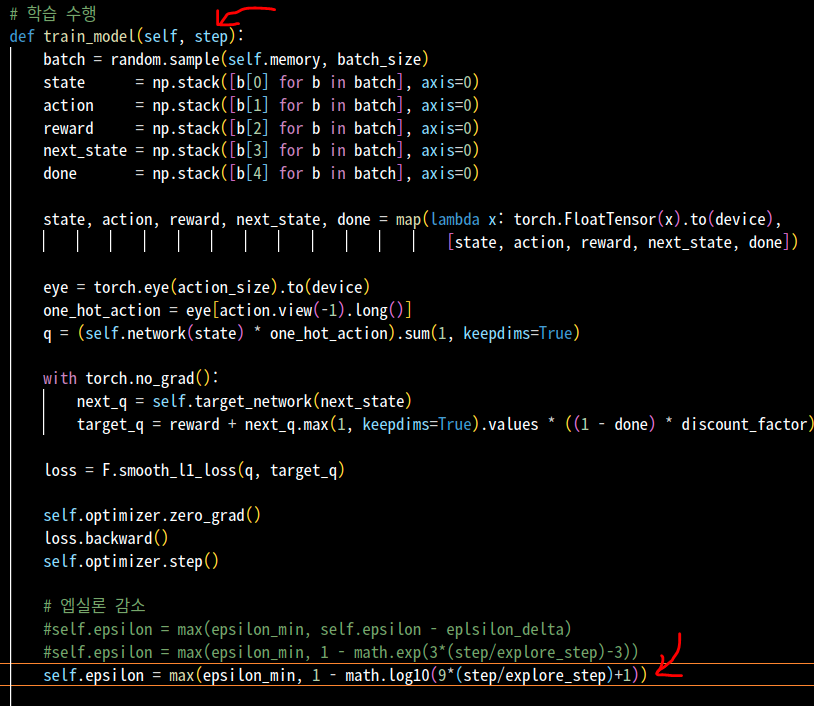
Eplision의 감소를 linear decay에서 지수 감소로 바꾸기 위해 eplsilon\_delta를 제거하고 eplsilon을 새로 정의했다.



|  |  |
| --- | --- |
| Mem\_maxlen | * 20000 |
| Run\_step | * **300000** |
| Target\_update\_step | * **1000** |
| Epsilon\_min | * **0.05** |
| Eplsilon\_delta | * **X** |
| 학습 시간 | * **32.86min** |

1. Experiment 6

Experiment 5와 동일해 보이지만, Eplsilon의 감소를 다른 지수 감소로 정의를 했다.



|  |  |
| --- | --- |
| Mem\_maxlen | * 20000 |
| Run\_step | * **300000** |
| Target\_update\_step | * **1000** |
| Epsilon\_min | * **0.05** |
| Eplsilon\_delta | * **X** |
| 학습 시간 | * **34.1min** |

1. A2C

A2C의 경우 python script의 state size가 6\*2로 설정되어 있다. 본 프로젝트에서 실험하는 환경인 Grid World의 경우 Goal의 개수가 총 2개가 아닌 6개로 변경되었으므로, state size를 14\*2로 설정해줘야 한다. 마찬가지로 unity에서 build를 할 때에 vector observation의 space size를 14로 설정해줘야 한다.

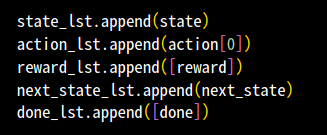


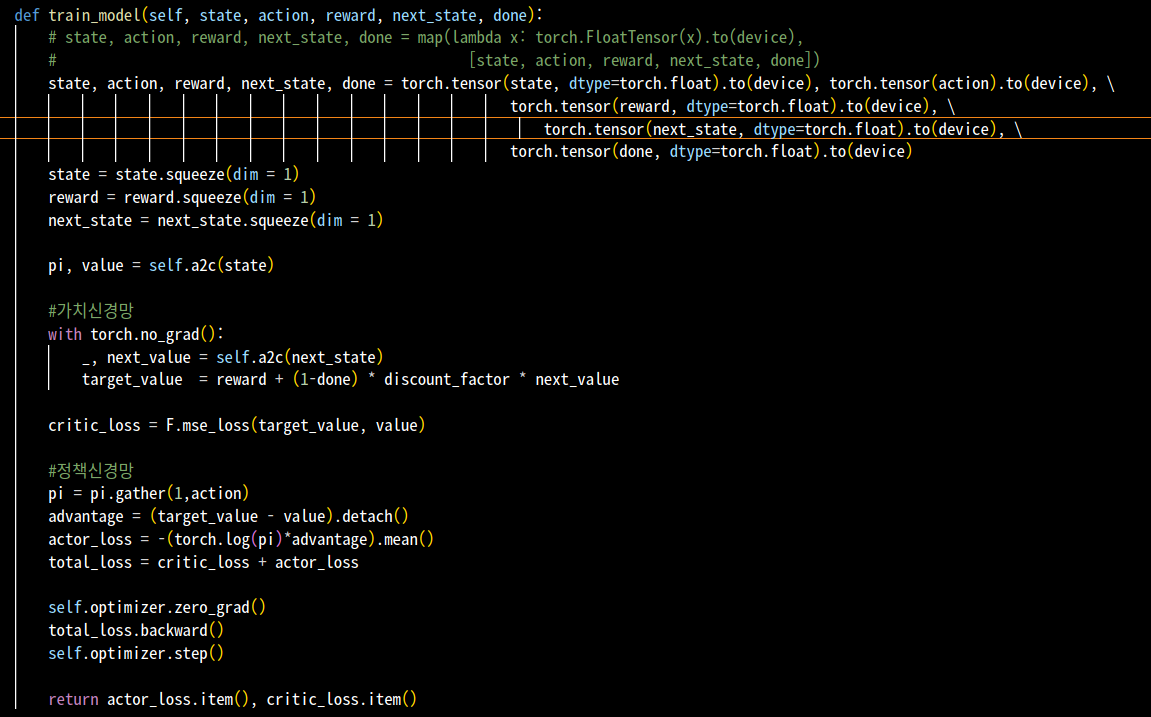


또한, 해당 A2C python script에선 batch size가 존재하지 않는다. Environment가 커질 경우 batch size가 큰 영향을 끼칠 거라고 예상하기 때문에 코드를 수정하여 batch size를 입력할 수 있도록 했다. 코드는 다음과 같이 수정했다.









1. Original

|  |  |
| --- | --- |
| Batch\_size | * 1 |
| Discount\_factor | * **0.9** |
| Learning\_rate | * **0.00025** |
| Run\_step | * **50000** |
| Num\_layers | * **2(pi, v 제외)** |
| 학습 시간 | * **15.54min** |

1. Experiment 1

|  |  |
| --- | --- |
| Run\_step | * 150000 |
| 학습 시간 | * **16.95min** |

1. Experiment 2



|  |  |
| --- | --- |
| Run\_step | * 150000 |
| Num\_layers | * **4(pi, v 제외)** |
| 학습 시간 | * **17.13min** |

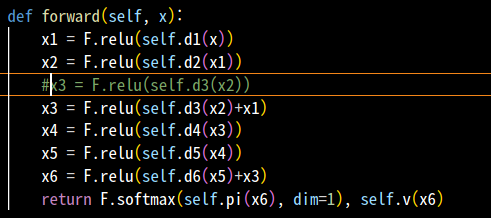
1. Experiment 3

|  |  |
| --- | --- |
| Batch\_size | * 10 |
| Run\_step | * **100000** |
| 학습 시간 | * **1.433hr** |

1. Experiment 4



|  |  |
| --- | --- |
| Batch\_size | * 32 |
| Learning\_rate | * **0.000125** |
| Run\_step | * **100000** |
| Num\_layers | * **4(pi, v 제외)** |
| 학습 시간 | * **2.475hr** |

1. Experiment 5

|  |  |
| --- | --- |
| Batch\_size | * 32 |
| Run\_step | * **100000** |
| Num\_layers | * **6(pi, v 제외) + skip connection** |
| 학습 시간 | * **2.542hr** |

1. Experiment 6

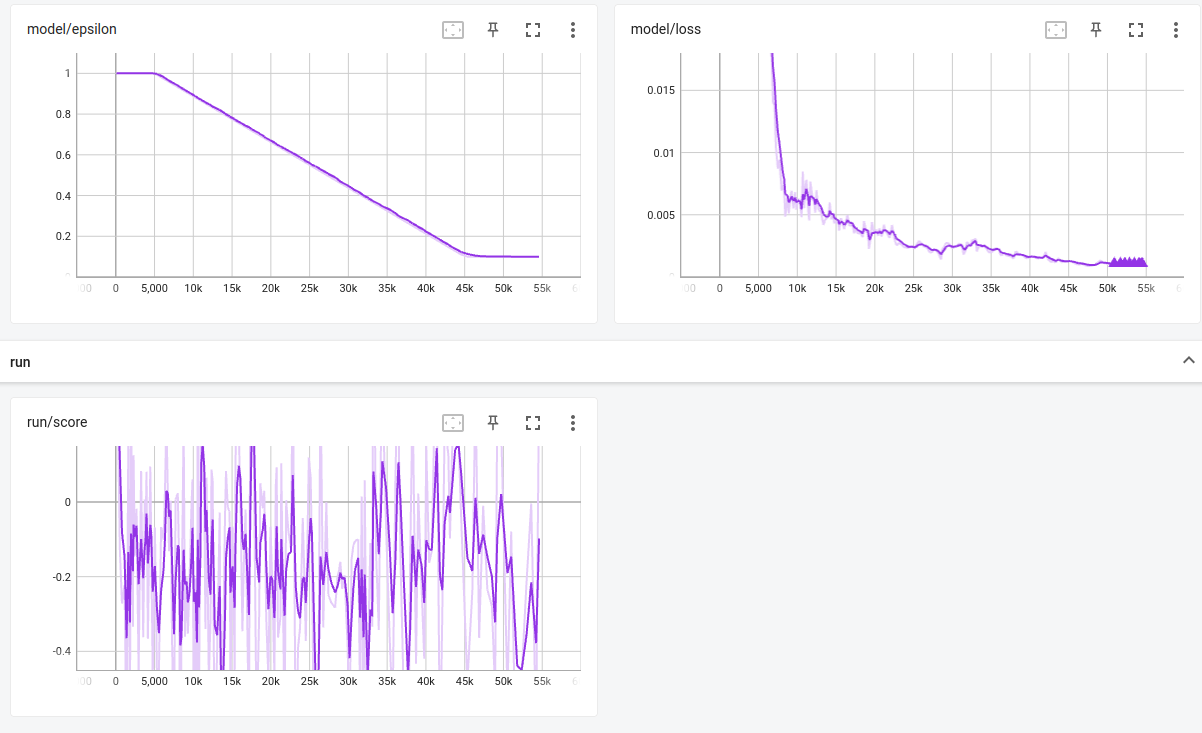


|  |  |
| --- | --- |
| Batch\_size | * 32 |
| Run\_step | * **150000** |
| Num\_layers | * **4(pi, v 제외)** |
| 학습 시간 | * **5.088hr** |

1. **실험 결과**

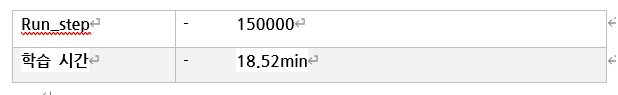
8 by 8 Grid World에 대해서 2가지 알고리즘(DQN & A2C)을 다양하게 바꿔가면서 실험을 했다. 실험에 대한 기본 설정은 [실험 내용] 부분에서 자세하게 기술했으므로 간략하게만 언급한다.

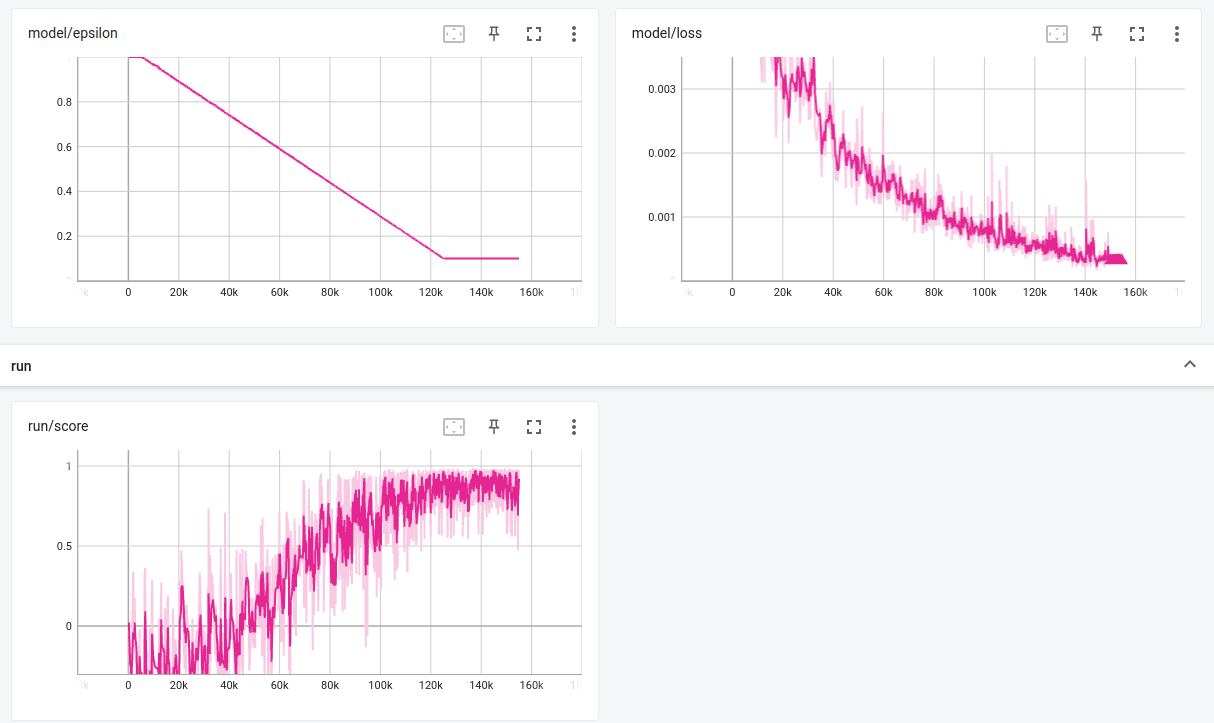
1. DQN
2. Original



* Epsilon이 linear하게 줄어든다.
* Loss가 일정하게 줄어든다.
* Score가 0을 넘기지 않는 경우가 많고 학습이 불안정하다.

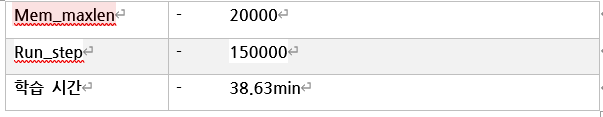
1. Experiment1

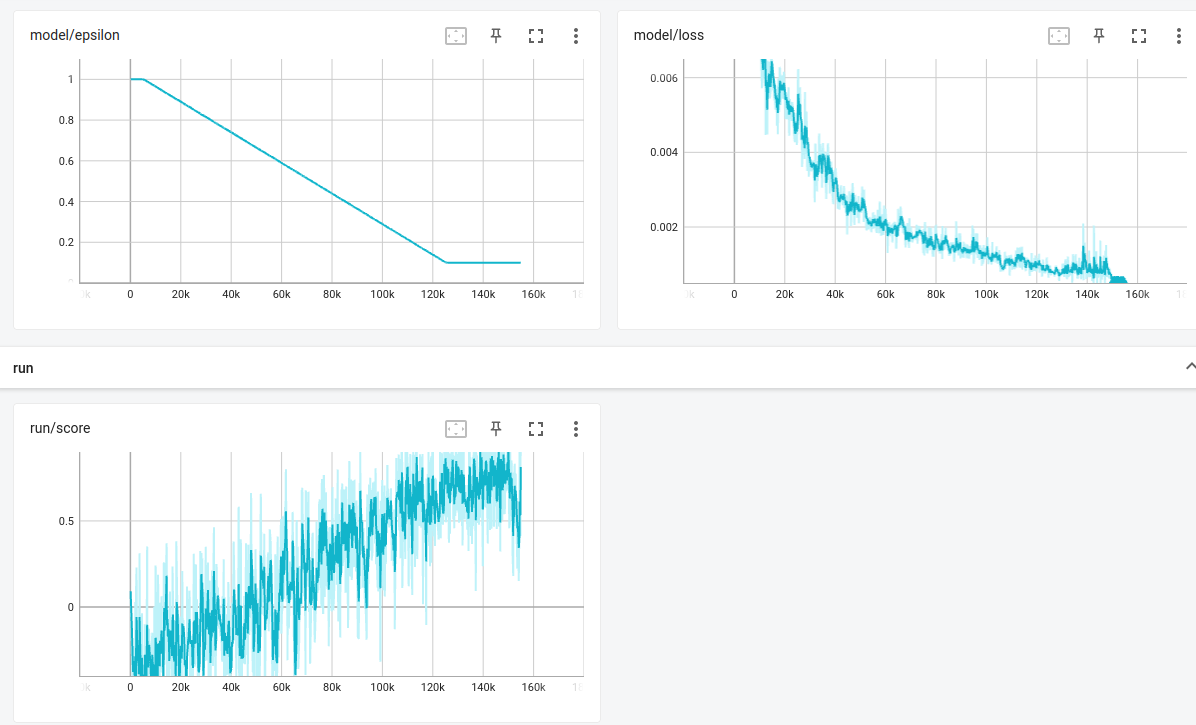




* Epsilon은 linear하게 줄어든다.
* Loss가 일정하게 감소하는 걸 볼 수 있다.
* Score가 1에 근접하게 올라간다.

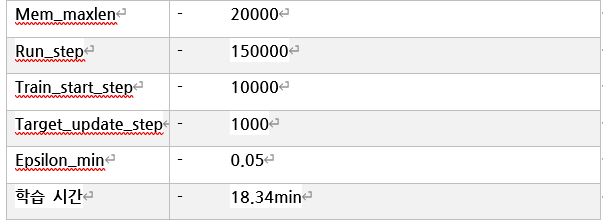
1. Experiment 2

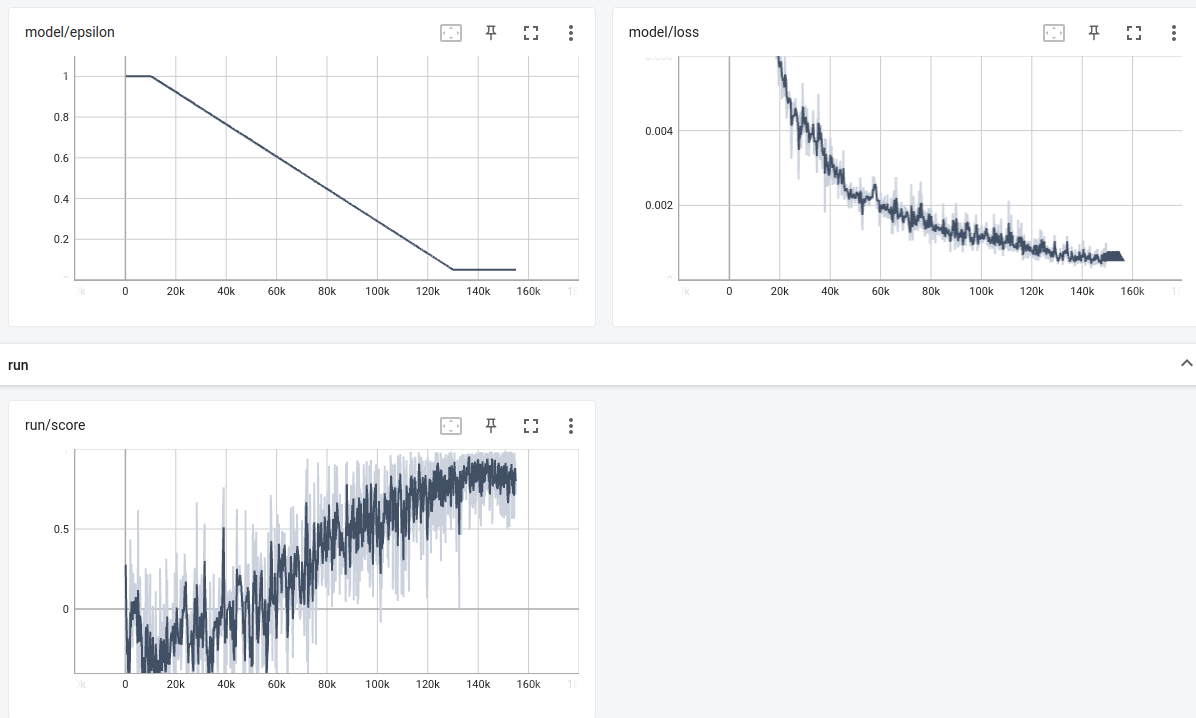




* Epsilon이 linear하게 줄어든다.
* Loss가 일정하게 줄어든다.
* Score가 1에 근접하게 가지만 약간의 불안정함이 존재

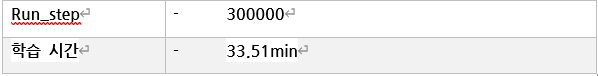
1. Experiment 3

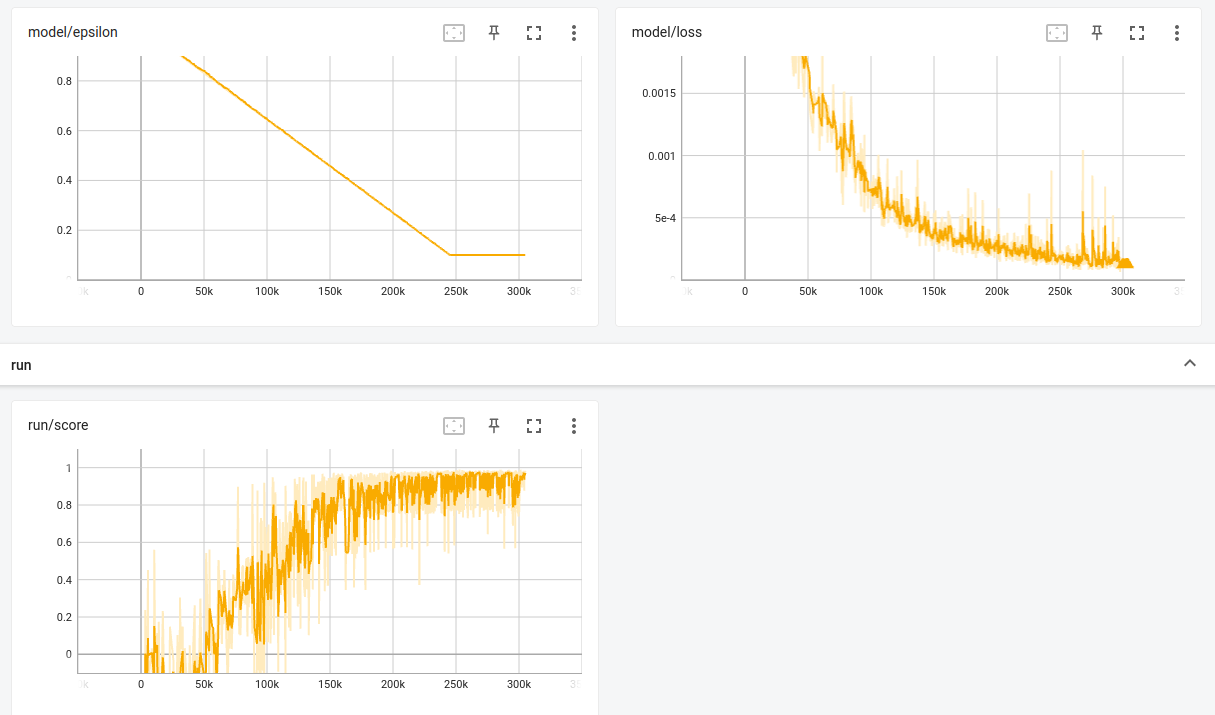




* Epsilon이 linear하게 줄어든다.
* Loss가 일정하게 줄어든다.
* Score가 1에 근접하게 학습된다.

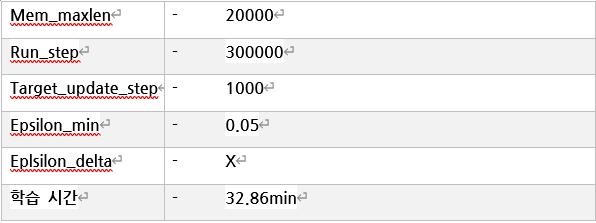
1. Experiment 4

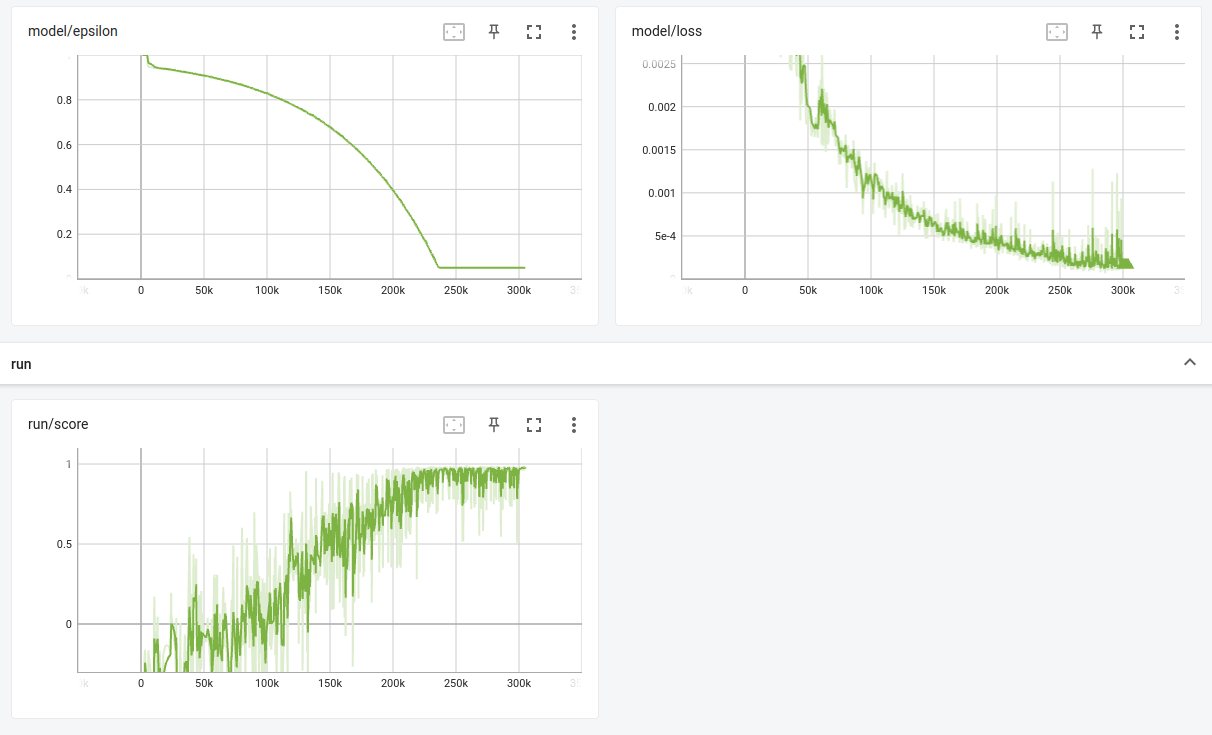




* Epsilon이 linear하게 줄어든다.
* Loss가 일정하게 줄어든다.
* Score가 1에 매우 근접하게 학습된다.

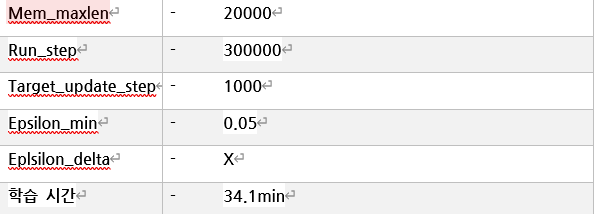
1. Experiment 5

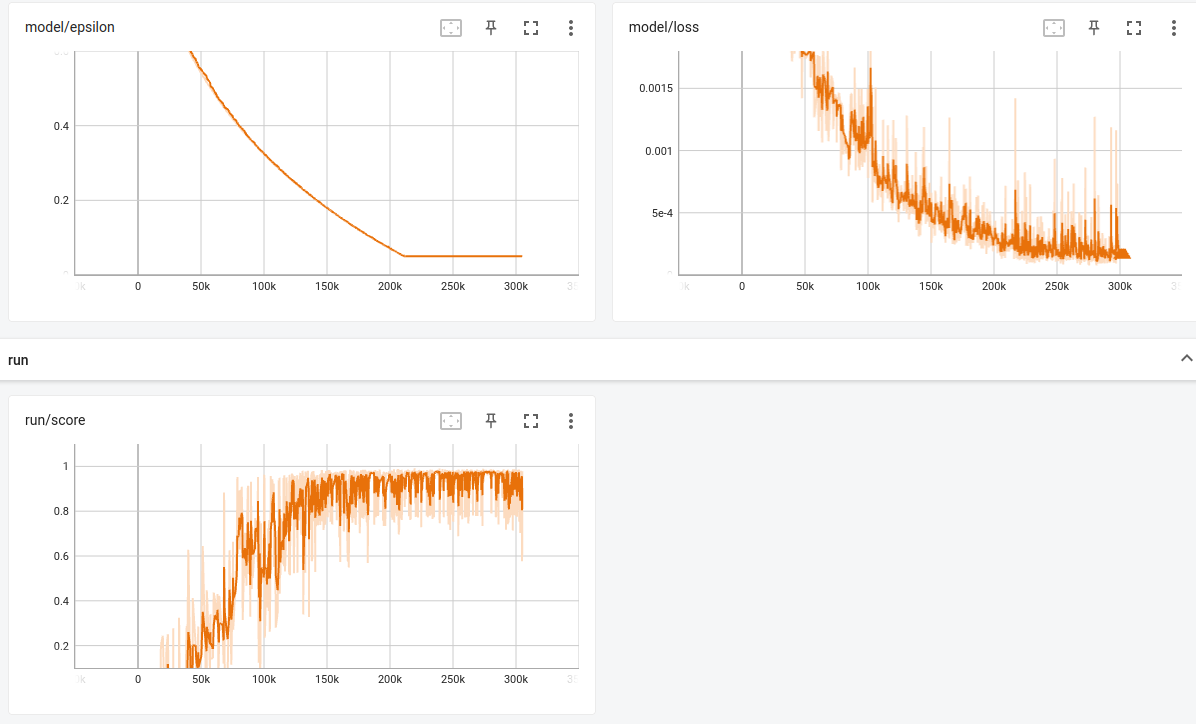




* Epsilon이 지수 형태로 감소한다.
* Loss가 일정하게 줄어든다.
* Score가 1에 매우 근접하게 학습된다.

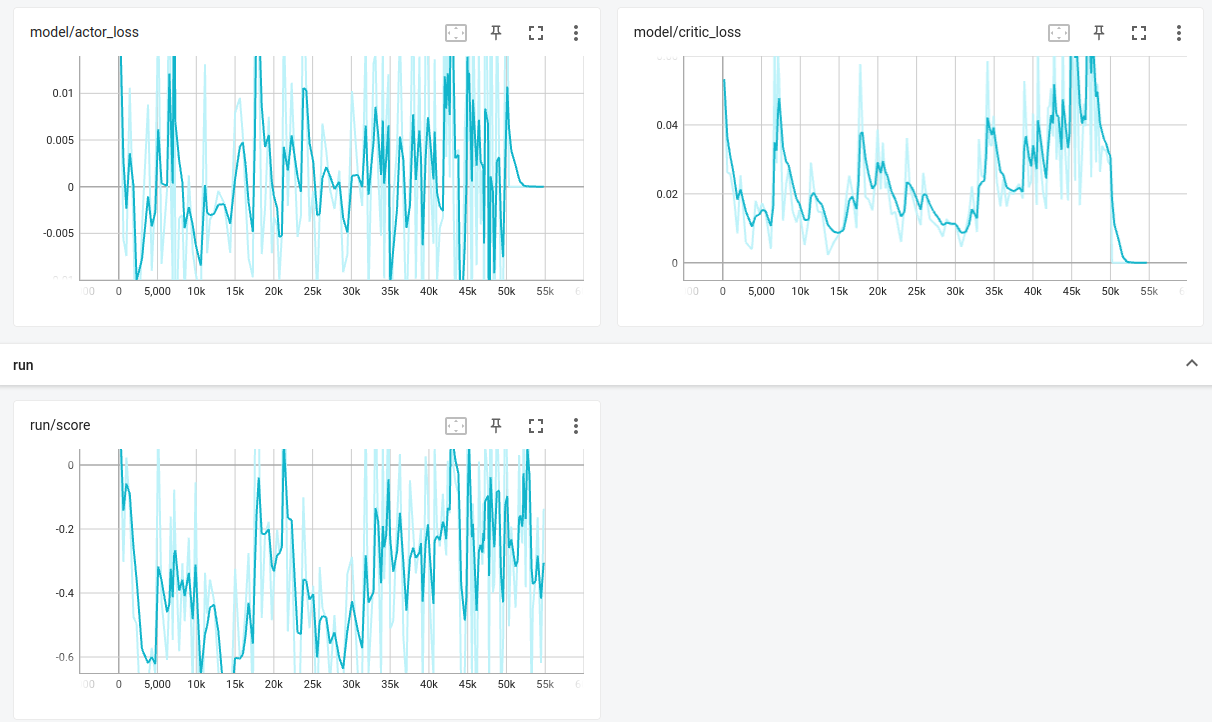
1. Experiment 6





* Epsilon이 지수 형태로 감소한다.
* Loss가 일정하게 줄어든다.
* Score가 1에 매우 근접하게 학습된다

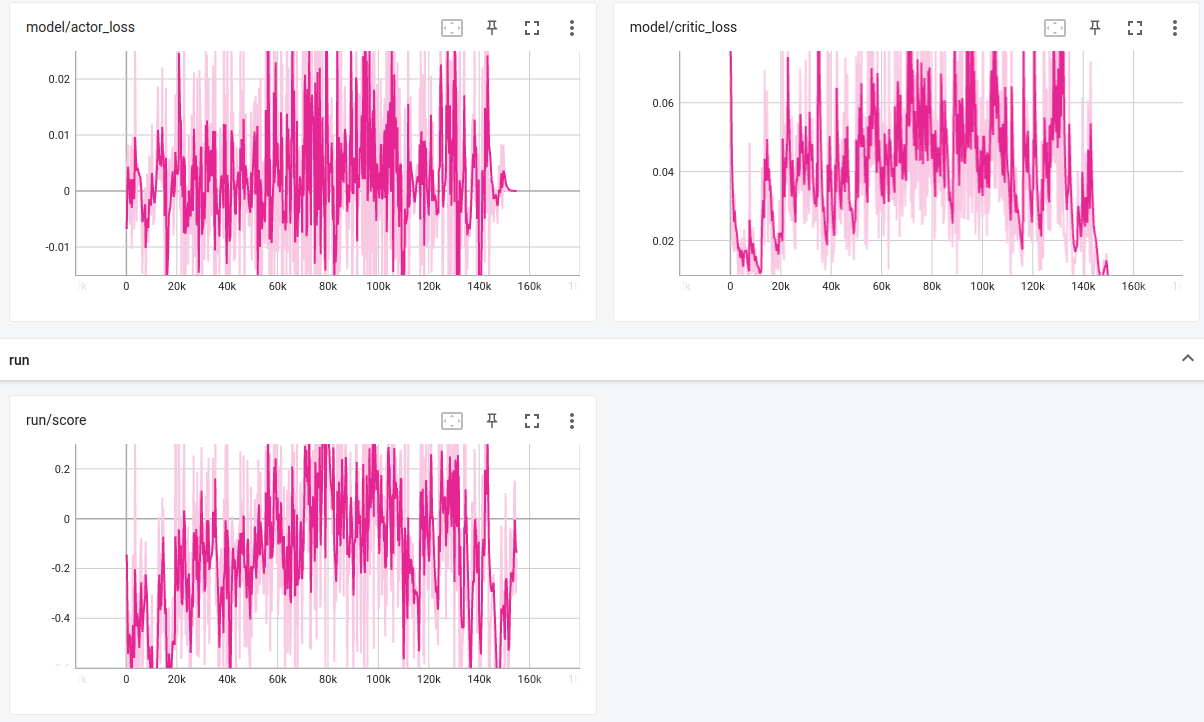
1. A2C
2. Original



* Loss가 불안정하게 학습된다.
* Score가 0 밑으로 떨어진다. 학습이 제대로 되지 않음

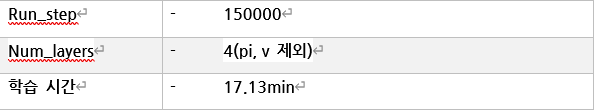
1. Experiment 1

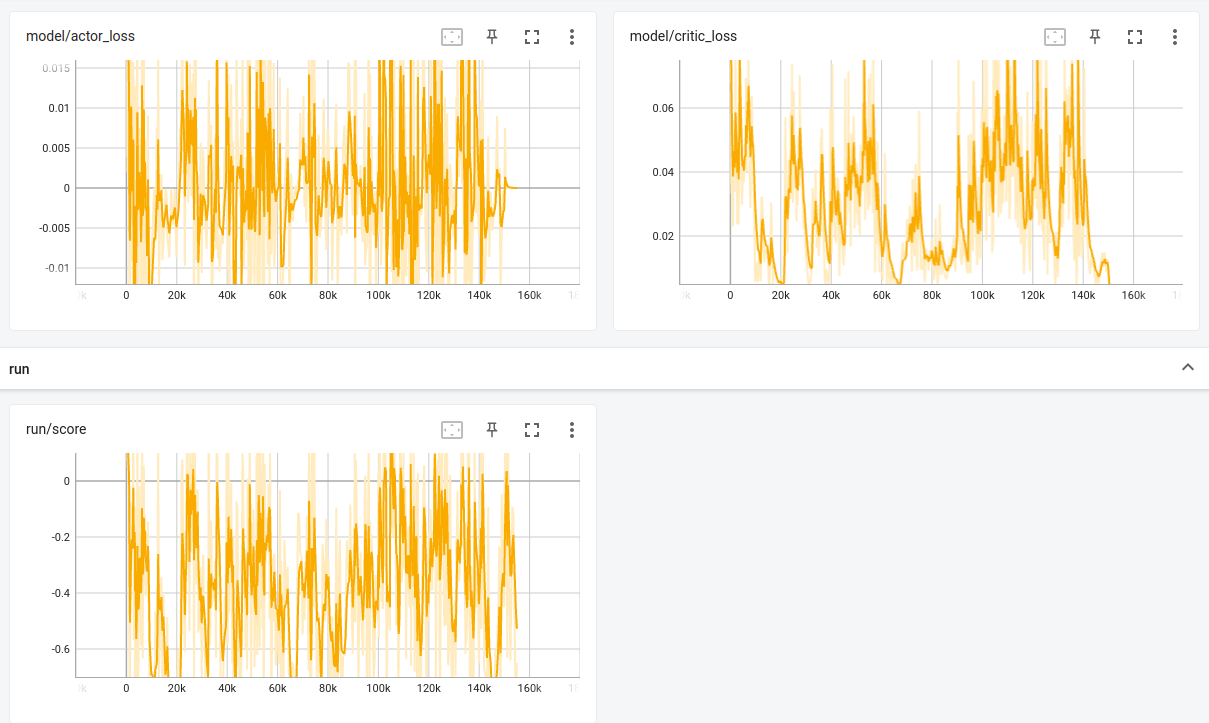




* Loss가 불안정하게 학습된다.
* Score가 0 밑으로 떨어지는 경우가 많다. 학습이 제대로 되지 않음

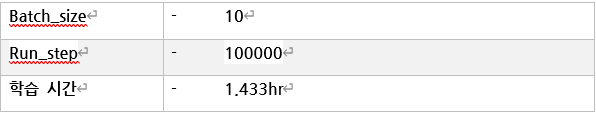
1. Experiment 2





* Loss가 불안정하게 학습된다.
* Layer 수를 늘렸음에도, score가 0 밑으로 떨어진다.

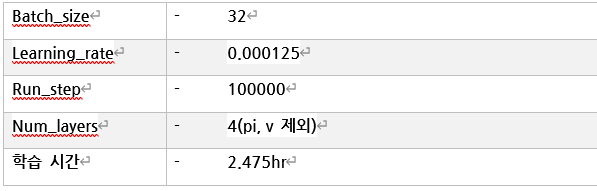
1. Experiment 3





* Actor loss가 불안정하게 떨어진다.
* Critic loss는 오히려 올라가는 걸 볼 수 있다.
* Score의 관점에선 0.6 이상을 획득하는 걸 볼 수 있다.
* Batch size를 추가함으로써 학습의 효율성이 향상된 걸 볼 수 있다.

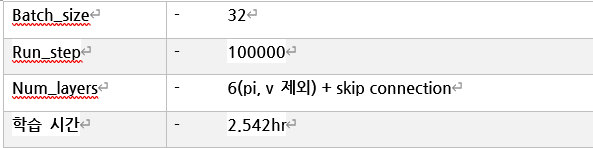
1. Experiment 4

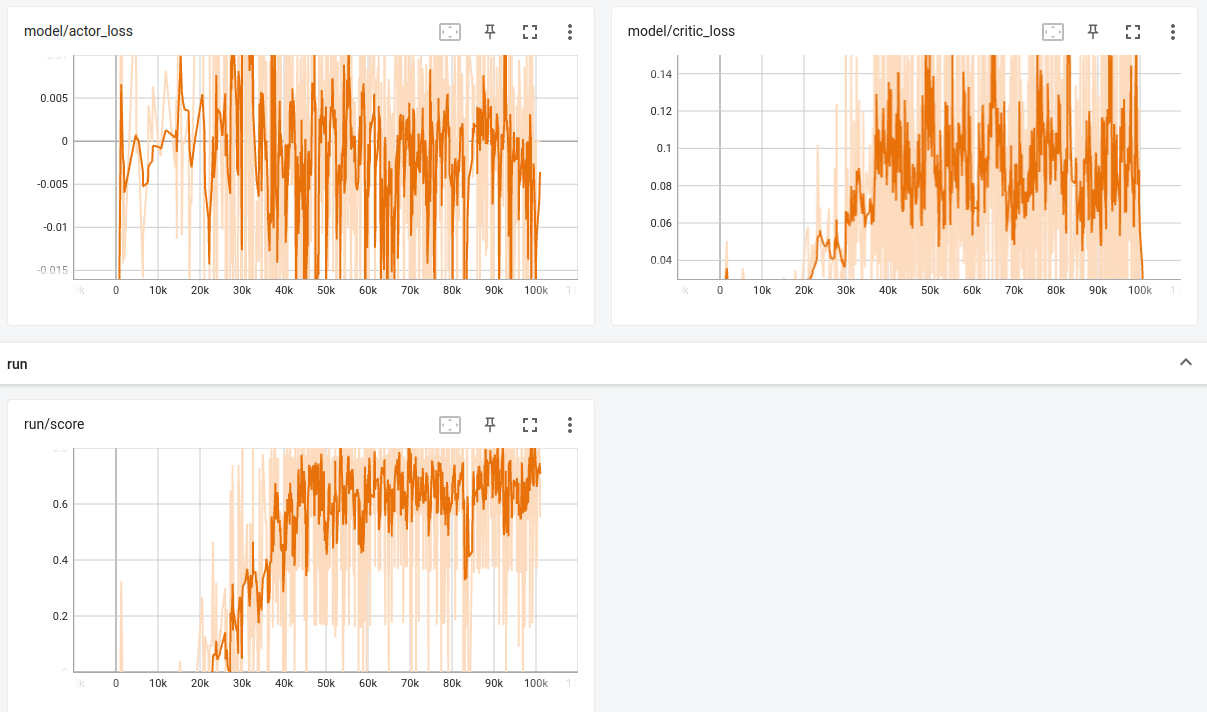




* Loss가 불안정하게 떨어지는 걸 볼 수 있다.
* Score 관점에선 0.7 이상을 획득함으로써 batch size를 더 늘리니 효과를 봤다.

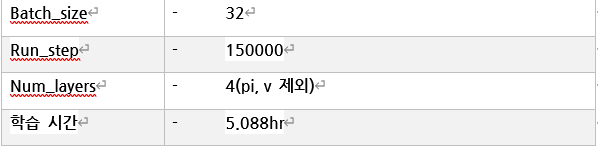
1. Experiment 5

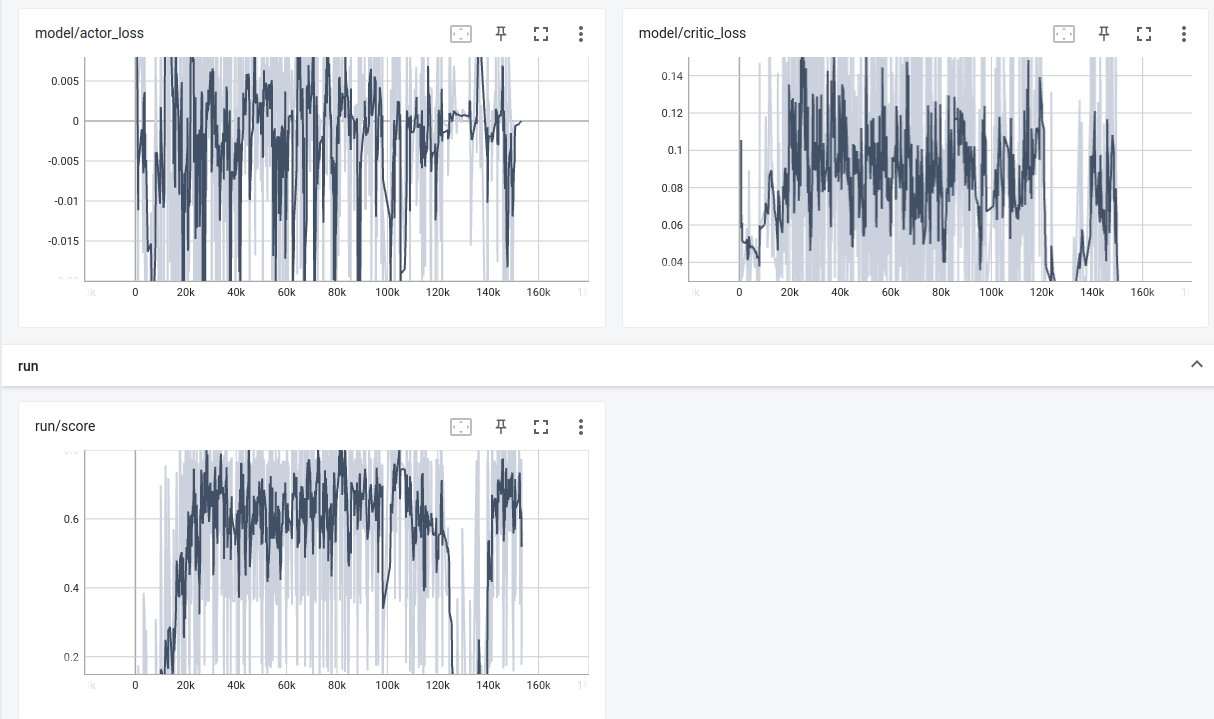




* Actor loss가 불안정하게 떨어진다.
* Critic loss는 오히려 올라가는 걸 볼 수 있다.
* Score의 관점에선 0.6 이상을 획득하는 걸 볼 수 있다.

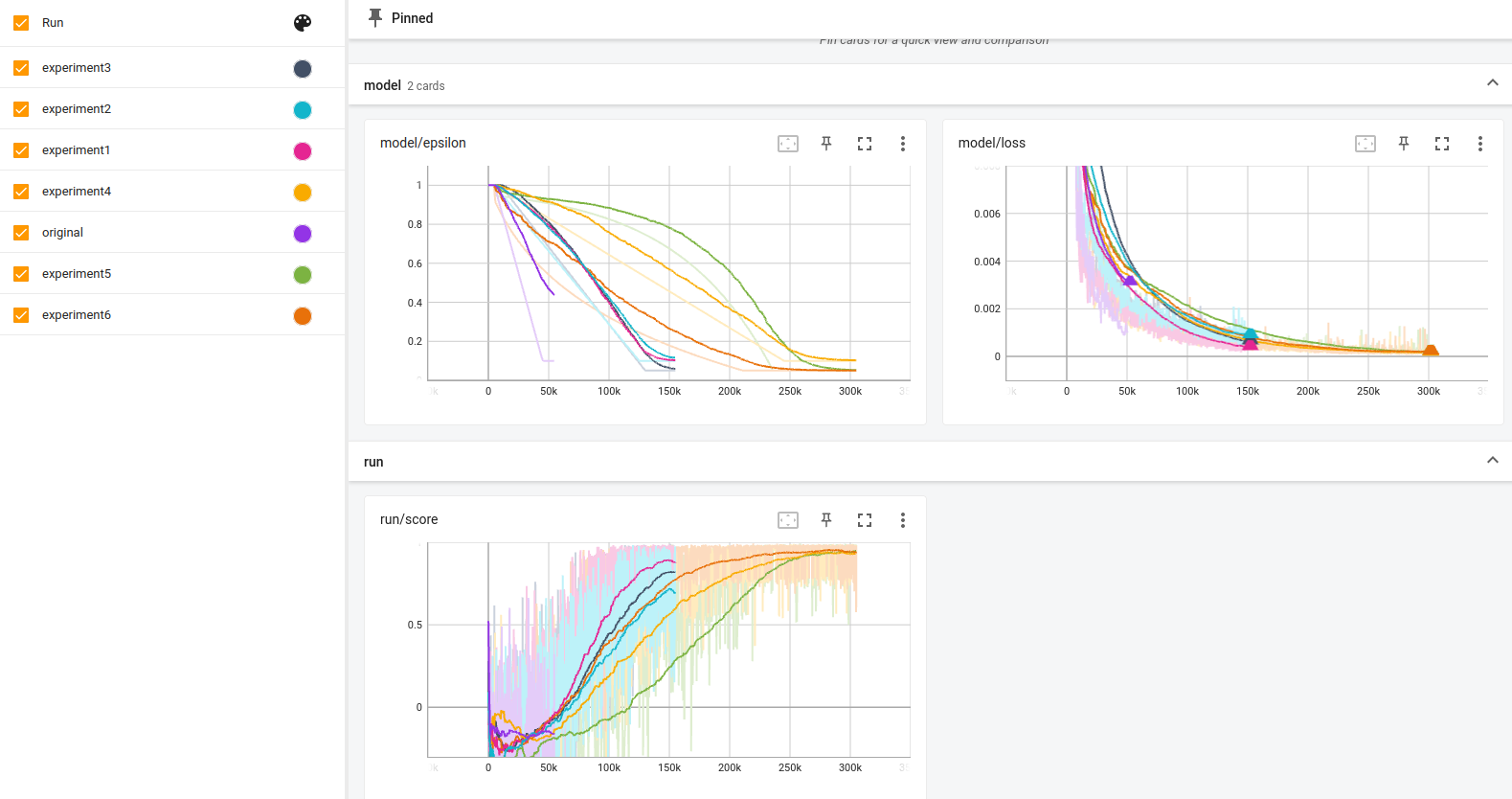
1. Experiment 6





* Actor loss가 불안정하게 떨어진다.
* Critic loss는 오히려 올라가는 걸 볼 수 있다.
* Score의 관점에선 0.6 이상을 획득하는 걸 볼 수 있다.

1. **종합 결과 분석 및 결론**

먼저 DQN의 종합적인 결과를 보자면 다음과 같다. 

해당 지표를 보고 판단하길,   
original -> experiment 2 -> experiment 1 -> experiment 3 -> experiment 4 -> experiment 5 -> experiment 6 순으로 성능이 올라간다. 그 이유로는 첫째, run step이 증가해야 한다. 학습을 오래 할수록 model의 정교함이 올라가는 것으로 보인다. 또한 target\_update\_step을 500에서 1000으로 바꿔줌으로써 update 주기를 늘려주어 안정성을 추가해줬다. 세 번째로는 epsilon의 감소 경향이다. 지수 감소로 바꾼 후로 훨씬 안정적이고 성능도 올라간 것을 확인할 수 있었다.

A2C의 종합적인 결과는 다음과 같다.



대체로 모든 experiment에서 학습이 불안정하지만 그럼에도 좋은 성능을 낸 실험들이 존재한다.   
Original -> experiment 2 -> experiment 1 -> experiment 3 -> experiment 4 -> experiment 5 & 6 순으로 성능이 증가한다.

학습의 성능이 올라간 가장 중요한 이유는 batch size를 추가했기 때문이다. 실제로 batch size를 추가한 experiment부터 성능이 확연하게 상승한 걸 볼 수 있다. 학습에서 batch size가 1이라면 한 sample의 gradient에 의존적이기 때문에, 해당 model의 학습은 매우 불안정할 것이다. 또한 environment 자체의 크기가 커져서 기존 환경보다 난이도가 어려워졌기 때문에 학습의 안정성을 위해서 batch size는 필수 요소라고 볼 수 있다.

Skip connection 기법 또한 layer에 추가해서 실험을 해봤지만 layer의 길이가 길지 않은 탓인지, 큰 효과는 없었다.

DQN과 A2C를 같이 보고 판단하자면, 대부분의 실험에서 DQN이 A2C 알고리즘의 성능을 가볍게 뛰어넘을뿐더러 학습의 안정성을 확보했다. A2C의 실험 결과를 보면 학습이 매우 불안정하여 score의 폭이 큰 것을 확인할 수 있다. 즉, DQN의 experiment 6가 최종적으로 결과가 가장 좋다.