

ToBigGo

오목 강화학습

김수지, 서석현, 안상준



INDEX

- 01 주제
- 02 강화 학습이란?
- 03 Architecture
- 04 구현
- 05 결론 및 한계점



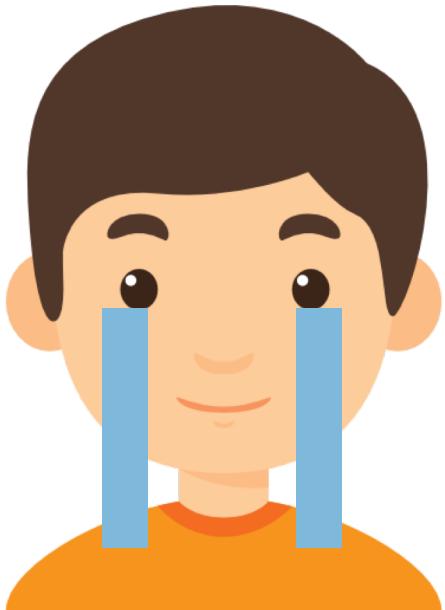
1. 주제

나는 왜 이 친구를 이길 수 없을까?

0

:

10



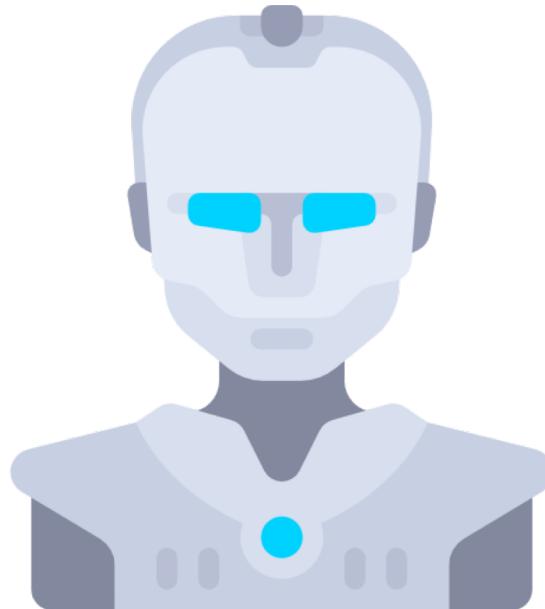
VS



1. 주제

오목을 잘 두는 AI를 직접 만들어보자!

10



:

0

VS

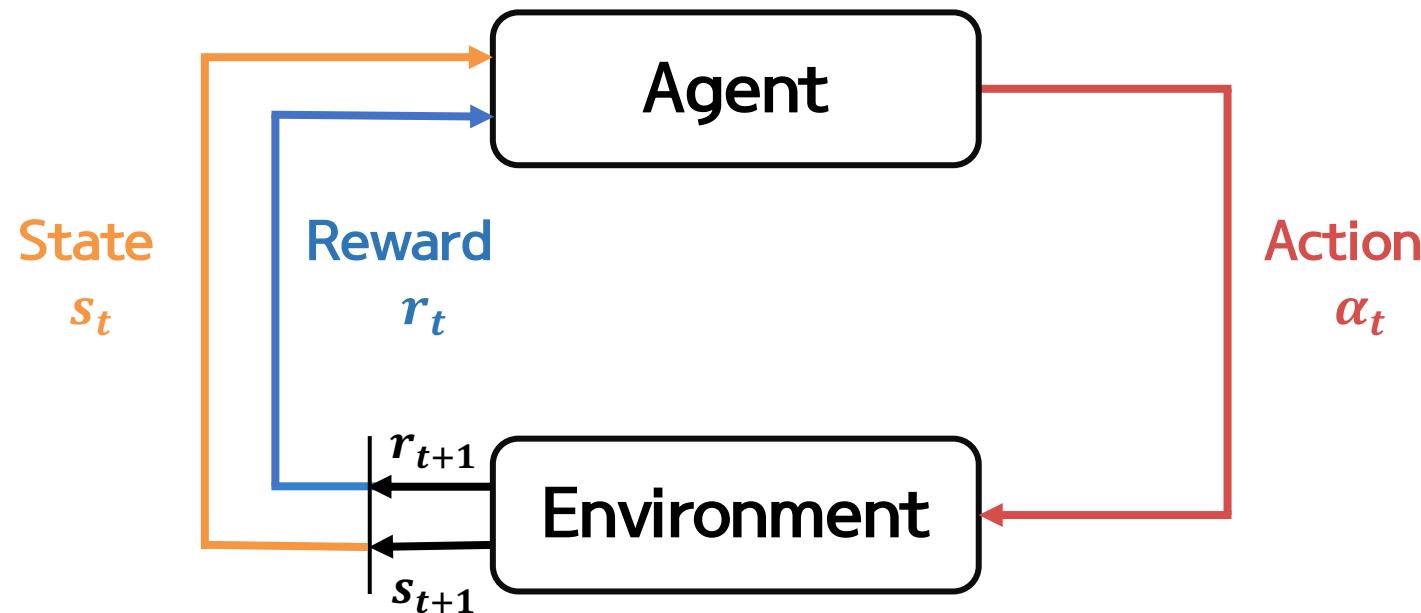


1. 주제

어떻게? 강화학습으로!

2. 강화학습이란?

어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여,
선택 가능한 행동들 중
보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법



2. 강화학습이란?

오목판

흑돌 / 백돌

흑돌과 백돌이 놓인 상태

어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여,

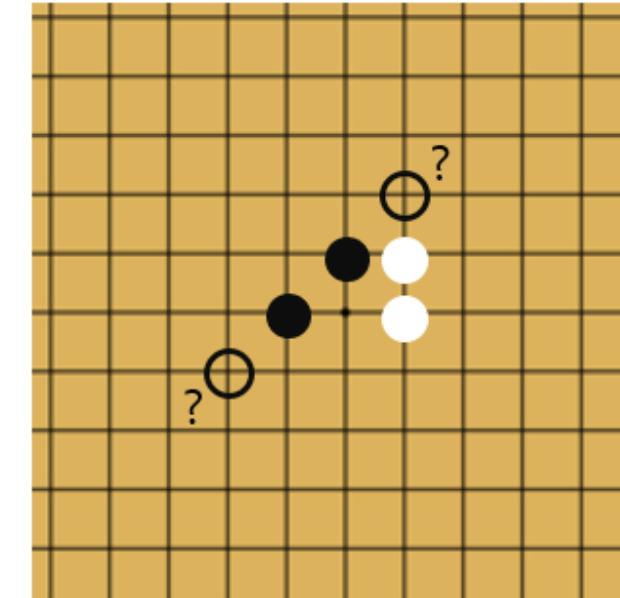
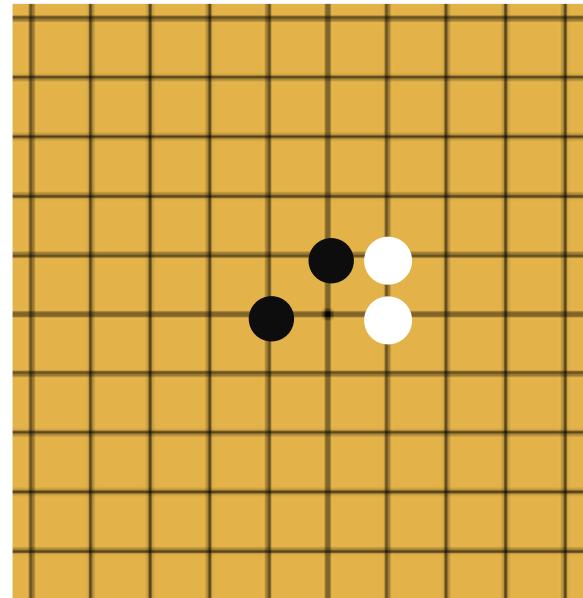
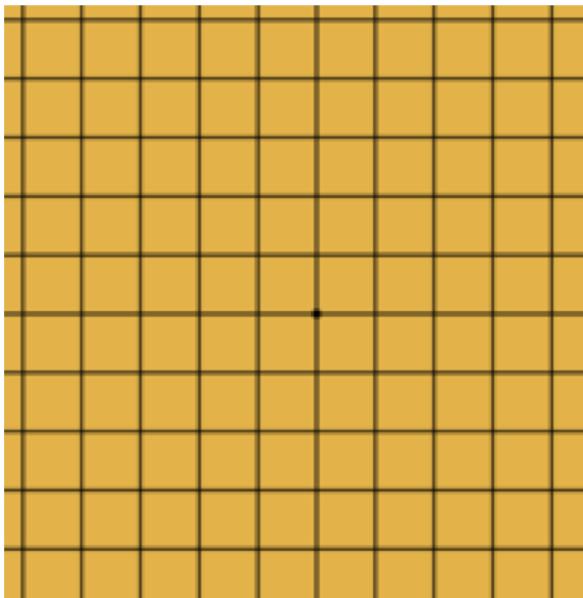
선택 가능한 행동들 중

비어 있는 자리

보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법

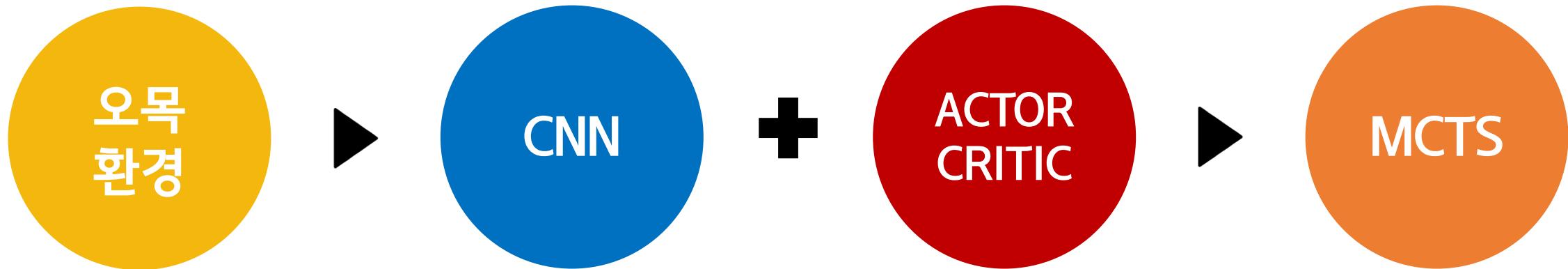
승리

다음수



3. Architecture

시도 과정



3. Architecture

시도 과정



오목
환경

3. Architecture

시도 과정

State: 현재 오목 판에 돌들이 놓인 상태

[0, 0, 0, 1, -1, 0, -1, 1, 0 …] → 9 * 9 * 3 Array (흑돌의 관점, 백돌의 관점, 실제 오목판의 상태)

Action: 다음 수를 둘 위치

[0, 0, 0, 0, 0, -1, 0, 0, 0, 0, 0, 0] → 9 * 9 Array, 다음 수에 대한 판의 위치를 Array 내 1 또는 -1로 표현

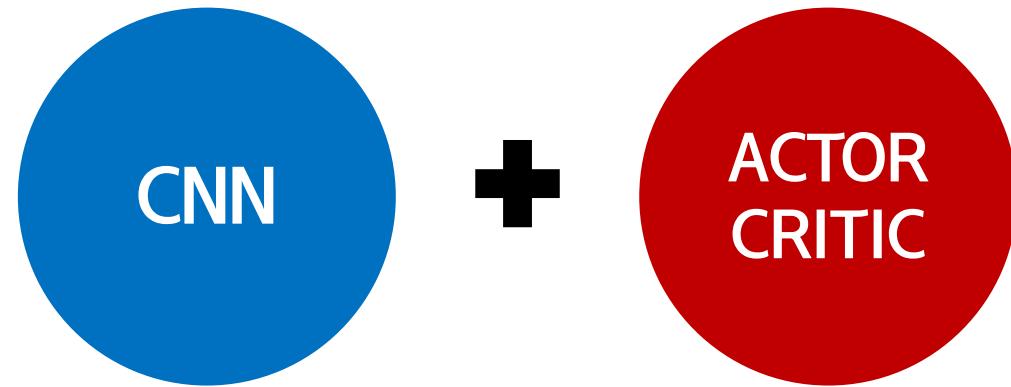
Reward: 승리 +1, 패배 -1, 무승부 0

```
episode : 64
[[ 0.  1.  1.  1.  0. -1.  1. -1.  0.]
 [ 1. -1. -1.  0. -1.  1. -1. -1.  1.]
 [ 1. -1. -1.  0. -1.  0. -1.  0.  1.]
 [ 1.  1. -1. -1.  0.  1.  1. -1. -1.]
 [-1.  1.  0. -1.  0.  0. -1.  1.  0.]
 [-1. -1. -1.  0. -1.  1. -1.  0. -1.]
 [ 1. -1.  1.  1.  1. -1.  1.  0.  1.]
 [ 1.  1.  0. -1. -1.  0.  1. -1. -1.]
 [ 1.  1.  1.  1. -1.  1.  1. -1.  1.]]
```

<9 by 9 오목판에서 64 수 만에 백 돌이 이긴 경우>

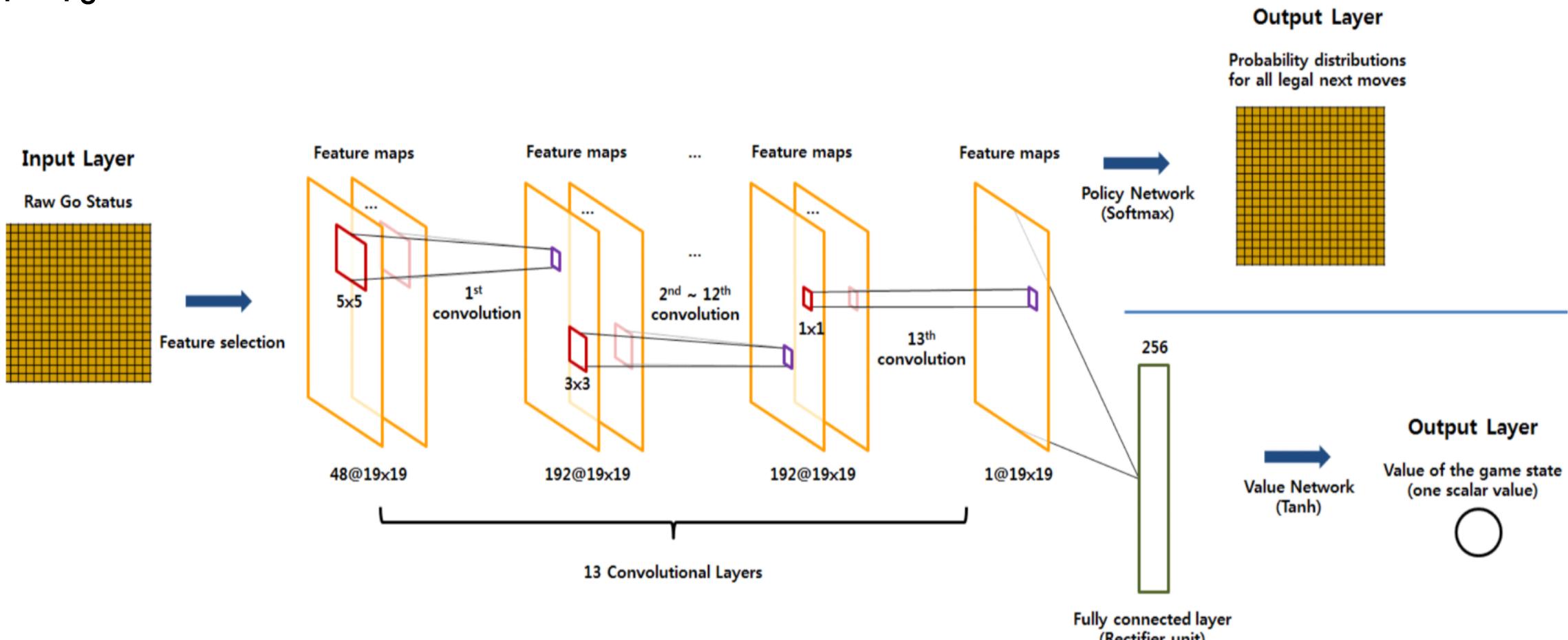
3. Architecture

시도 과정



3. Architecture

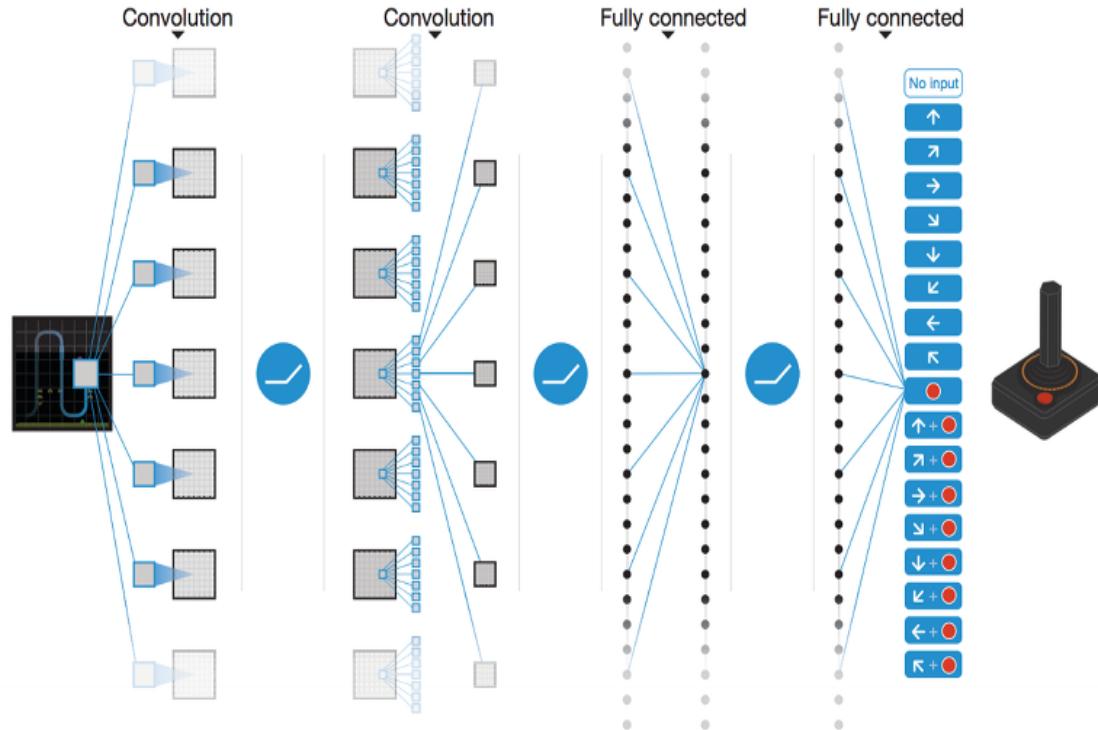
시도 과정



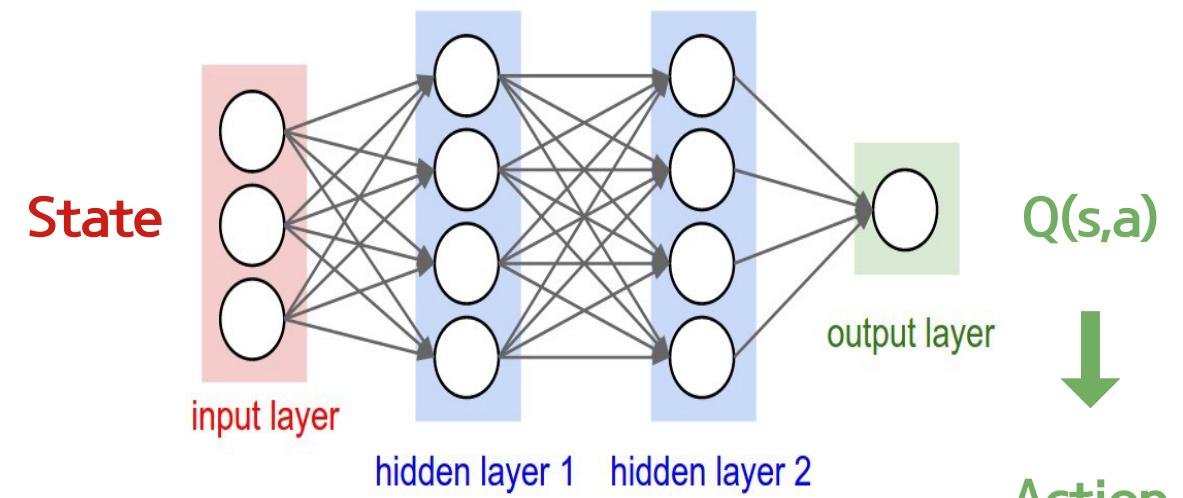
CNN

3. Architecture

시도 과정

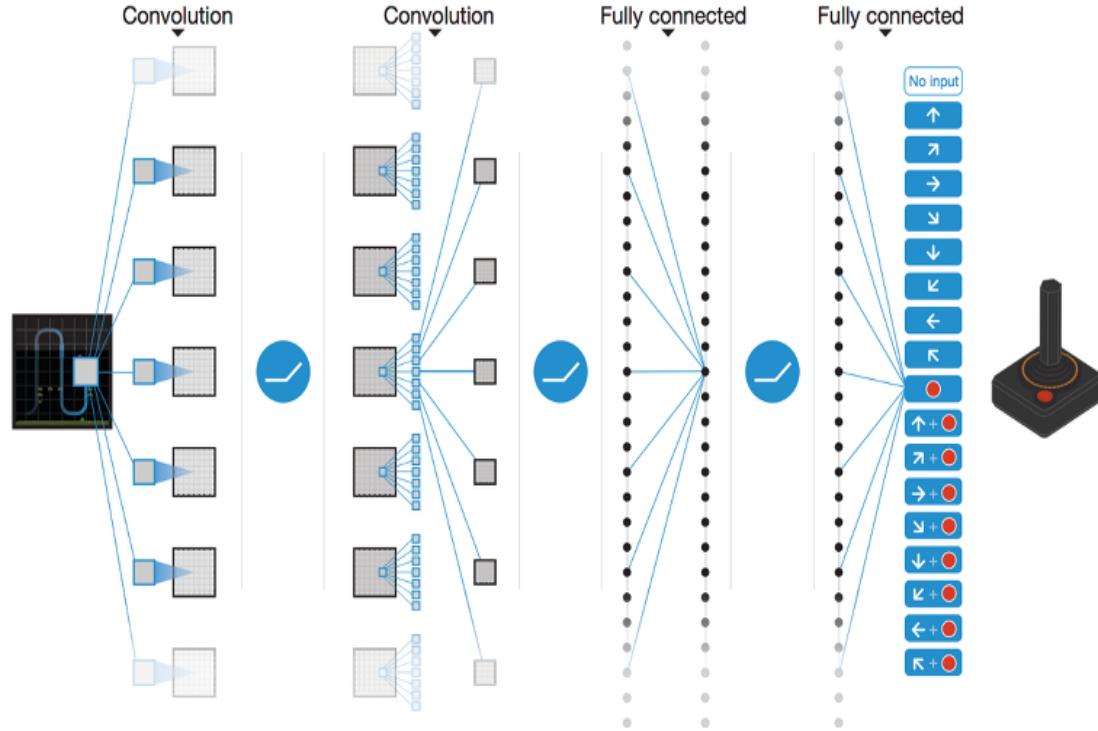


Deep Q Network
(Value Base Reinforcement)

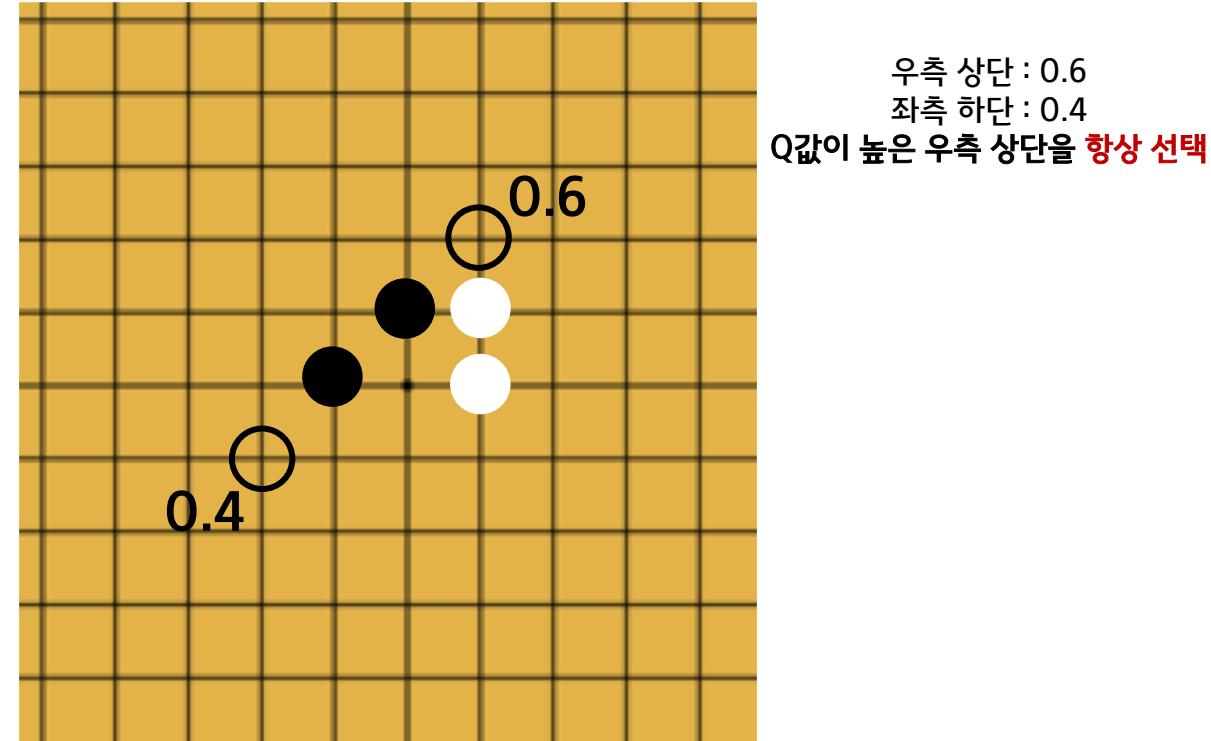


3. Architecture

시도 과정



Deep Q Network
(Value Base Reinforcement)

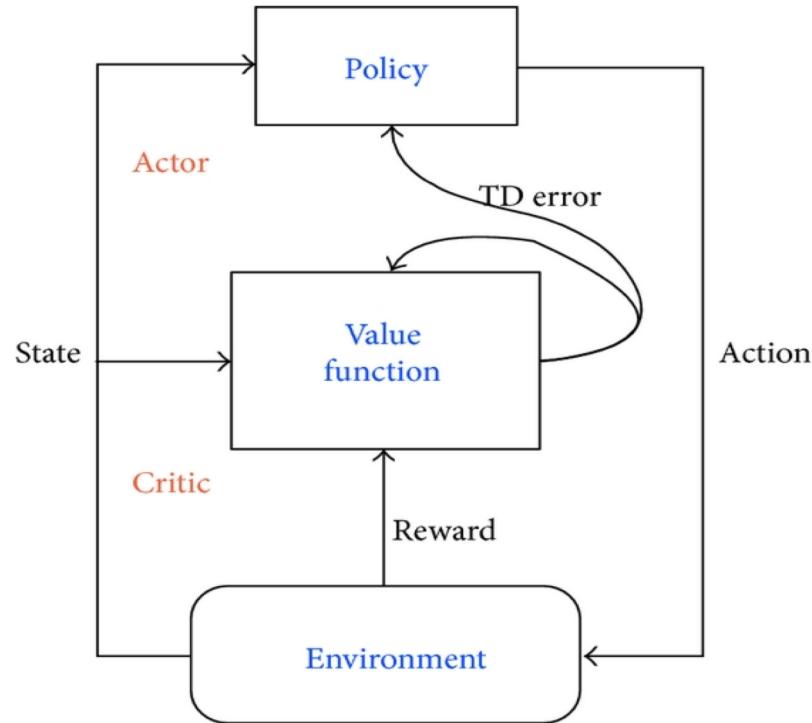


같은 State에서 Q값이 가장 높은 곳을 선택하기 때문에
같은 State에서 같은 수를 두게 됩니다.

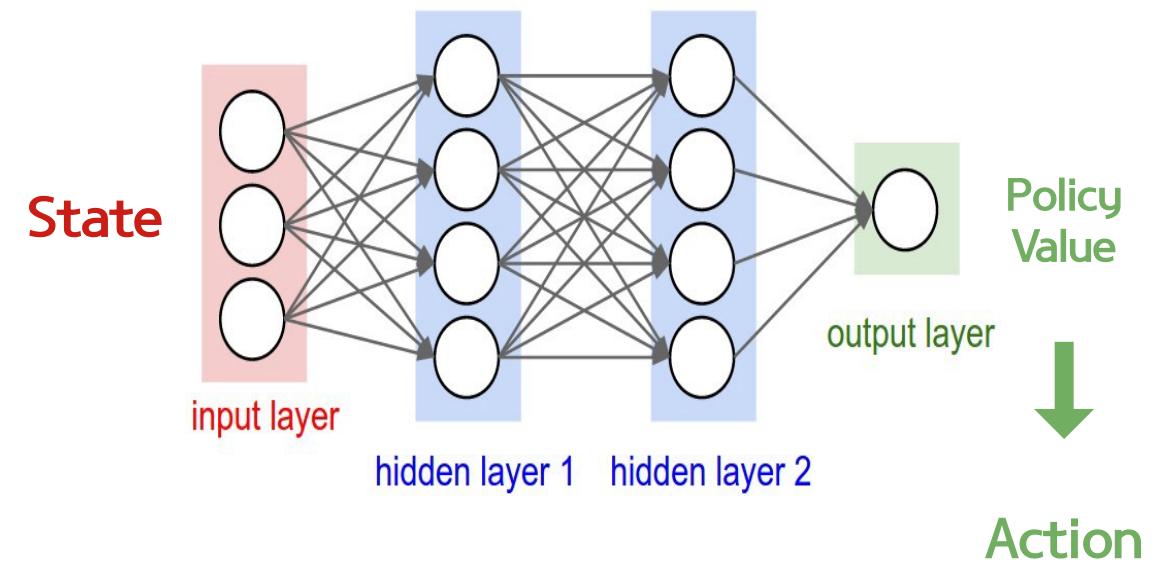
이는 오목의 다양한 경우의 수를 고려하지 못하게 됩니다.

3. Architecture

시도 과정

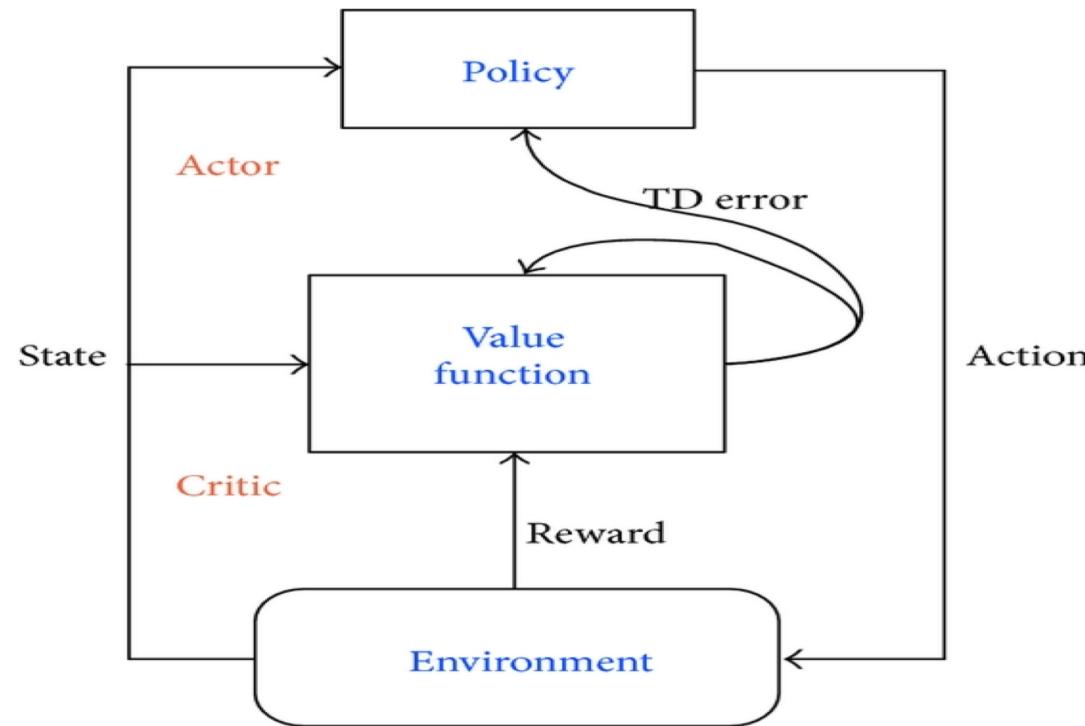


Actor - Critic
(Policy Base Reinforcement)

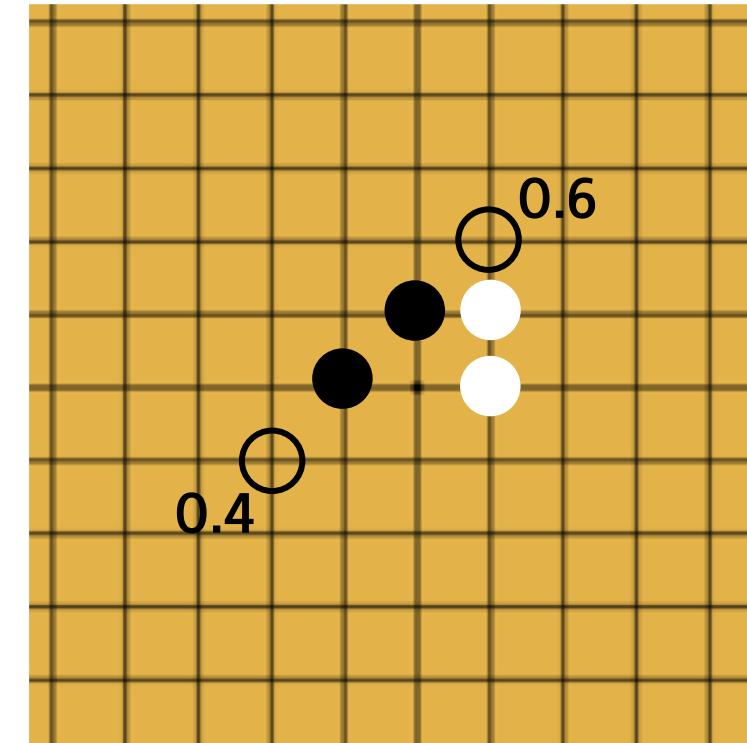


3. Architecture

시도 과정



Actor - Critic
(Policy Base Reinforcement)

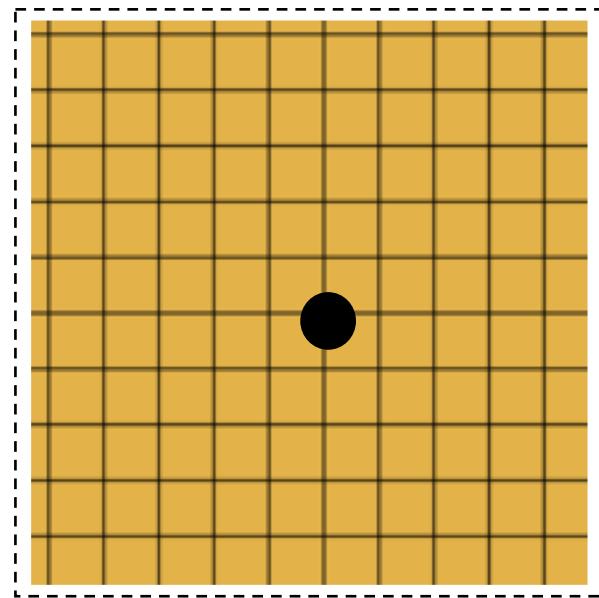


같은 State에서 Policy값에 따라 Sampling하여 다음 수 선택
같은 State에서 여러 수를 두게 됩니다.

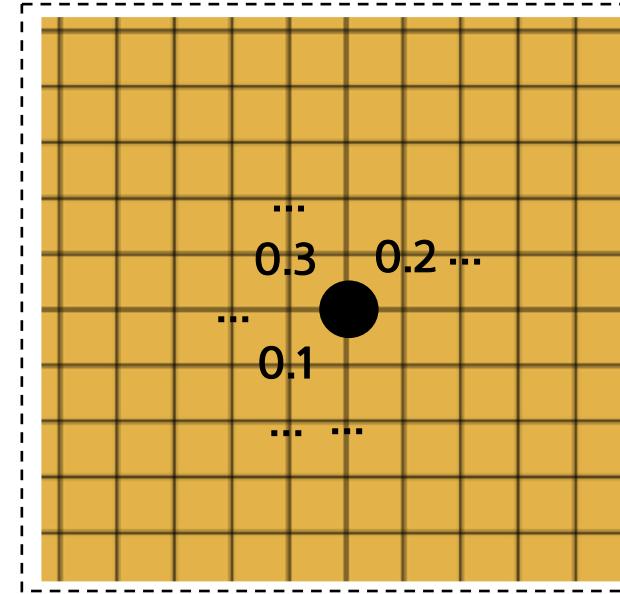
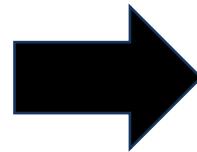
이는 오목의 다양한 경우의 수를 고려할 수 있게 되었습니다.

3. Architecture

시도 과정



Input: State



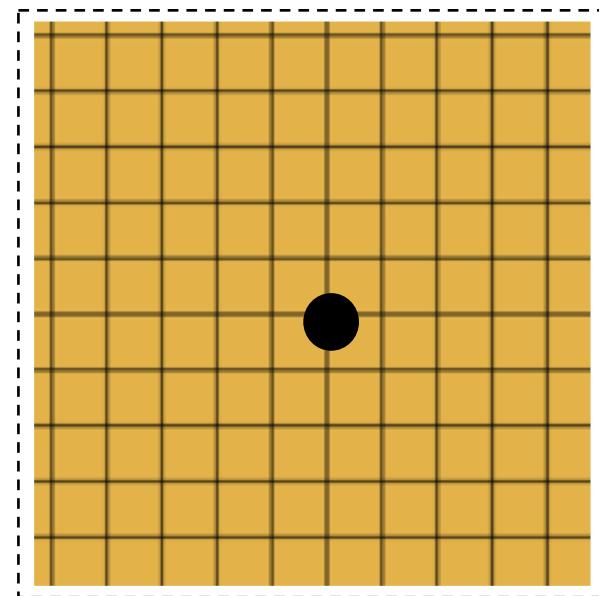
Output: Policy, Value



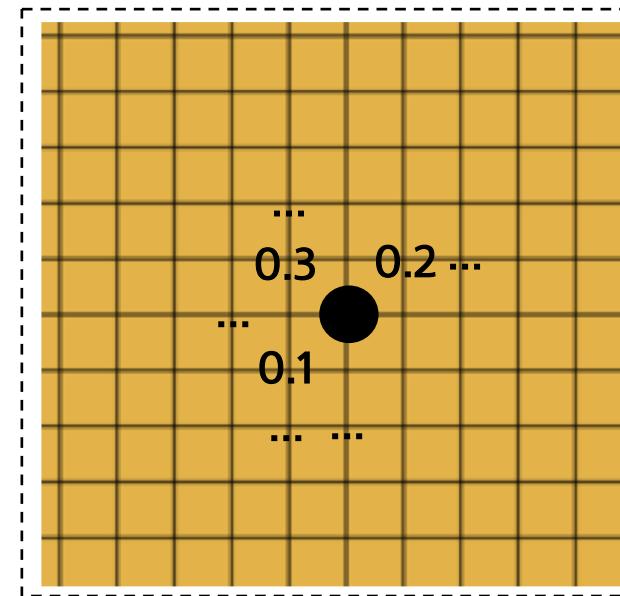
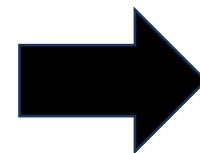
Agent가 다음 수를
어디에 놓으면 좋을 지에 대한 확률 값
→ 각 확률에 대해 다음 수 Sampling

3. Architecture

시도 과정



Input: State



Output: Policy, Value



현재 상태에서
Agent가 최종적으로 이길 것인가,
질 것인가에 대한 판세 판단,
 $-1 \leq v \leq 1$

3. Architecture

시도 과정

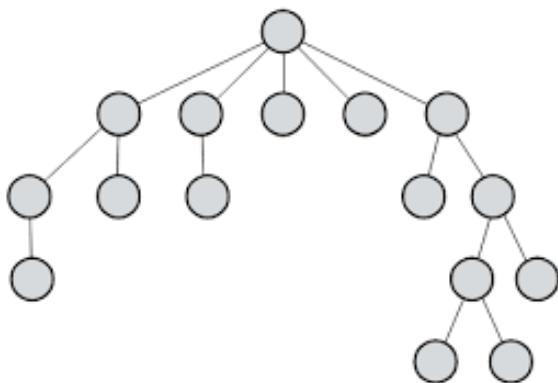


MCTS

3. Architecture

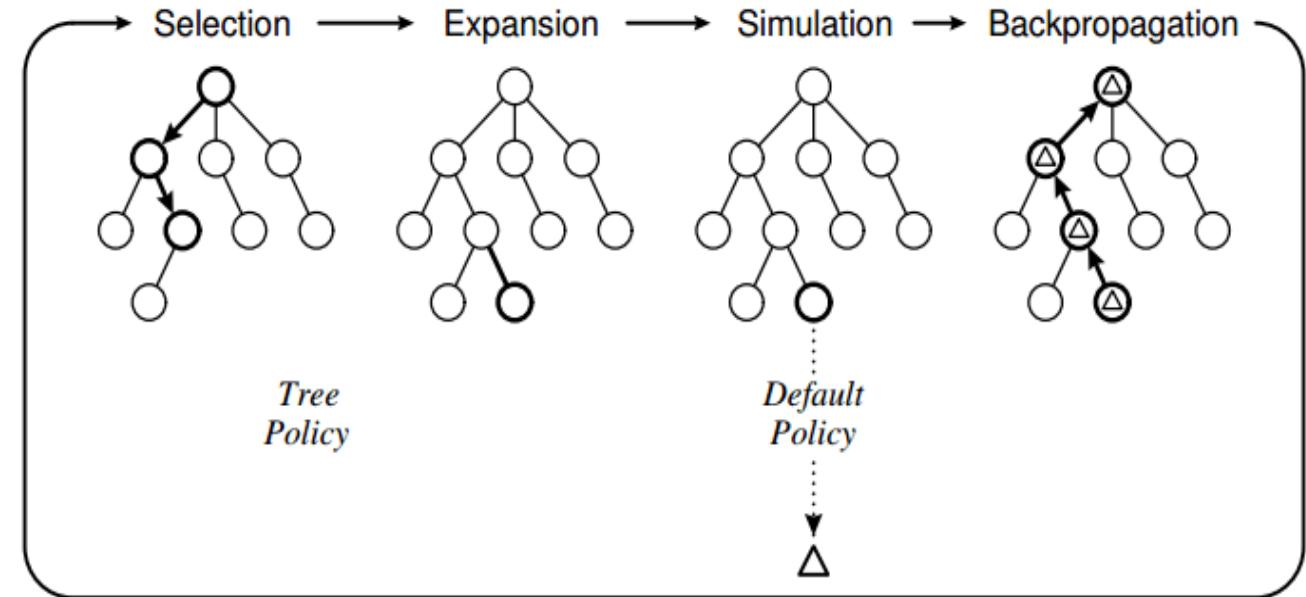
시도 과정

Tree Search



State 기반 모든 경우의 수를 고려

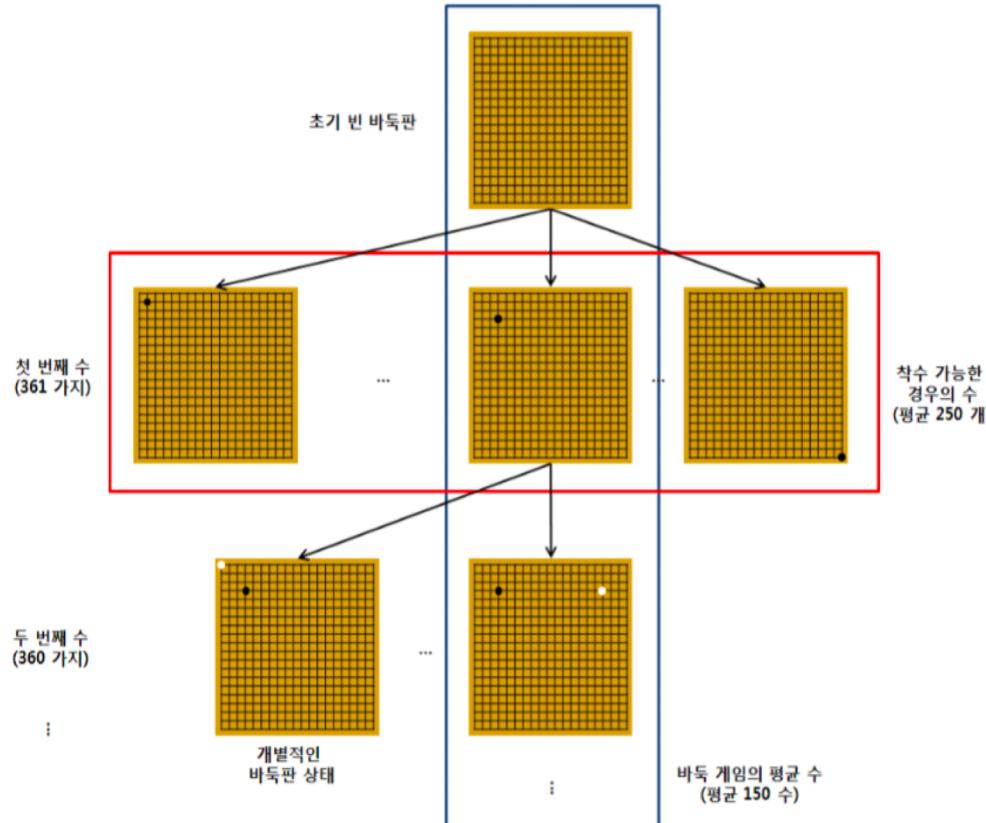
Monte Carlo Tree Search



적절한 시뮬레이션을 통해
모든 경우의 수를 고려한 결과와 근사한 값

3. Architecture

시도 과정



Tree의 **너비**와 **깊이**를 줄여 Search Space 축소
→ Actor-Critic의 Output (Policy, Value)을 활용하자 !

Policy

- Tree의 **너비**를 제한하는 역할.
- 특정 상태에서 가능한 모든 수 중 가장 승률이 높은 수를 선택, 승률이 높을 수록 다른 경우의 수를 고려하지 않는 방식

Value

- Tree의 **깊이**를 제한하는 역할.
- Value는 현재 상황의 승산을 나타낸 것으로, 승산이 정확할 수록 많은 수(더 깊은 노드)를 고려하지 않는 방식

3. Architecture

시도 과정



4. 구현

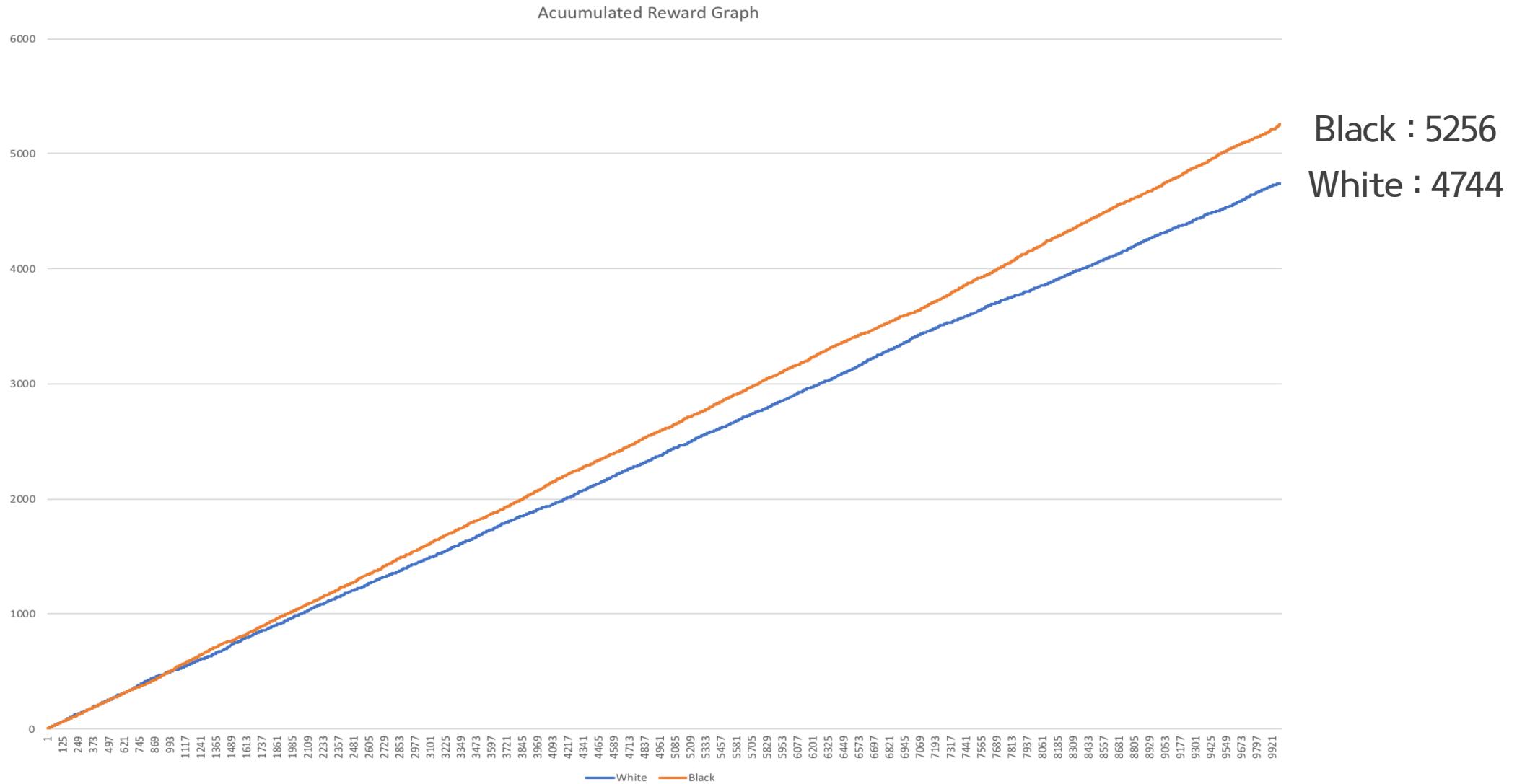
결과

승률

방식

4. 구현

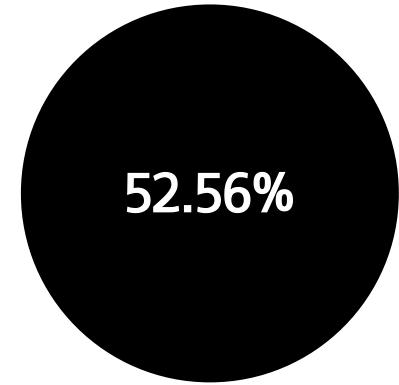
결과 - 승률



4. 구현

결과 - 승률

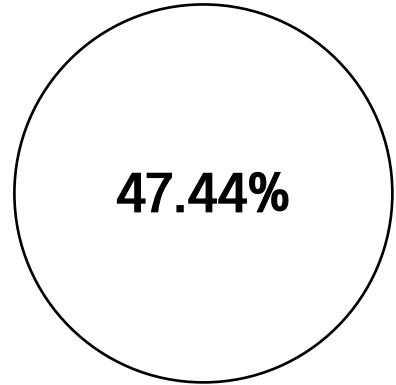
흑



52.56%

VS

백

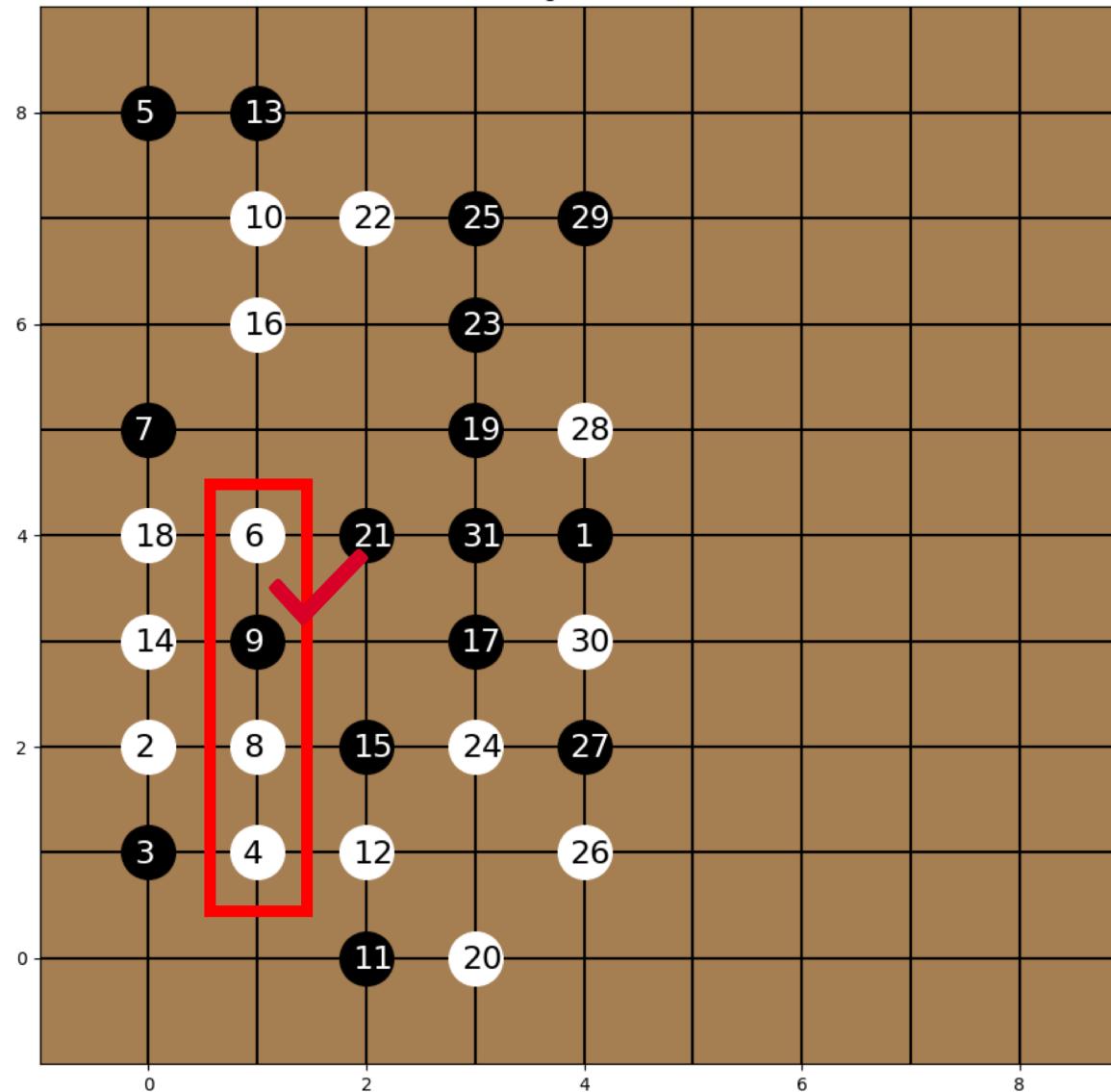


47.44%

4. 구현

결과 - 방식

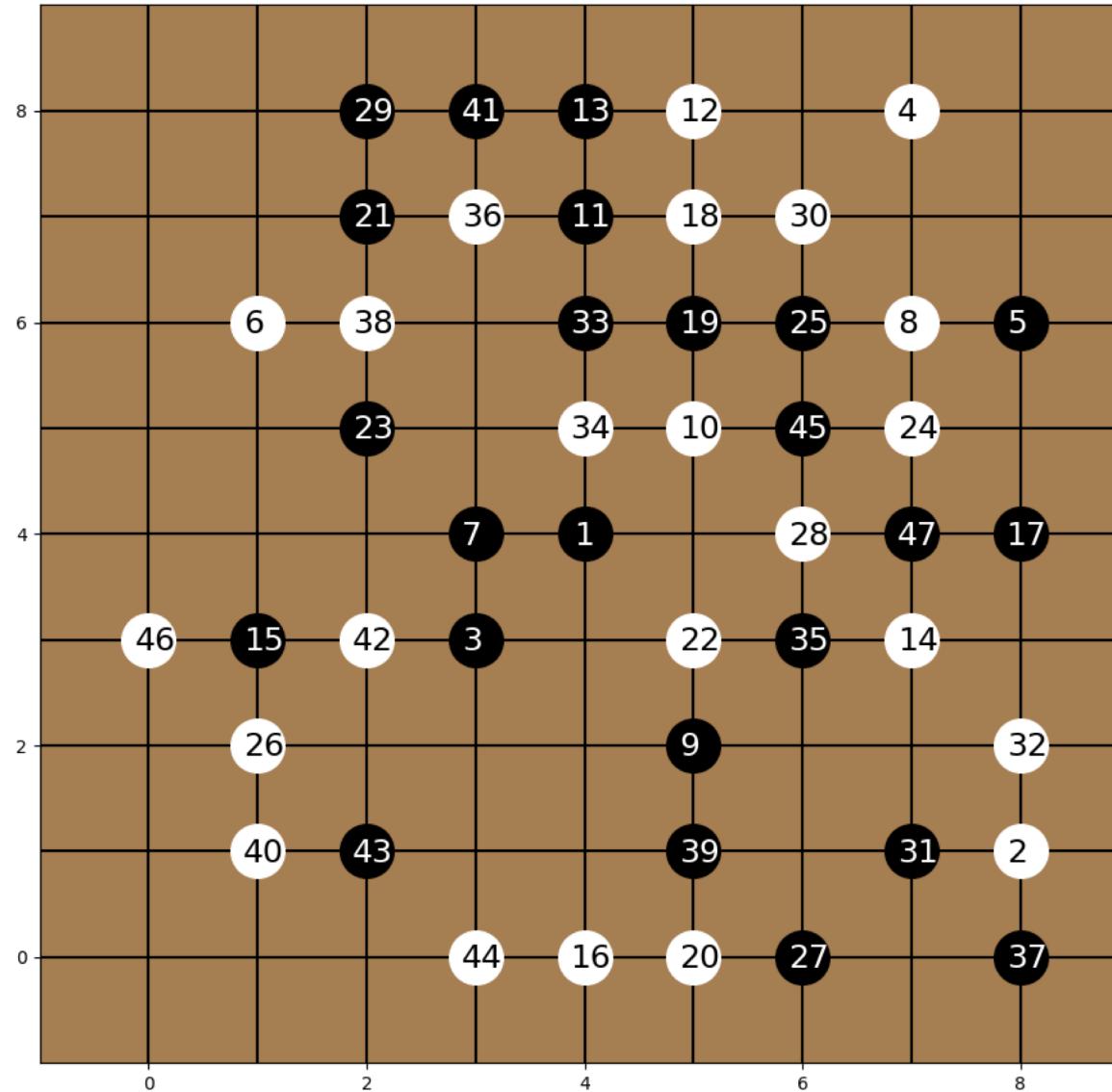
Tobigo : AI vs AI



4. 구현

결과 - 방식

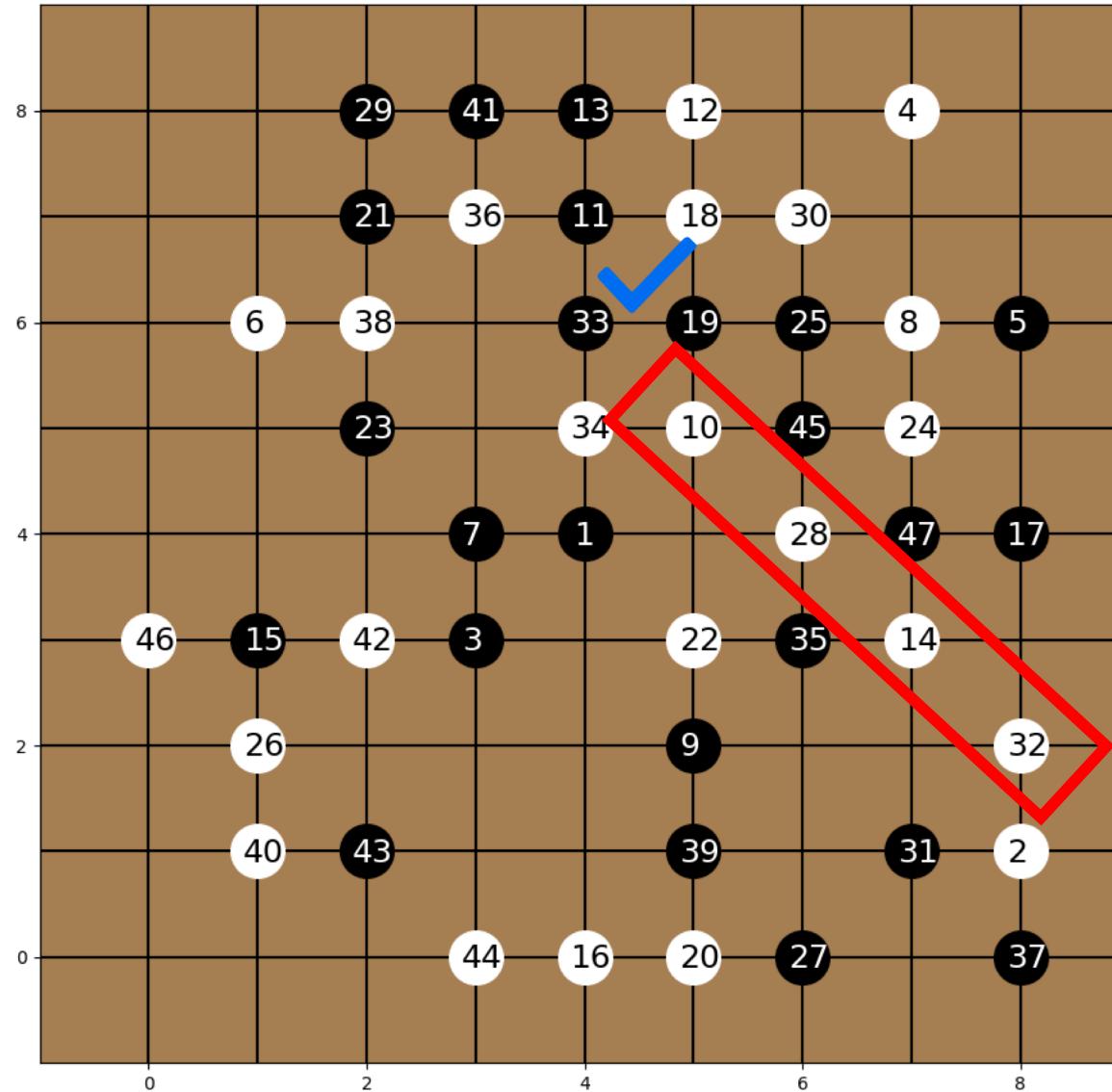
Tobigo : AI vs AI



4. 구현

결과 - 방식

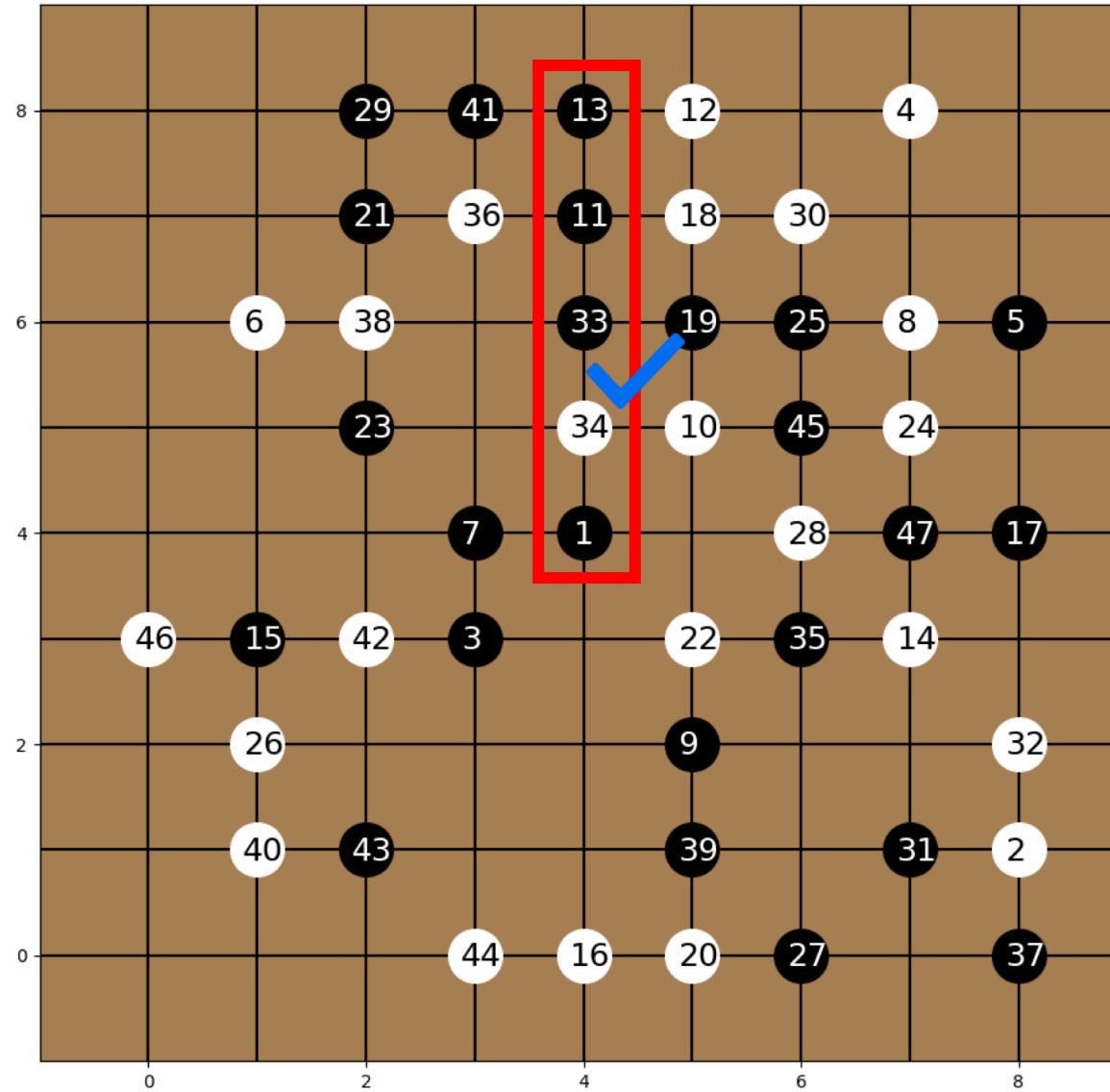
Tobigo : AI vs AI



4. 구현

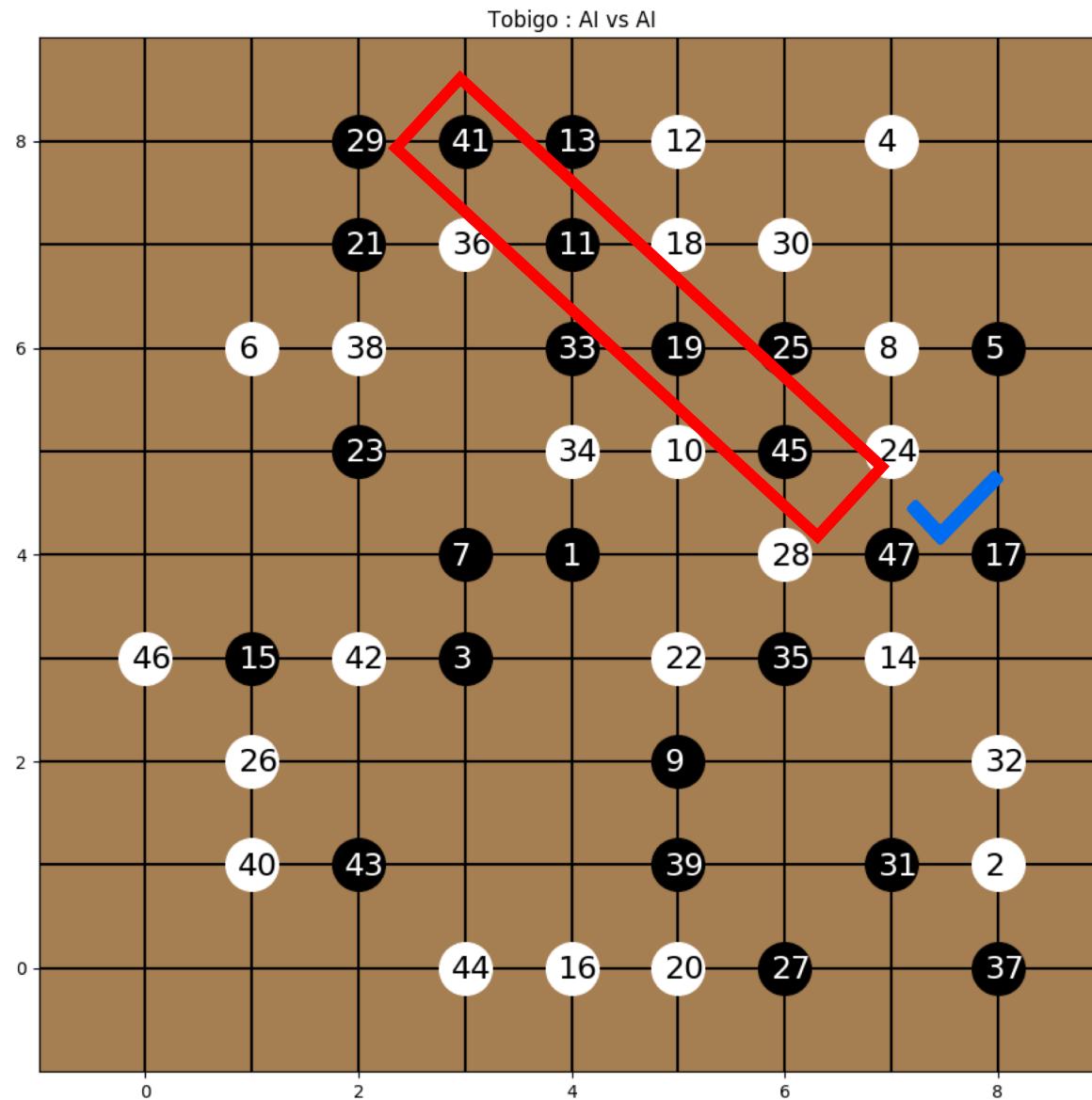
결과 - 방식

Tobigo : AI vs AI



4. 구현

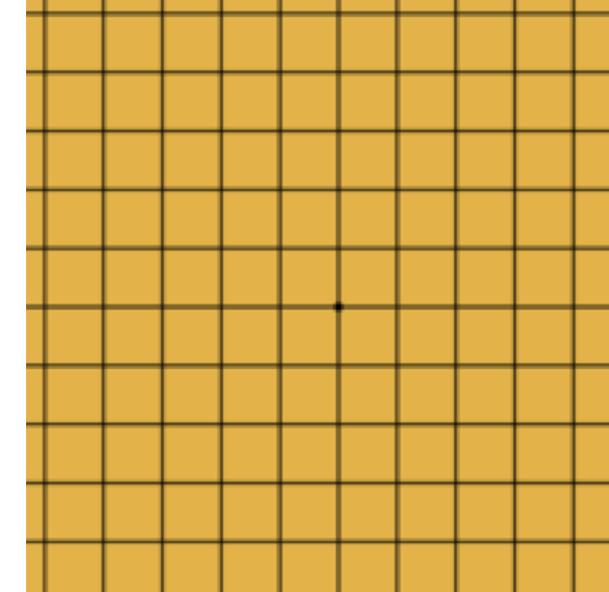
결과 - 방식



5. 결론 및 한계점



AI 구축에 대한
접근 용이성



한 수 → Local: 9초 / Server: 2초
오래 걸리는 시행착오 과정

팀 소개



김수지
덕성여자대학교
통계학과 (14)



서석현
고려대학교
통계학과 (14)



안상준
국민대학교
빅데이터경영통계(13)

ToBigGo

Q & A

ToBigGo

감사합니다.

