나만의 알파고 만들기

알파 오목

김정태

알파고를 사랑하는 수학 강사
알파고, 강화학습, 딥러닝, 코딩 독학
퇴근 후에만 두뇌가 풀가동되는 이상종

RL Korea 알파오목팀에서 프로젝트 진행 중
(with 민규식, 이웅원, 김태영)

kekmodel@gmail.com
facebook.com/kekmodel
github.com/kekmodel

알파고의 진화과정과 핵심 아이디어

알파오목 프로젝트 소개

Google DeepMind

CEO Demis Hassabis

알파고의 아버지 David Silver

"범용 인공지능을 만드는 것이 목표"

2010년 DeepMind 창업

- 스스로 학습하는 강화학습 알고리즘 개발

2014년 구글이 5억달러에 인수

2015년 알파고 공개 (AlphaGo Fan)

- Nature 논문 발표
- 바둑 AI 최초로 프로 바둑기사(판 후이 2단)에게 승리

2016년 이세돌 9단과 대결 (AlphaGo Lee)

- 4 : 1 알파고 승리

2017년 5월 커제 9단과 대결 (AlphaGo Master)

- 3 : 0 알파고 승리

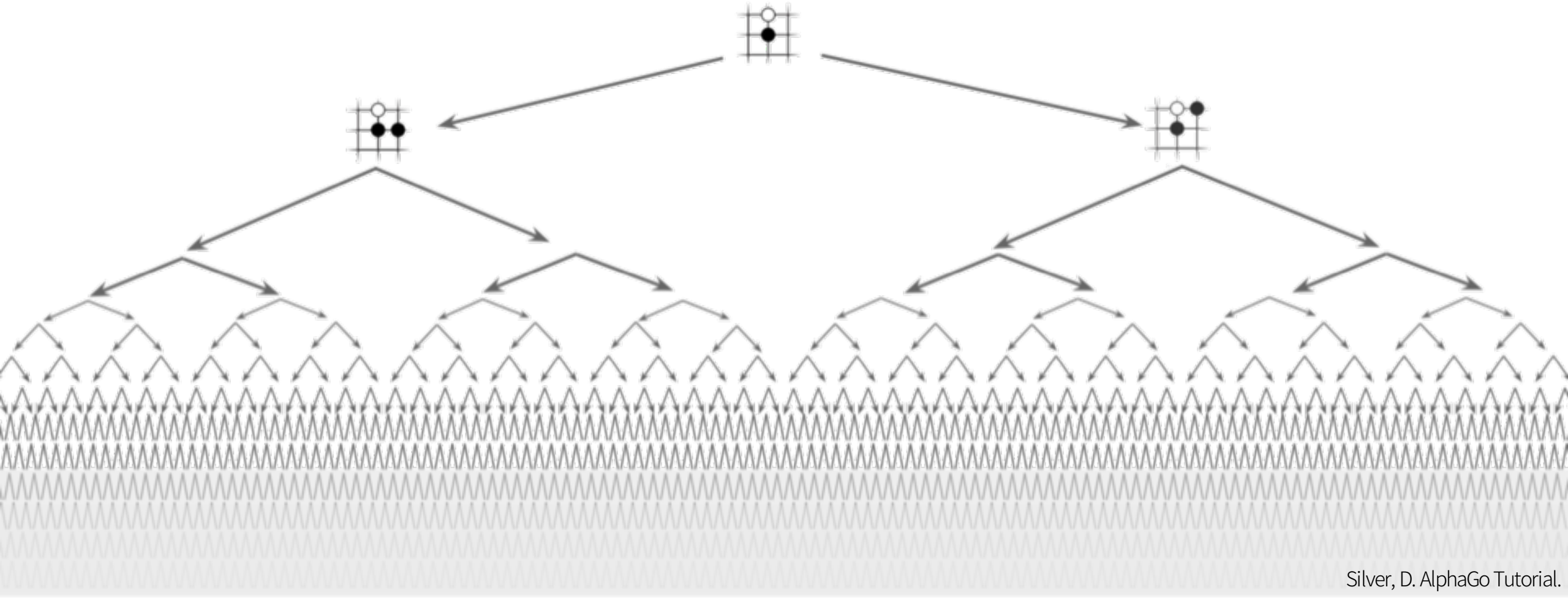
2017년 10월 알파고 제로 공개 (AlphaGo Zero)

- Nature 논문 발표
- 인간 기보 없이 스스로 학습
- 바둑 AI의 SOTA

2017년 12월 알파 제로 공개 (Alpha Zero)

- arXiv 논문 발표
- 보드게임 범용 알고리즘 (체스, 쇼기, 바둑)
- 체스, 쇼기 AI의 SOTA


왜 바둑에 도전했을까?



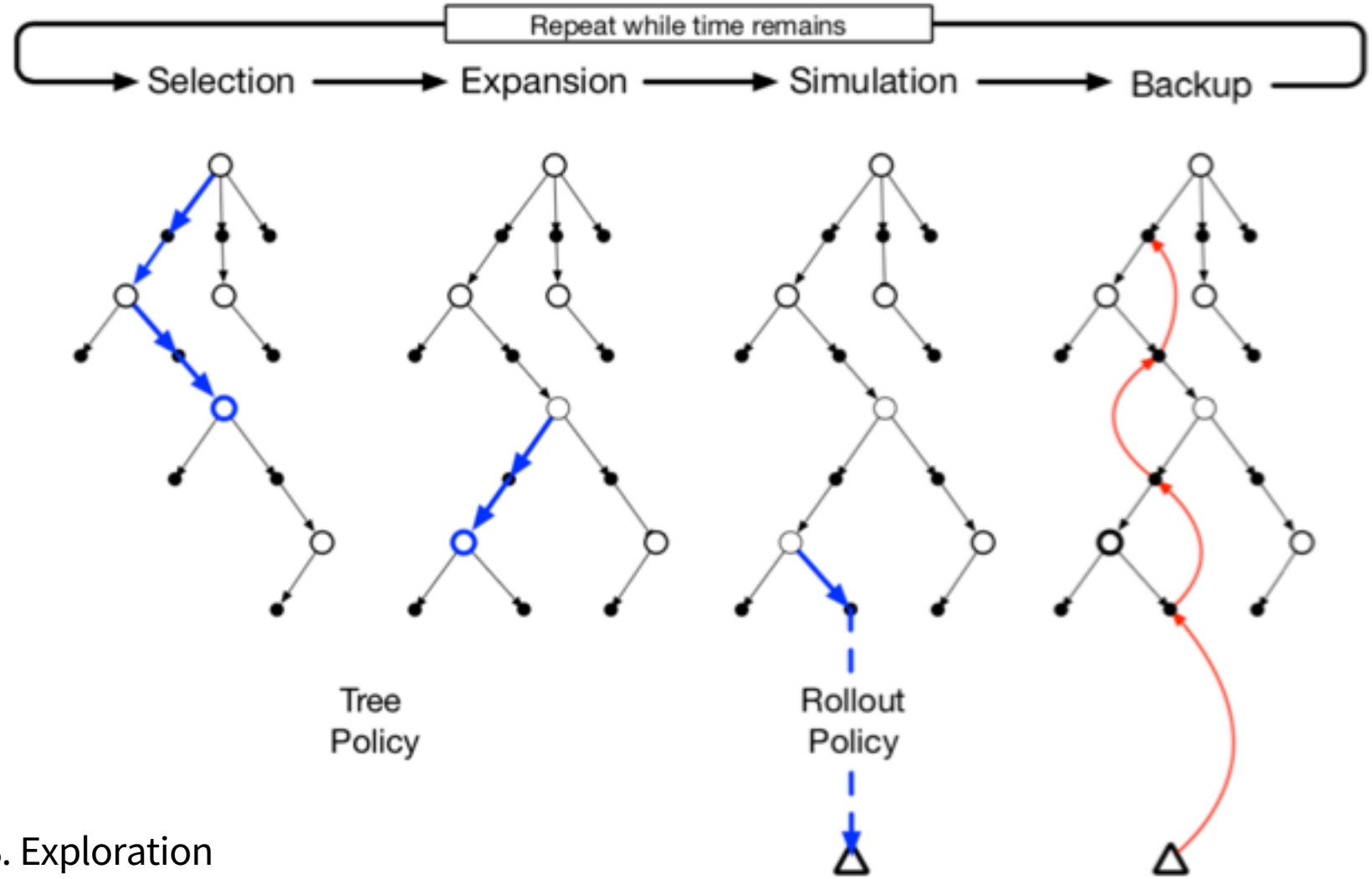
Silver, D. AlphaGo Tutorial.

바둑의 경우의 수: 약 2×10^{170} 개

우주의 원자의 수: 약 12×10^{78} 개

“모든 AI의 무덤”

Monte Carlo Tree Search



Exploitation VS. Exploration

$$UCB = \frac{W}{N} + \sqrt{\frac{2 \ln T}{N}}$$

Sutton, Richard S. Reinforcement Learning: An Introduction. 2nd ed., The MIT Press, 2018.

MCTS의 핵심 아이디어

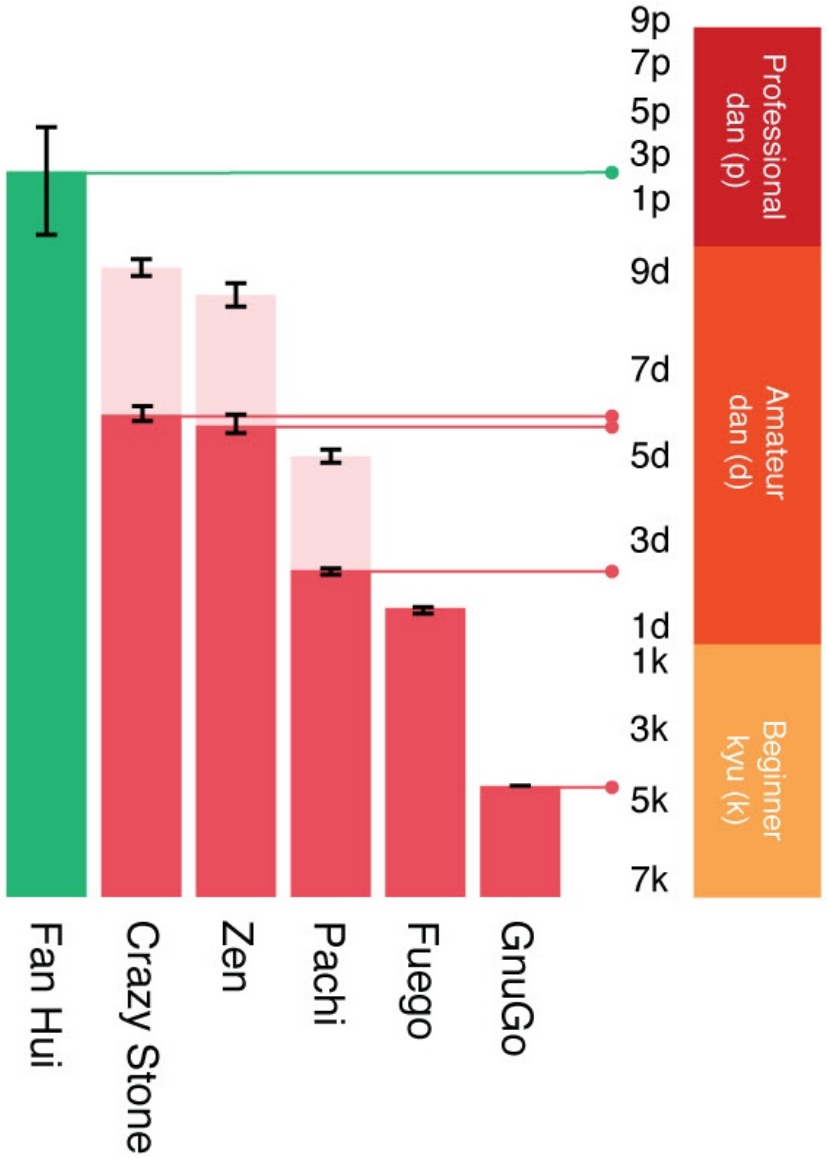
Rollout Policy

랜덤 시행으로 승부 결과를 통계적으로 근사

Tree Policy

Rollout 결과를 바탕으로 의미 있는 곳 위주로 탐색

AlphaGo 이전 AI들의 기력



MCTS의 등장으로..

AI의 기력 수직 상승
(아마 6단 수준)

그러나..

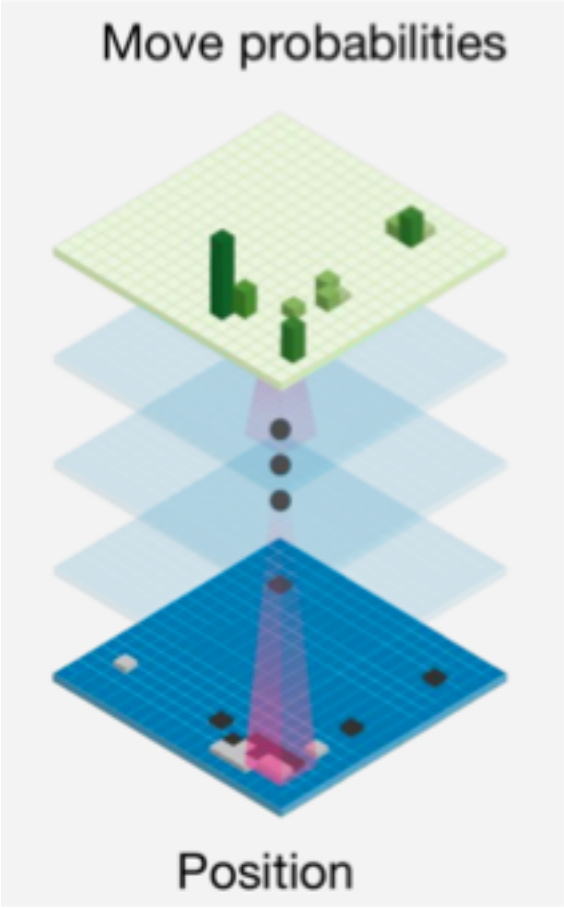
프로기사에겐 여전히 역부족

Silver, D. et al. Mastering the game of Go with deep neural networks and tree search (2016).

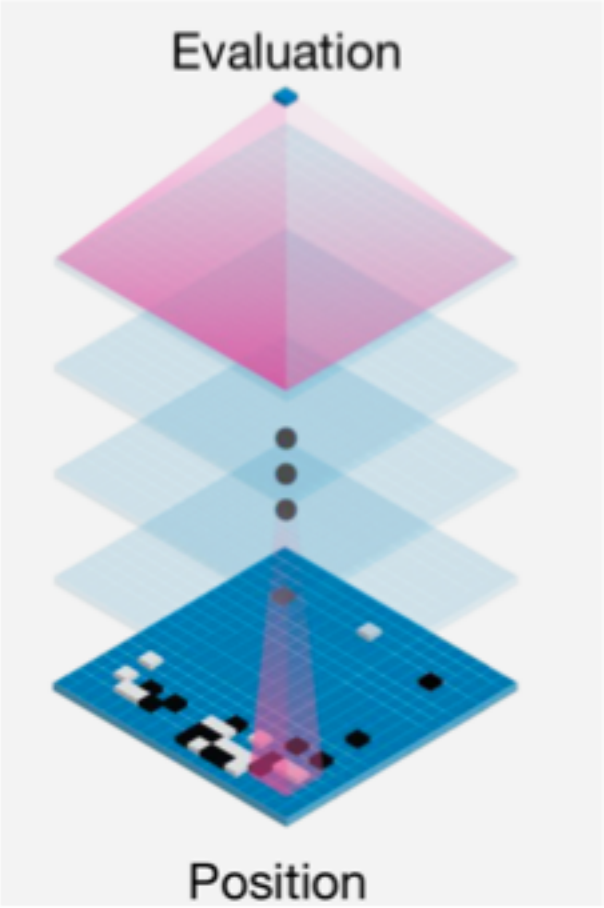
여기서 덥터닝이 출동한다면?

알.파.고

2 Neural Networks



Policy network



Value network

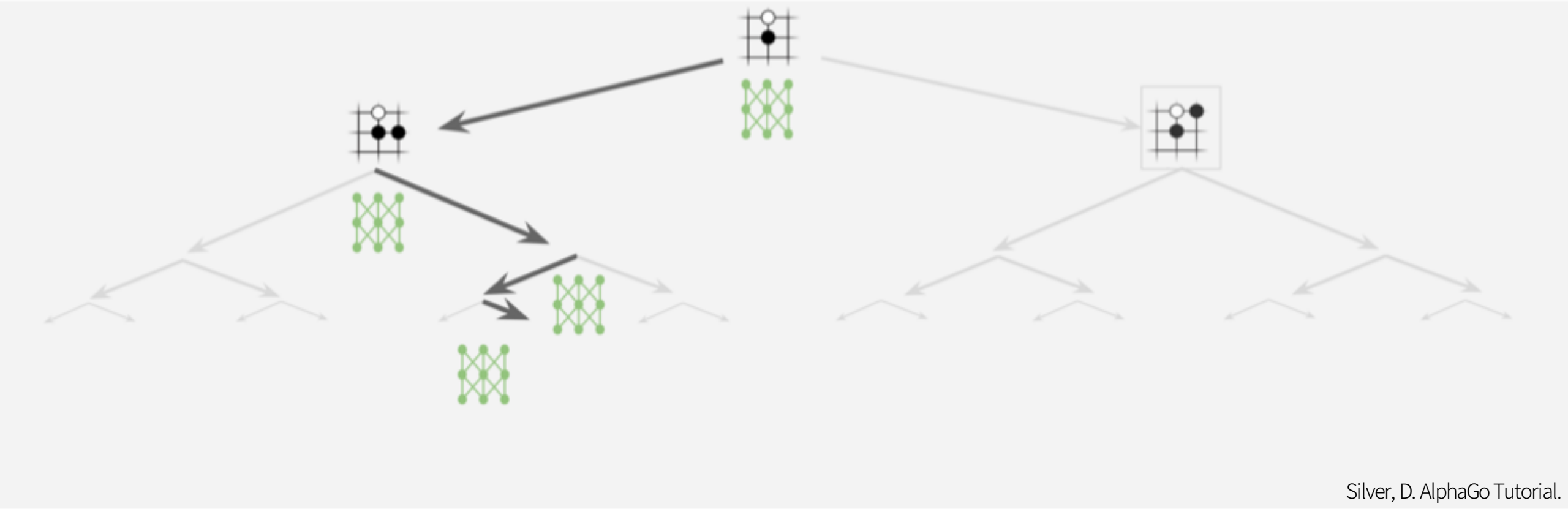
Silver, D. AlphaGo Tutorial.

AlphaGo Fan ~ Master

$$UCB = \frac{W}{N} + \sqrt{\frac{2 \ln T}{N}} \quad \rightarrow \quad PUCB = \frac{\boldsymbol{V}}{N} + \boldsymbol{P} * \frac{5\sqrt{T}}{1+N}$$

Silver, D. AlphaGo Tutorial.

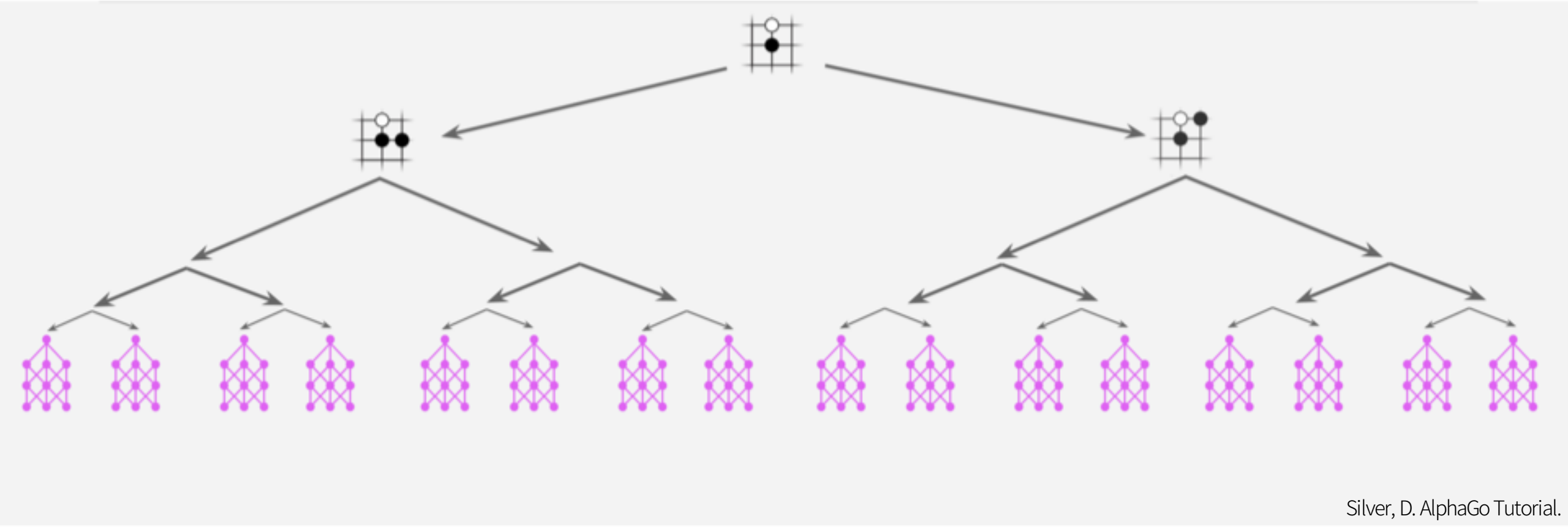
AlphaGo: MCTS



Silver, D. AlphaGo Tutorial.

Policy Network로 선택의 폭을 줄임

AlphaGo: MCTS



Silver, D. AlphaGo Tutorial.

Value Network로 승부 예측을 더 정확하고 빠르게

AlphaGo Algorithm 요약

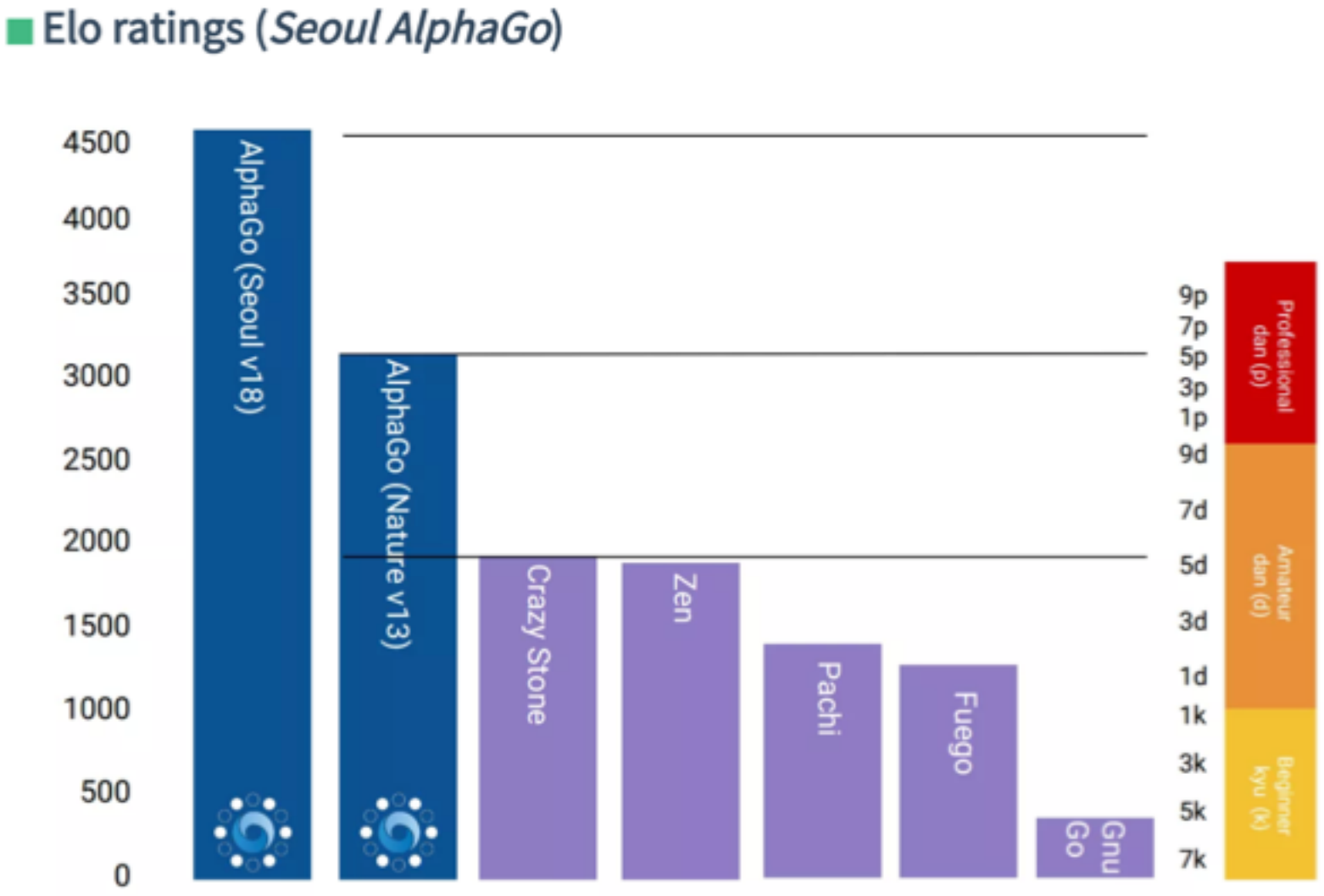
SL → RL → MCTS

1. SL로 Policy network 초기화
2. Policy network를 셀프플레이 RL로 개선
3. 개선된 Policy network이 셀프플레이한 데이터로 Value network 학습

+ 이미 알려진 패턴들 일부 적용(e.g. 축 탈출, 자충수 등)

→ P, V network를 MCTS에 붙여서 실제 플레이에 사용

AlphaGo의 기력



Silver, D. AlphaGo Tutorial.

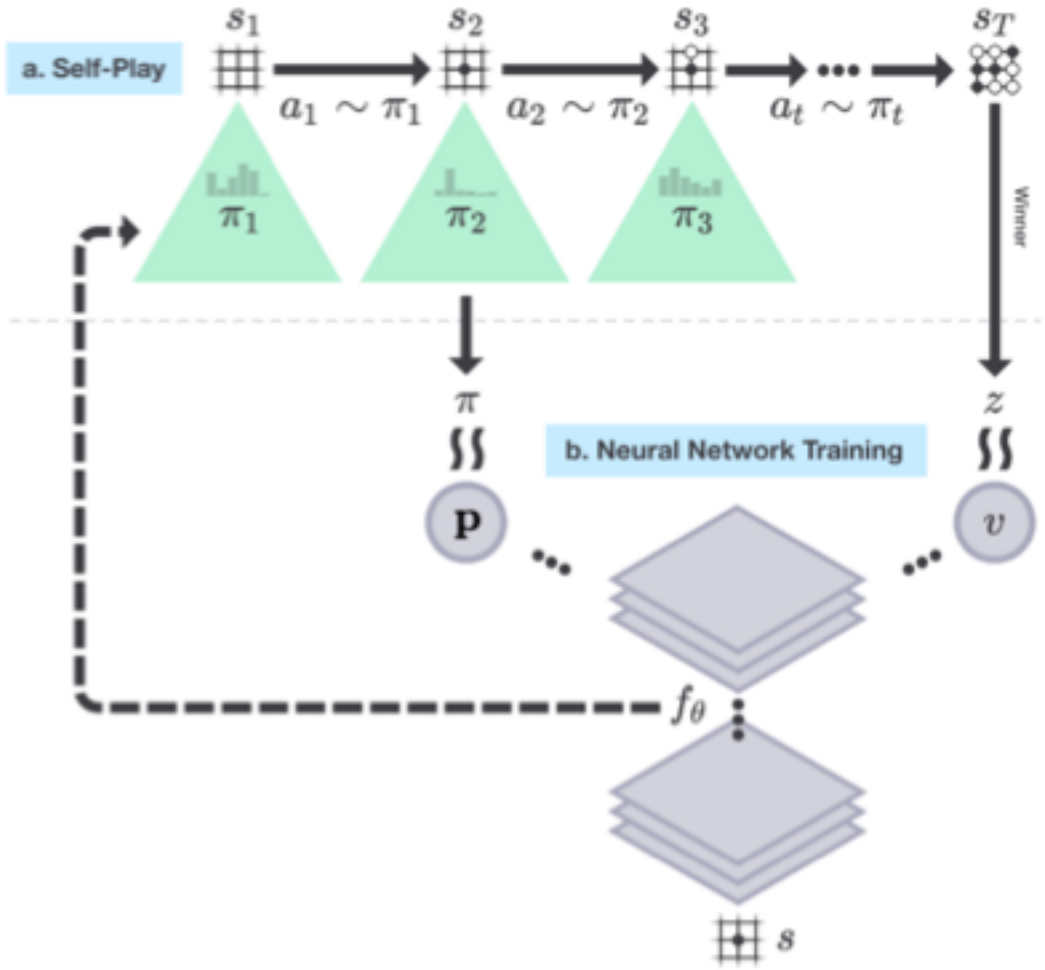
인간의 주관을 완전히 배제한다면?

알파고 제로

AlphaGo Zero

- No human data
  - 랜덤 초기화된 네트워크의 셀프플레이 데이터로만 학습
- No human features
  - 오직 바둑판의 정보만으로 학습
- Single neural network
  - Policy network와 Value network를 하나로 통합
  - ResNet 기반
- Simpler search
  - Rollout을 Value network의 평가로 완전히 대체

AlphaGo Zero: Learning



$$l = (z - v)^2 - \boldsymbol{\pi}^{\mathrm{T}} \log \boldsymbol{p} + c\|\theta\|^2$$

AlphaGo Zero Algorithm 요약

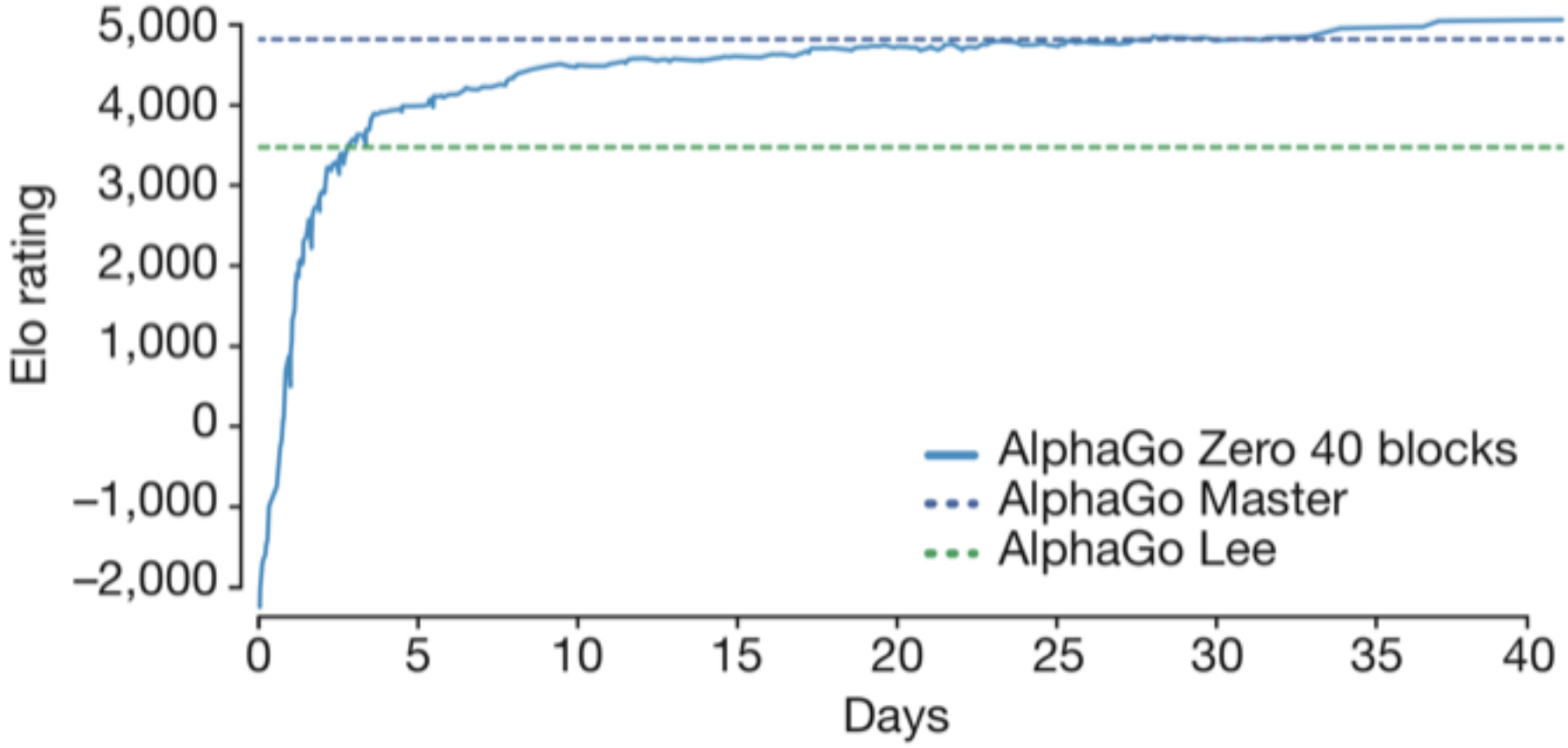
RL + MCTS ➔ MCTS

1. 랜덤 초기화한 P, V network를 탑재한 MCTS로 셀프플레이 데이터 생성
2. 셀프플레이한 데이터(state, π , z) 로 P, V network를 학습
3. 주기적으로 이전 모델보다 강한지 평가
4. 평가기준을 통과한 P, V network + MCTS로 셀프플레이 데이터 재생성

+ 1:1 보드게임의 특성 상 대칭, 반전에 불변이므로 Data 확장에 사용

➔ 가장 강한 P, V network + MCTS을 실제 플레이에 사용

AlphaGo Zero: Learning Curve



Silver, D. et al. Mastering the game of Go without human knowledge (2017).

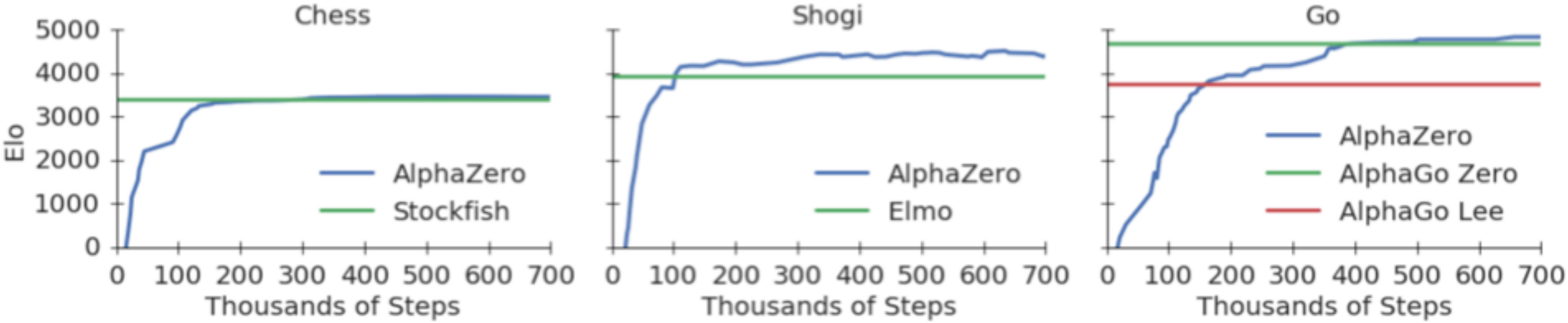
제로 알고리즘이 범용적으로 쓰일 수 있을까?

알파제로

AlphaZero

- **하나의 알고리즘으로 3가지 보드게임을 마스터**
  - 체스, 쇼기(일본 장기), 바둑
- **AlphaGo Zero 알고리즘을 단순화, 일반화 함**
  - Hyperparameter 튜닝 없음
  - 이전 모델보다 강한지 평가하는 과정 생략
  - 대칭, 반전 등 데이터 조작 일체 없음

AlphaZero: Learning Curve



Silver, D. et al. Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm (2017).

핵심 아이디어

- **어떻게 현재보다 개선된 데이터를 얻을 것인가?**
  - 셀프플레이 강화학습
    - 착수 확률 P와 승률 V 두가지를 학습하여 서로 보완
  - Neural net의 아웃풋을 그대로 사용하지 않고 MCTS로 보정
    - 비선형의 한계를 극복
  - 평가를 통해 더 높은 평가를 받은 Neural net이 데이터를 생성
    - 진화 알고리즘과 유사

알파 오목 프로젝트

알파 오목

- **RL Korea의 팀 프로젝트 중 하나**
  - 페이스북 그룹: ReinforcementLearningKR
- **취지**
  - 알파고 알고리즘을 제대로 공부하고 작은 규모에서 직접 구현해보자
- **목표**
  - 알파 제로 알고리즘을 적용하여 오목을 마스터 하기
  - 학습된 AI를 웹에 올려서 사람들이 직접 대결해볼 수 있도록 하기
- **알파오목 팀**
  - 매니저: 민규식
  - 팀원: 이웅원, 김태영, 김정대

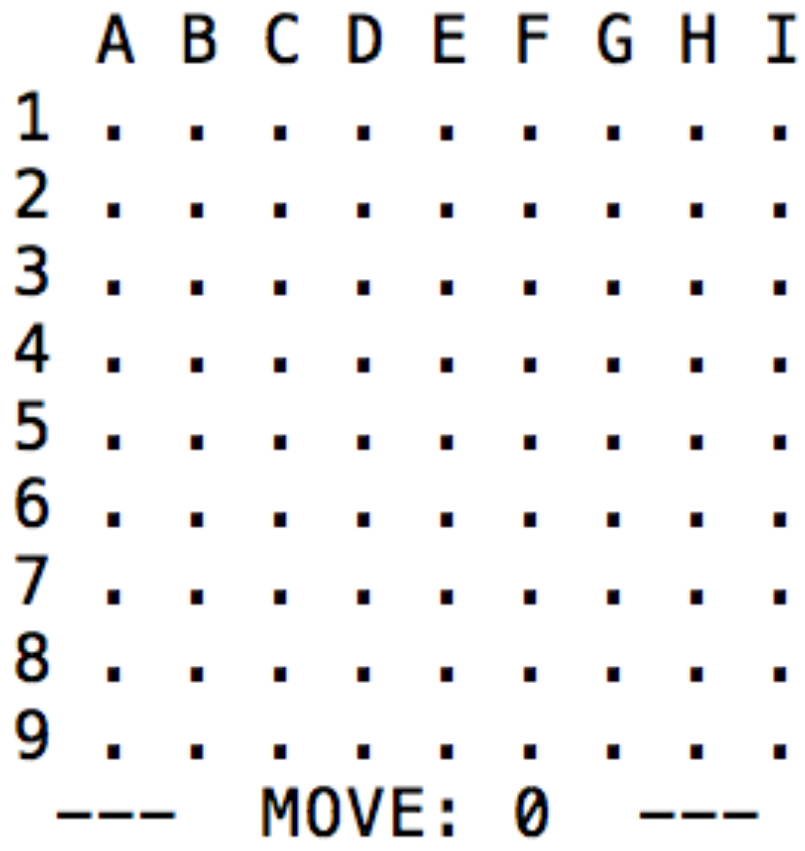
현재까지 진행 상황

- TicTacToe 환경 구현
  - 3x3 보드에 3목을 두면 이기는 심플한 게임
- 기본 MCTS 구현
- MCTS로 TicTacToe 마스터 하기
  - MCTS 600탐색으로 풀림

- 오목 환경 구현
- MCTS로 9x9 보드 오목 풀기
  - MCTS 10만 탐색으로 사람 테스터와 대등
- P, V network + MCTS 구현
- **제로 알고리즘으로 9x9 오목 마스터 하기 (진행 중)**

- 웹 환경 구현
  - 모바일 터치로 랜덤 에이전트와 오목 게임이 가능하도록 프로토타입 구현

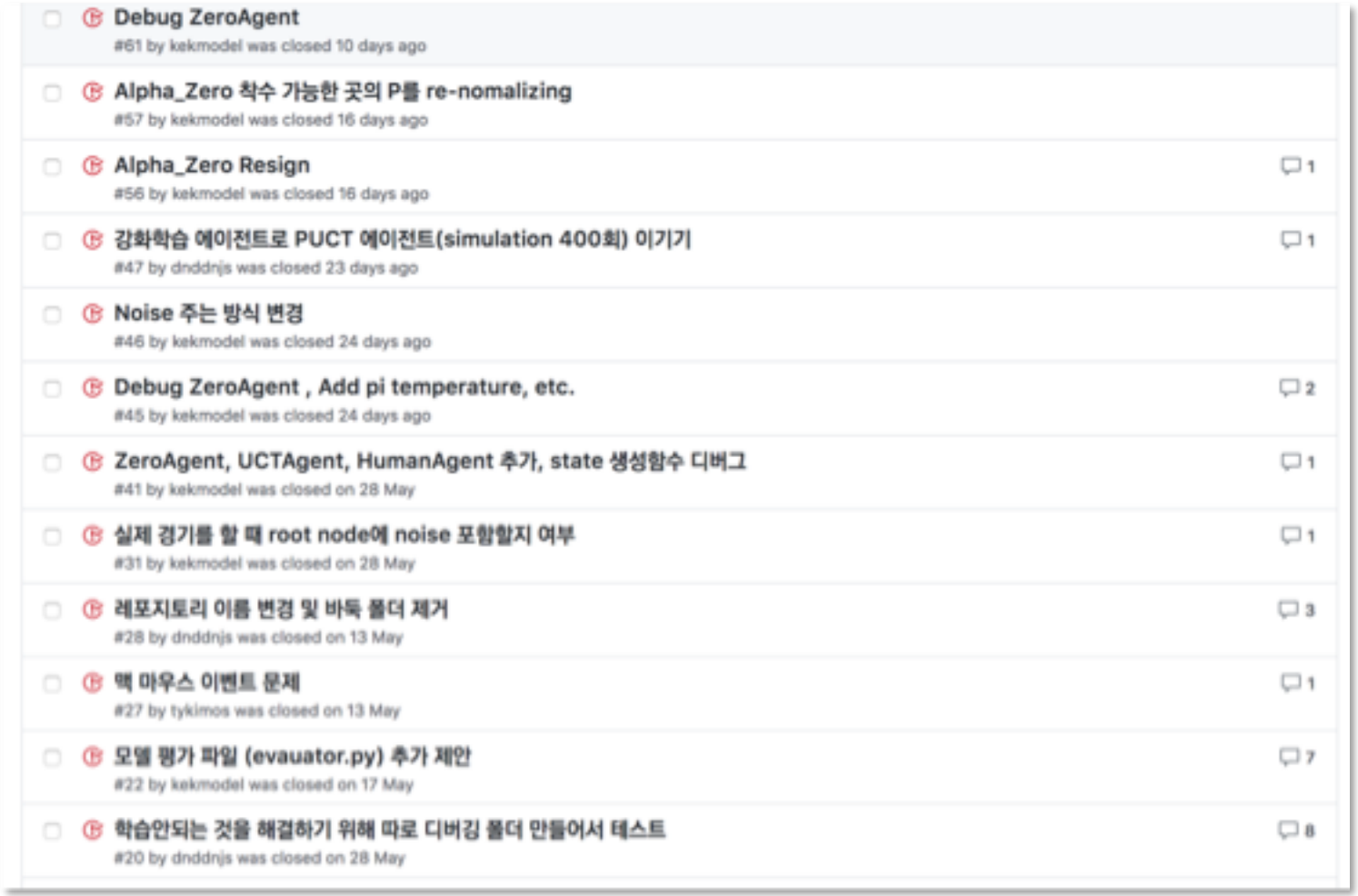
MCTS 10만 탐색



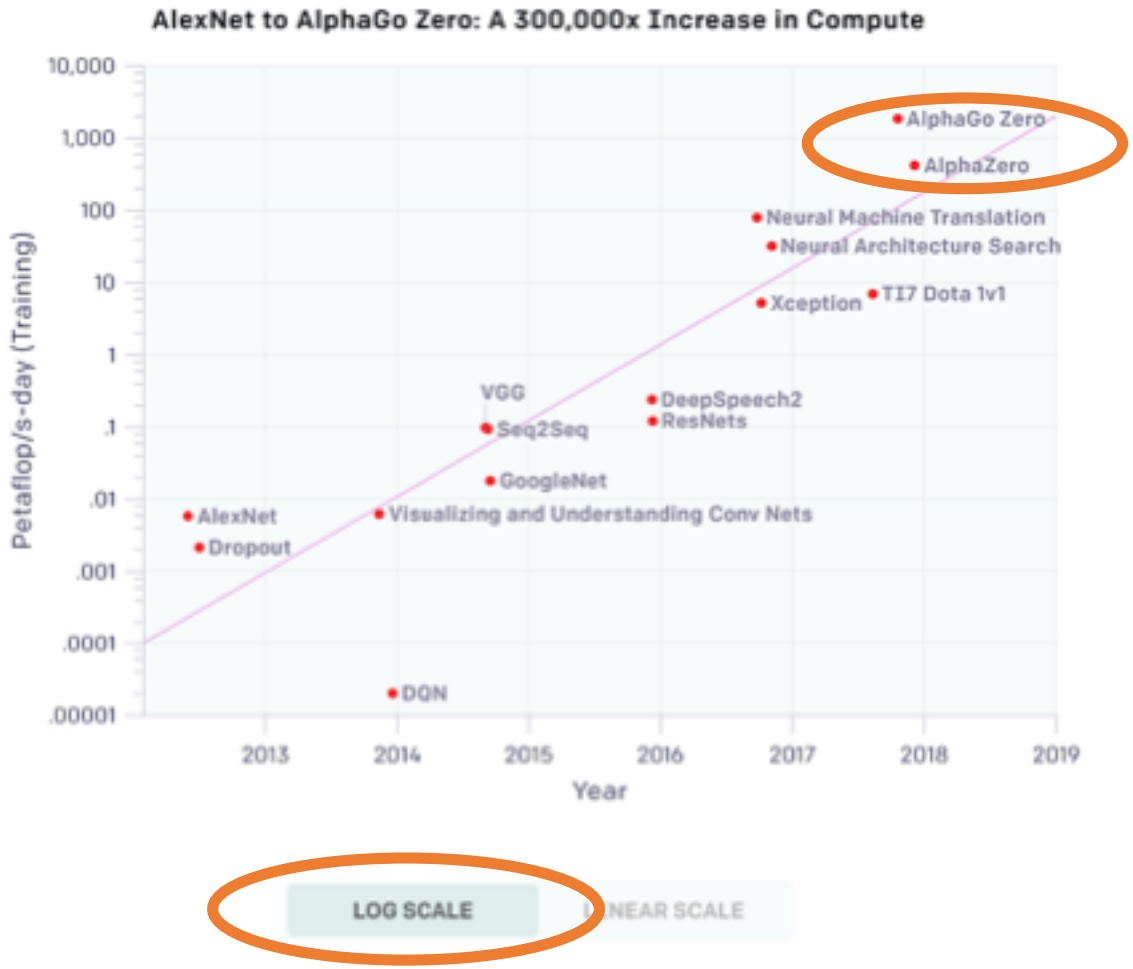
https://github.com/kekmodel/mcts-omok/blob/master/img/omok.gif

이제 Network만 붙이면 금방 풀리겠지...?

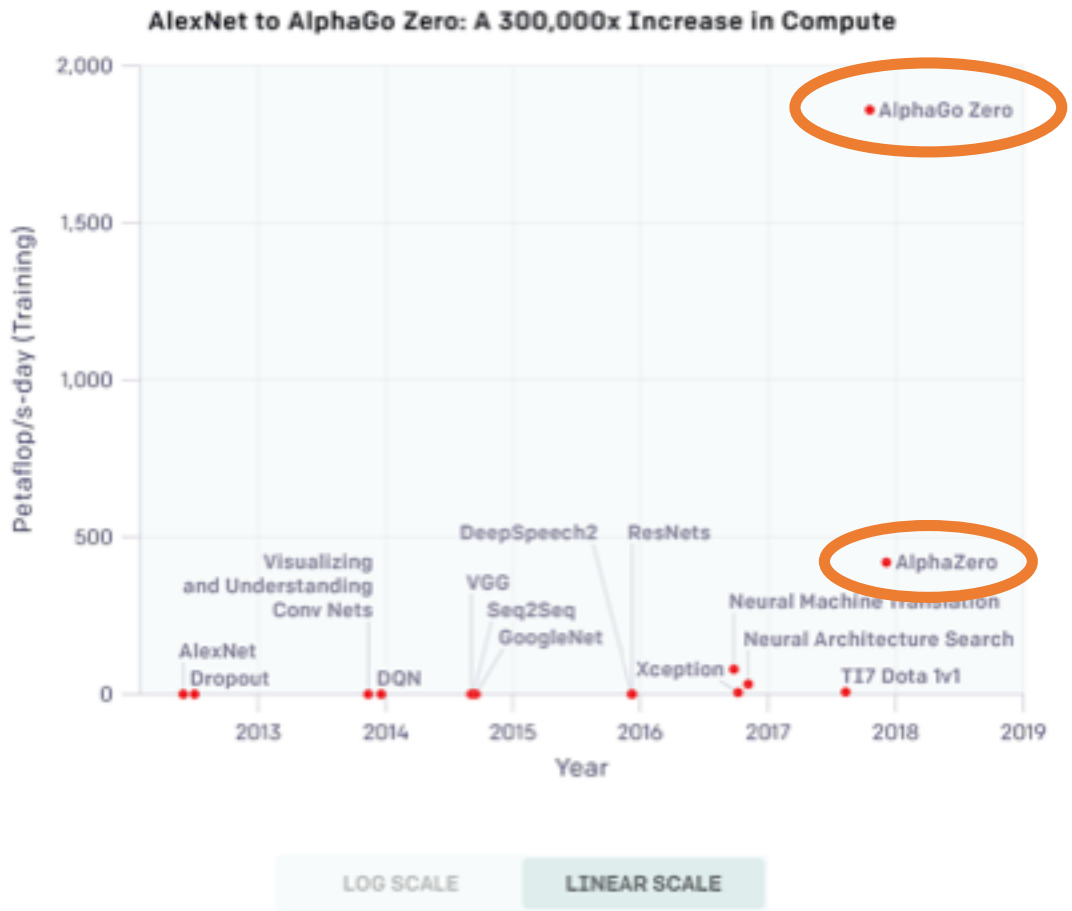
?!


Debug ZeroAgent
#61 by kekmodel was closed 10 days ago
Alpha_Zero 착수 가능한 곳의 P를 re-nomalizing
#57 by kekmodel was closed 16 days ago
Alpha_Zero Resign
#56 by kekmodel was closed 16 days ago
1
강화학습 에이전트로 PUCT 에이전트(simulation 400회) 이기기
#47 by dnddnjs was closed 23 days ago
1
Noise 주는 방식 변경
#46 by kekmodel was closed 24 days ago
Debug ZeroAgent , Add pi temperature, etc.
#45 by kekmodel was closed 24 days ago
2
ZeroAgent, UCTAgent, HumanAgent 추가, state 생성함수 디버그
#41 by kekmodel was closed on 28 May
1
실제 경기를 할 때 root node에 noise 포함할지 여부
#31 by kekmodel was closed on 28 May
1
레포지토리 이름 변경 및 바둑 폴더 제거
#28 by dnddnjs was closed on 13 May
3
맥 마우스 이벤트 문제
#27 by tykimos was closed on 13 May
1
모델 평가 파일 (evauator.py) 추가 제안
#22 by kekmodel was closed on 17 May
7
학습안되는 것을 해결하기 위해 따로 디버깅 폴더 만들어서 테스트
#20 by dnddnjs was closed on 28 May
8

알고리즘 별 Compute



알고리즘 별 Compute



알고리즘을 경량화하는 과정에서 여러 문제 발생

각종 이슈들

- 상대방을 고려하지 않는 이기적인 행동
- 막힘 없는 3목을 잘 못 막음
- 4목인 상황에서 5목을 안놓고 즐김?
- 학습된 모델을 테스트 시 탐색 횟수를 늘리면 오히려 승률 감소
- 알려진 최강 수인 정중앙 첫수를 선택 안함
- 테스트한 알고리즘의 유용성을 평가하려면 적어도 2~3일 소요

대부분 충분히 exploration 하지 않아서 발생한 것으로 보임

시행착오 끝에 현재 밀고있는 설정

- MCTS 횟수
  - 800회 (0.4초) ➔ 400회 (2초)
- 셀프플레이 횟수
  - 25,000판 ➔ 400판 (5시간)
- 히스토리 수
  - 흑, 백 각각 8수까지 ➔ 2수까지
- 신경망 크기
  - 40Block, 256Channel ➔ 10Block, 128Channel
- 노이즈 주는 방식
  - 루트 노드 확장 시 1회 영구적으로
- 학습 방법
  - 최근 50만판까지 저장하고 학습 시 메모리에서 랜덤 샘플링하여 사용

    ➔ 400판만 저장하고 셔플하여 Epoch 방식으로 학습
- 평가 방법
  - 비동기로 1000step 학습 후 이전 모델과 400판 대결
    - 승률 55%가 넘으면 새로운 모델로 업데이트

  ➔ 학습 시 매 Epoch 마다 MCTS없이 Neural net의 P로만 이전 모델과 400판 대결
    - 승률 55%가 넘지 않으면 1 Epoch 더 학습 후 평가
    - 20 Epoch 안에 달성 못할 시 모델을 롤백하여 다시 셀프플레이

시연

3일 학습한 귀여운 모델과 한겜

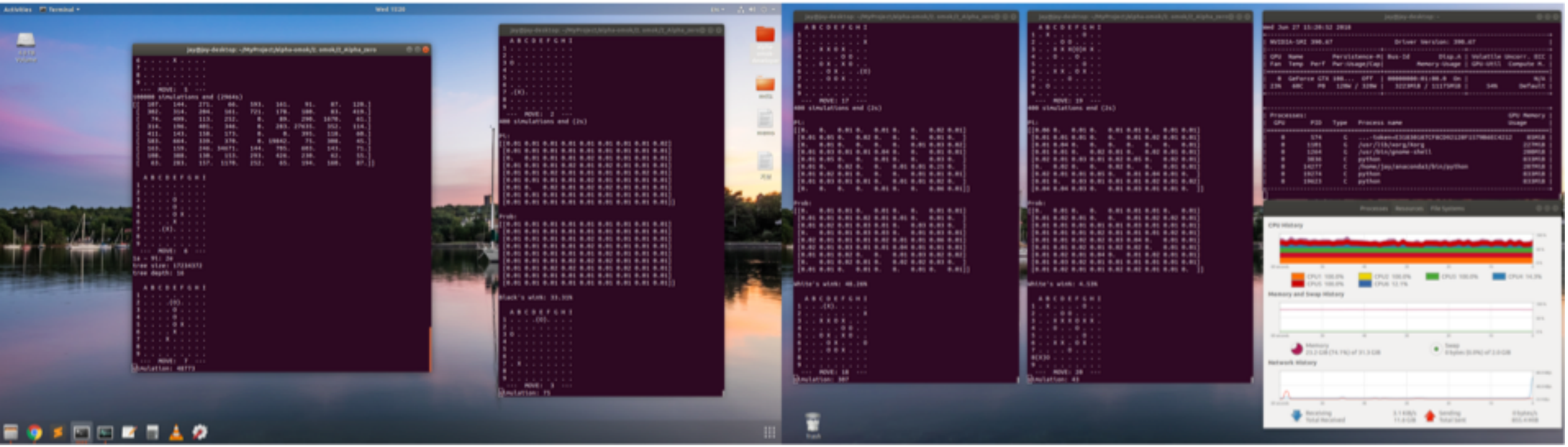
고려해볼 사항들

- MCTS 횟수 늘리기
- 셀프플레이 횟수 늘리기
- 신경망 크기 조정
- 노이즈 주는 방식 변화
- 기권 적용
- tau annealing 조정
- 학습방법 다각화
  - 최신의 방법들 접목시켜 보기
- 코드의 버그 확인

앞으로의 과제

- 효율적인 학습 알고리즘 확정 하기
  - 오목에 알맞는 방법, 1 GPU/ 1 CPU 로 해볼만한 방법
- MCTS를 C++로 포팅
  - 셀프플레이 시간, 메모리 관리에 유리하도록
- MCTS를 병렬로 탐색할 수 있도록 업그레이드
  - 병렬로 탐색 시 실력이 상승한다는 논문이 여럿 있음
- 오목 정식룰인 15x15 보드에 '렌주룰' 을 적용하여 학습
- 적은 자원으로 웹에서 돌릴 수 있는 아이디어

퇴근 후 저의 모습



감사합니다.