

# Dive into MuZero

곽윤혁

# MuZero의 Contribution

- AlphaZero 를 한층 더 General 하게 확장
- 시각적으로 복잡하거나 높은 계획 능력을 요구하는 환경 모두에 적용 가능
  - 바둑, 체스, Shogi 과 Atari 57 환경에서 SOTA 달성
- 환경의 Model을 모르는 상황에도 적용 가능하다
- Single Agent 나 intermediate reward를 제공하는 환경에서도 적용 가능
- representation learning, model learning, planning 로 독립되어있던 과정을 end-to-end 로 통합
  - observation이나 real state를 예측하려 하는 대신, reward와 policy, value를 정확히 예측 하려는 목적하에 hidden state가 학습된다.
  - 즉, 세가지 요소를 예측하는 데 필요없는 것들은 hidden state에 담겨있지 않아도 된다

# 알파고를 들어보셨나요?

- 2016년, 이세돌 기사를 4:1로 이김.
- 세간의 주목을 받음



# 영화도 무료로 풀었어요!

꼭 보시길 추천드립니다.

[AlphaGo - The Movie | DeepMind](#)

# 역사

## 1. Deep Blue by IBM, 1990년대

- 러시아의 체스 챔피언인 Garry Kasparov를 이겼다.
- 휴리스틱, 무작위 탐색법(Brute-force)

## 2. AlphaGo Fan, 2015

- 판후이(Fan Hui), European Professional Go Champion in 2016 를 이겼다.
- 5:0, October 2015

### 3. AlphaGo Lee

- 이세돌(프로 9단)을 상대로 승리
- 4:1, March 2016

### 4. AlphaGo Zero, 2017

- 기보 데이터 사용 X.
- 오직 자가 경기(self-play)만으로 바둑에서 초인간적인(Super-human) 성능 달성

### 5. AlphaZero, 2017

- AlphaGo Zero 논문 발표 후 약 한 달만에 발표.
- 체스, 쇼기 등의 보드게임에서도 초인간적인(Super-human) 성능 달성

### 6. MuZero, 2019

- 바둑, 체스 등의 보드게임과 Atari 57 아케이드 게임에서도 SOTA 달성

## RL에서 주로 사용해온 게임 환경의 두 가지 부류

# 고전 아케이드 게임 VS 보드 게임

- 고전 컴퓨터 아케이드 게임(Atari 57, Doom, ...)
  - 복잡한 룰과 환경 역학 가짐
  - Atari 57에 존재하는 게임들을 해결하는 데 높은 계획 능력은 요구 되지 않음.



- Model-free algorithm 들이 주로 SOTA 달성.

## 고전 아케이드 게임 VS 보드 게임

- 보드 게임(바둑, 체스, 쇼기, ...)
  - 명확하고 단순한 룰과 환경 역학을 가짐
  - 그러나 높은 수준의 계획 능력을 요구

왜 그 중 바둑이 다른 보드 게임들에 비해 더 도전적이었을까요?

## 아주 넓은 탐색 공간

- Connect4: Opening 6 possible moves.
- Chess: Opening 20 possible moves
- Go: Opening 361( $19 \times 19$ ) possible moves

Action space와 state space가 커지거나, terminial state가 흔하지 않을 수록 좋은 판단을 내리기 위해 탐색해야하는 node개수가 엄청나게 늘어난다.

- "무엇을 해도 되고 무엇을 하면 안 되는지는 알겠는데 무엇을 해야 하는지는 모르겠다."
  - 바둑 초보들이 흔히 하는 말.
- **What is the optimal decision in this state?** : 대답하기 힘들다...

## 아주 넓은 탐색 공간

이 때문에 Planning을 하는 주로 Model-based algorithm 들이 보드게임에서 SOTA 달성

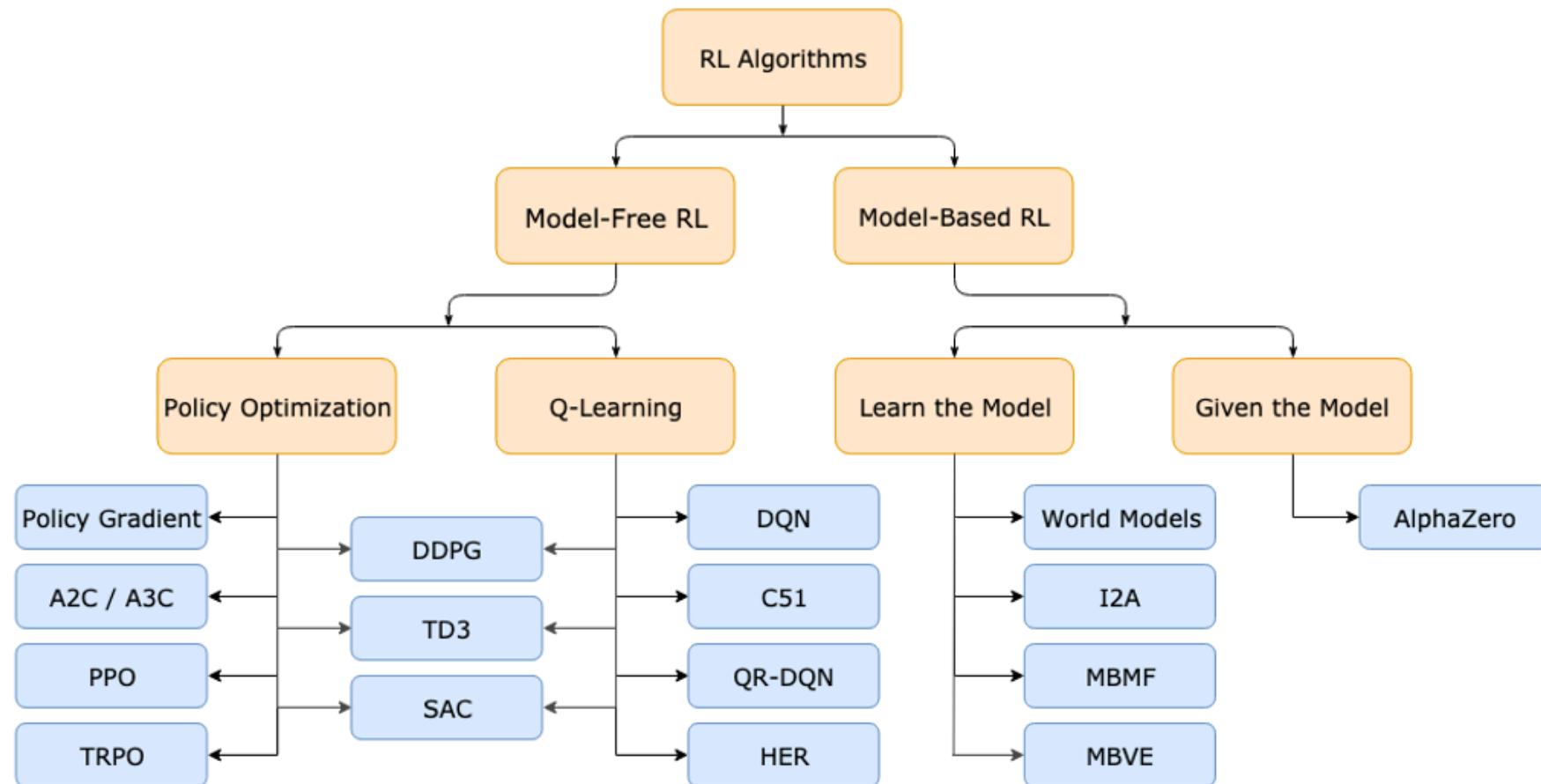
## 왜 AlphaZero, MuZero가 중요할까요?

- 우리의 세상은 단순히 직관으로 판단할 수 없는 것들로 가득합니다.
- 인간의 직관도 매우 부실하죠.
- 하지만 직관은 빠릅니다.
- 숙고/계획은 이성적인 판단을 할 수 있게 만들지만, 느립니다.
- 좋은 결정을 위해서는 직관과 계획 능력 둘 다 필요합니다.

AlphaZero, MuZero는 좋은 결정을 위해

직관과 계획 능력을 둘 다 사용합니다!

# 기존 방법들의 문제점



## 기존 방법들의 문제점

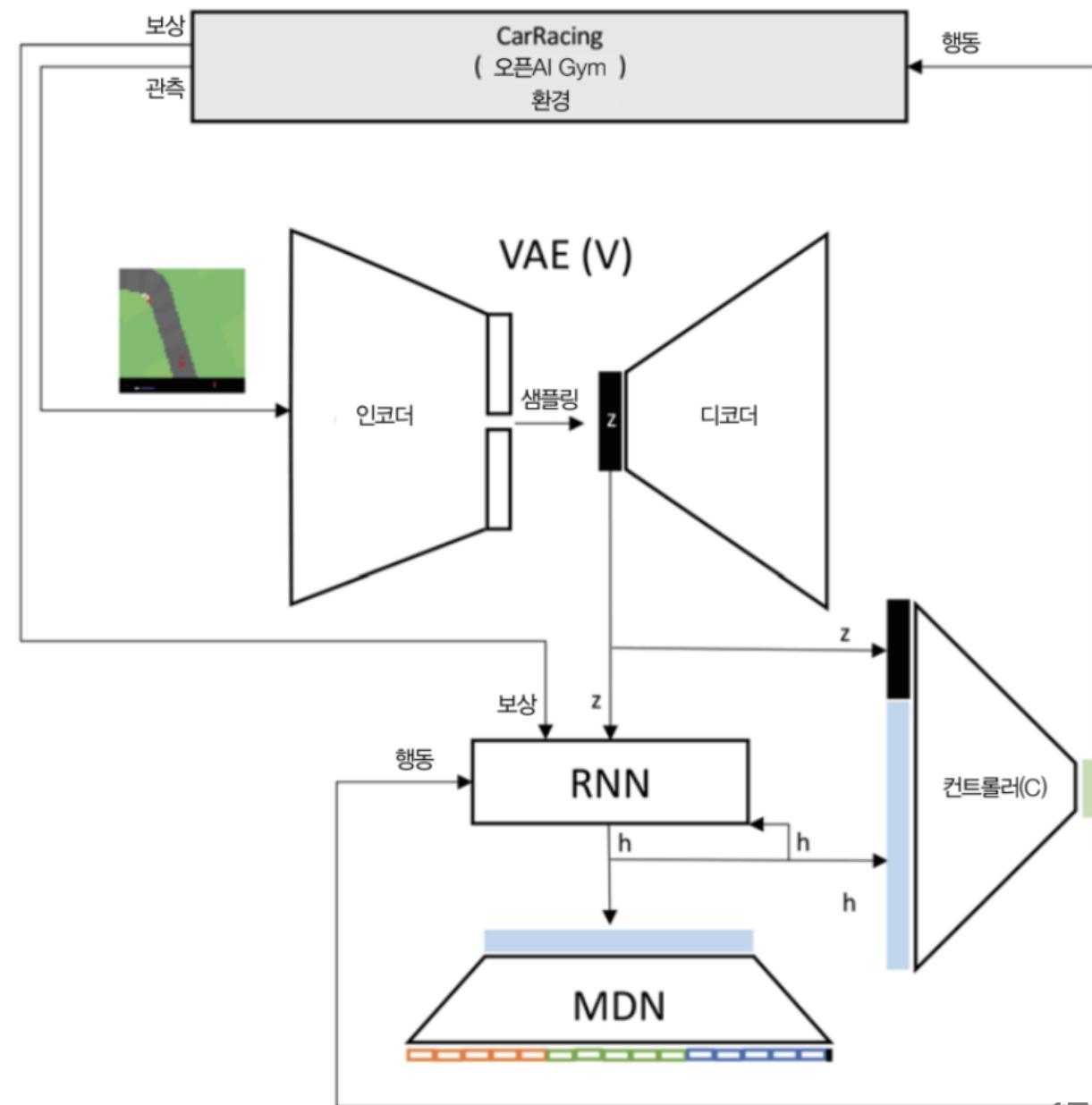
### AlphaZero

- two-player 상황에서만 동작
- intermediate reward를 주는 환경에서는 사용 불가
- Perfect simulator가 있어야지만 적용 가능

# World Models

- 세 개로 분리되어 있는 학습 과정
  - representation learning,  
model learning, planning  
세 가지 부분으로 나눠져 있다.
- 성능 향상에 필요 없는 표현과 모델도  
학습하게 되어,  
planning 과정에서 에러가 쌓인다.  
(compound)
- 결국 쌓인 에러 때문에  
data efficiency 측면에서도  
model-free 방법론들에서  
밀리게 된다.

그림 8-19 오픈AI Gym 환경에서 컨트롤러 훈련



## 기존 방법들의 문제점

### 그 외 기존 방법들

- Model-free 알고리즘들
  - Atari 게임들에 잘 적용되었으나,  
이 게임들의 대부분들은 높은 계획 능력을 요구하지 않았다.
  - 높은 계획 능력을 요구하는 보드 게임엔 잘 적용되지 않았다.

# 기존 방법들의 문제점

## 그 외 기존 방법들

- Model-based 알고리즘들
  - 3가지의 단계로 나누어져 있었다.(representation learning, model learning, planning)
    - observation을 활용해서 환경의 다음 state를 예측하고자 했던 방법들은 성능을 올리는 데 관련 없는 부분들까지도 예측하는데 모델 capacity의 대부분이 사용되어서 잘 되지 않았다.
    - observation 그 자체를 예측하는 모델은 compounding error 때문에 잘 되지 않았음.
      - Deep하고 stochastic한 모델이 이 문제를 완화해줄 것이라 예상했으나 잘 안되었다.

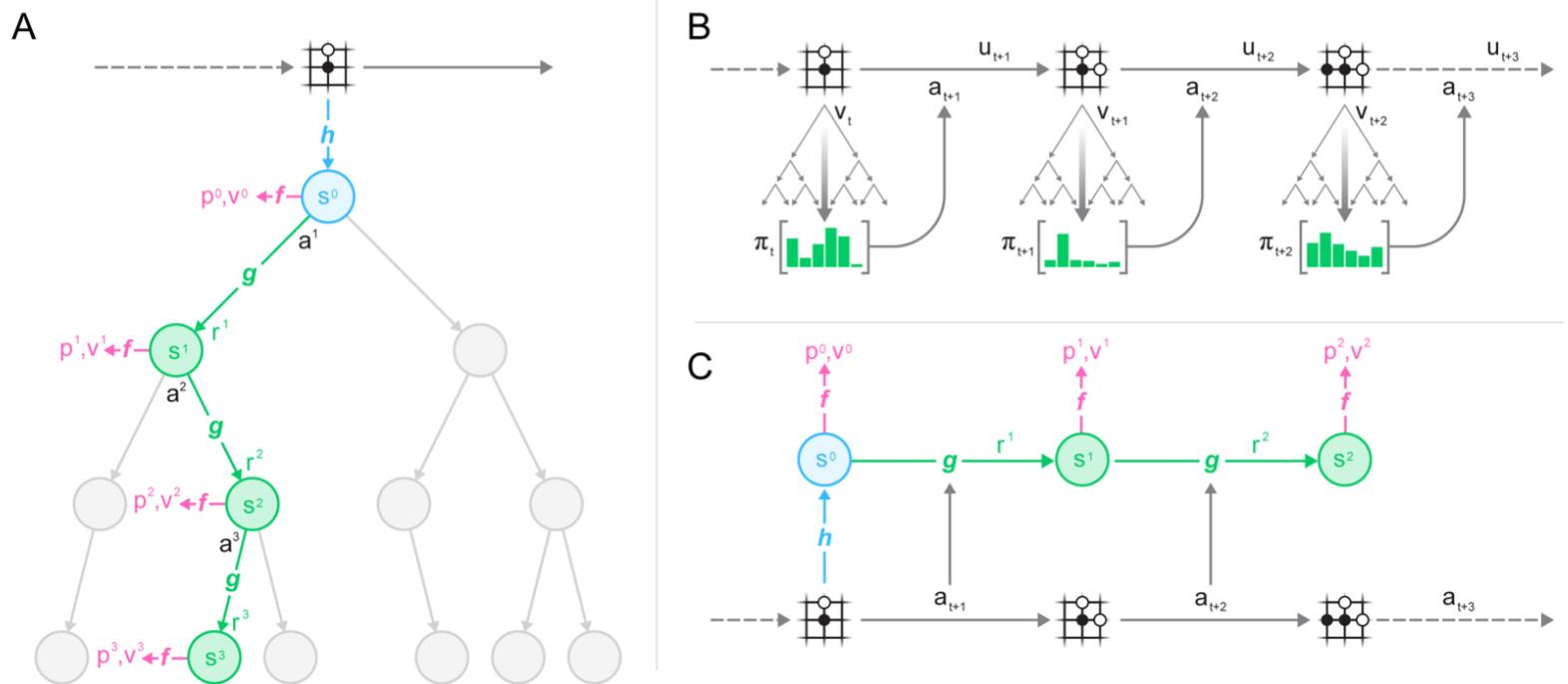
## MuZero의 key idea

- Value Equivalence가 보장된 abstract model을 학습
  - 강화학습의 목적은 보상 합의 최대화이기 때문에
- Raw observation 그대로 사용 X. hidden state를 사용.
- end-to-end로 policy, value, immediate reward를 예측하는 데 중요한 것들만 학습.
- 학습된 모델을 가지고 MCTS로 Planning 진행

자세히 들여다봅시다.

# Three-stage

1. Planning [A]
2. Acting in environment [B]
3. Training [C]

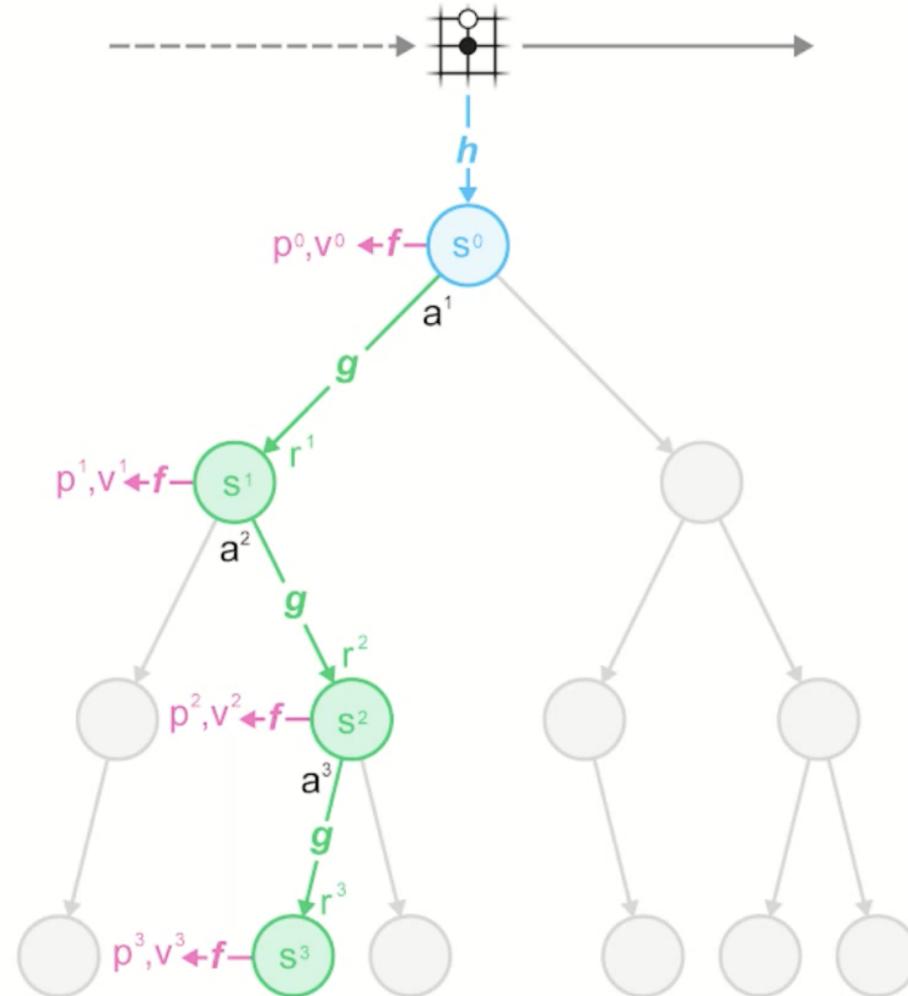


# Planning with a Learned Model

representation     $s^0 = h_\theta(o_1, \dots, o_t)$

prediction     $p^k, v^k = f_\theta(s^k)$

dynamics     $r^k, s^k = g_\theta(s^{k-1}, a^k)$



# Planning with a Learned Model

## Selection

- 각 edge에는  $\{N(s,a), Q(s,a), P(s,a), R(s,a), S(s,a)\}$ 가 저장된다.
  - 방문 횟수 N, mean value Q, policy P, reward R, and hidden state S
- 아래의 식(UCB)에 따라 탐색할 branch를 선택

$$a^k = \arg \max_a \left[ Q(s, a) + P(s, a) \cdot \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)} \left( c_1 + \log \left( \frac{\sum_b N(s, b) + c_2 + 1}{c_2} \right) \right) \right]$$

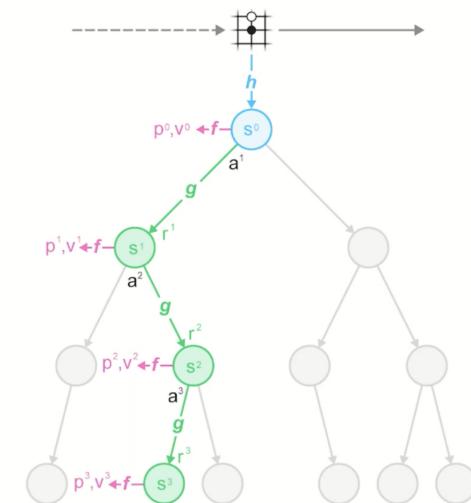
- $c1 = 1.25$  and  $c2 = 19652$ .

## Planning with a Learned Model

representation  $s^0 = h_\theta(o_1, \dots, o_t)$

prediction  $p^k, v^k = f_\theta(s^k)$

dynamics  $r^k, s^k = g_\theta(s^{k-1}, a^k)$



# Planning with a Learned Model

## Expansion

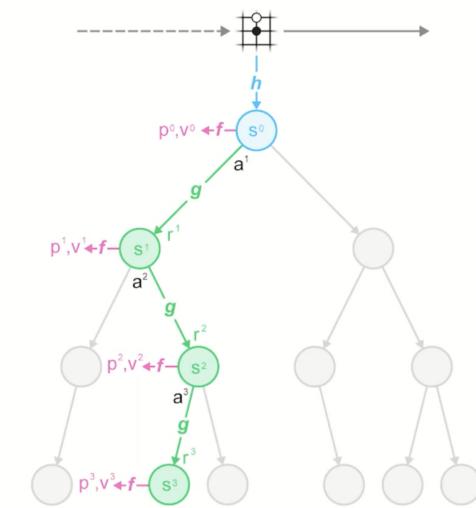
- leaf node에 도달 시 진행한다
- leaf node의 자식 노드들로 연결될 엣지들 생성
  - 각 엣지들에 대해, inference 진행.  
(사실 inference는 네트워크마다 leaf node에 대해 한번만 진행된다.)
  - $r, s = g(s, a), p, v = f(s)$
  - $N(s, a) = 0, Q(s, a) = 0, P(s, a) = p,$   
 $R(s, a) = r, S(s, a) = s$ 로 각 엣지 초기화

## Planning with a Learned Model

representation     $s^0 = h_\theta(o_1, \dots, o_t)$

prediction     $p^k, v^k = f_\theta(s^k)$

dynamics     $r^k, s^k = g_\theta(s^{k-1}, a^k)$



# Planning with a Learned Model

## Backup

- Expansion이 발생하여 simulation이 끝나면 Backup을 진행한다
- root node까지 거슬러 올라가면서 아래의 식에 따라 Q와 N을 갱신한다

$$G^k = \sum_{\tau=0}^{l-1-k} \gamma^\tau r_{k+1+\tau} + \gamma^{l-k} v^l$$

For  $k = l \dots 1$ , we update the statistics for each edge  $(s^{k-1}, a^k)$  in the simulation path as follows,

$$Q(s^{k-1}, a^k) := \frac{N(s^{k-1}, a^k) \cdot Q(s^{k-1}, a^k) + G^k}{N(s^{k-1}, a^k) + 1}$$

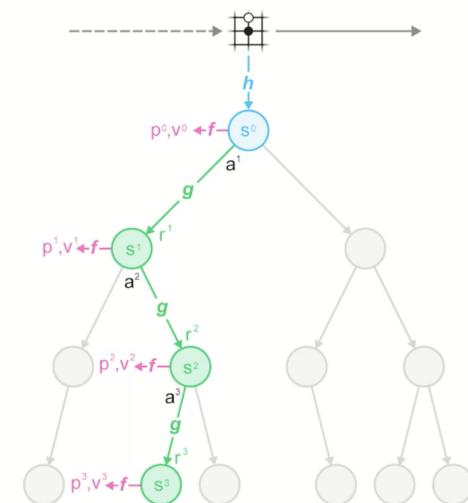
$$N(s^{k-1}, a^k) := N(s^{k-1}, a^k) + 1$$

## Planning with a Learned Model

representation  $s^0 = h_\theta(o_1, \dots, o_t)$

prediction  $p^k, v^k = f_\theta(s^k)$

dynamics  $r^k, s^k = g_\theta(s^{k-1}, a^k)$



# Planning with a Learned Model

1. Selection
2. Expansion
3. Backup

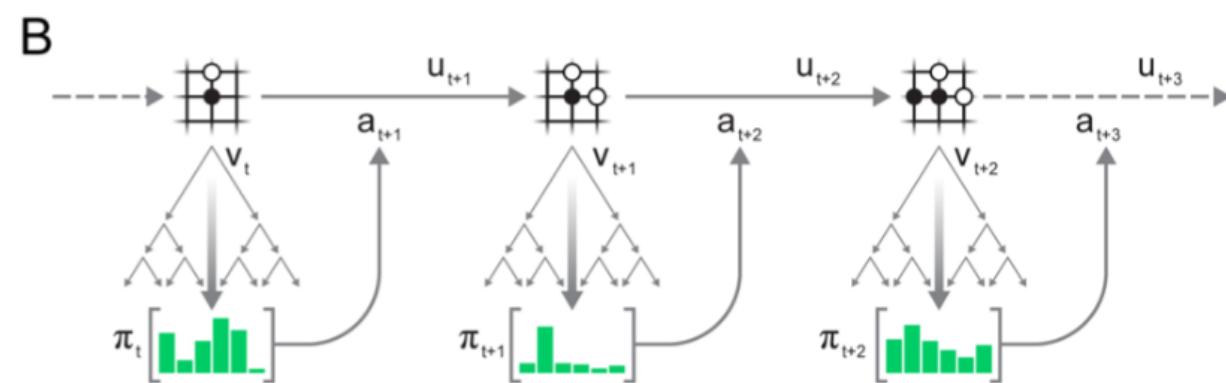
위의 과정이 모두 진행되면 simulation 한번은 완료한 것이다.  
이 과정(simulation)을 여러번 반복하며 경험을 쌓는다.

# Acting in environment

- MCTS를 통해 개선한 policy로부터 action을 샘플링하여 행동
- 각 branch를 방문한 횟수를 통해 policy 정의

$$p_\alpha = \frac{N(\alpha)^{1/T}}{\sum_b N(b)^{1/T}}$$

- T: Temperature parameter
- 학습을 진행하면서 점점 decay 된다.(1 => 0.25)
  - 점점 greedy해진다.



# Training

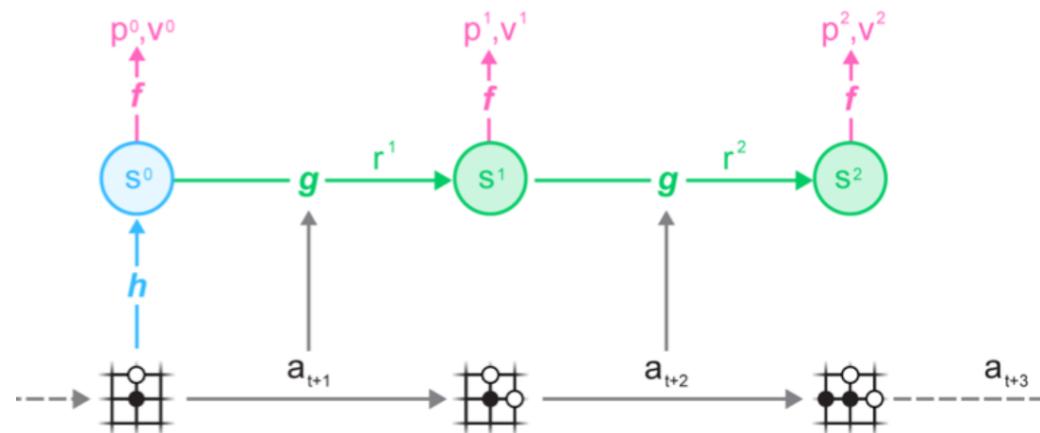
- Replay Memory에서 Sequence를 추출.

Learning Rule

$$\mathbf{p}_t^k, v_t^k, r_t^k = \mu_\theta(o_1, \dots, o_t, a_{t+1}, \dots, a_{t+k})$$

$$z_t = \begin{cases} u_T & \text{for games} \\ u_{t+1} + \gamma u_{t+2} + \dots + \gamma^{n-1} u_{t+n} + \gamma^n v_{t+n} & \text{for general MDPs} \end{cases}$$

$$l_t(\theta) = \sum_{k=0}^K l^r(u_{t+k}, r_t^k) + l^v(z_{t+k}, v_t^k) + l^p(\pi_{t+k}, p_t^k) + c\|\theta\|^2$$



## MuZero Reanalyze

- Sampling Efficiency를 높이기 위해 고안된 실험
- 학습을 진행하면서 이전에 수집된 trajectory의 target policy와 target value를 최근 model을 사용하여 다시 갱신한다.

# Experiment

Agent	Median	Mean	Env. Frames	Training Time	Training Steps
Ape-X [18]	434.1%	1695.6%	22.8B	5 days	8.64M
R2D2 [21]	1920.6%	4024.9%	37.5B	5 days	2.16M
<i>MuZero</i>	<b>2041.1%</b>	<b>4999.2%</b>	20.0B	12 hours	1M
IMPALA [9]	191.8%	957.6%	200M	–	–
Rainbow [17]	231.1%	–	200M	10 days	–
UNREAL <sup>a</sup> [19]	250% <sup>a</sup>	880% <sup>a</sup>	250M	–	–
LASER [36]	431%	–	200M	–	–
<i>MuZero Reanalyze</i>	<b>731.1%</b>	<b>2168.9%</b>	200M	12 hours	1M

Table 1: **Comparison of *MuZero* against previous agents in Atari.** We compare separately against agents trained in large (top) and small (bottom) data settings; all agents other than *MuZero* used model-free RL techniques. Mean and median scores are given, compared to human testers. The best results are highlighted in **bold**. *MuZero* sets a new state of the art in both settings. <sup>a</sup>Hyper-parameters were tuned per game.

# AlphaZero와의 차이점

- State transition: not perfect simulator. hidden state 를 사용하여 space 크기를 줄인다
- Available actions을 오직 root node에서만 직접 알 수 있다.  
training set의 trajectory에서 한번도 등장하지 않은 action들은 행하지 않도록 모델은 각 노드에서 불가능한 action들을 빠르게 학습한다.
- Terminal nodes  
AlphaZero에서는 종료조건을 만족하는 node(terminal node)에 도착하면 Tree search를 멈췄지만 MuZero에서는 terminal node에서 멈추지 않고 더 탐색이 진행될 수도 있다.  
terminal node이후의 node들에서는 항상 모두 같은 value를 예측하는 것이 기대된다.
- Single agent 에서 사용가능: AlphaZero에서는 two-player 상황에서만 동작하였다
- Intermediate reward를 주는 환경에서도 사용 가능: AlphaZero에서는 terminal state에 도달했을 때만 undiscounted reward가 1또는 -1로 주어졌다

# Future works

- 여전히 비싼 비용
  - 40개의 TPU를 사용해서 게임 당 12시간 동안 훈련
    - 1 TPU의 성능: 약 4 GPU
    - 하나의 GPU를 사용하여 게임 하나에 훈련시킬 때 예상 소요 시간: 약 80 일 ( $4 * 40 * 0.5$ )
    - Atari '57' \* 80 = 4560일 = 12.5년!
- Partially observable한 환경에 적용 잘 안된다
- Continuous action space 인 경우 적용 불가능
- Dynamics function이 deterministic하게 디자인 되었다.
  - stochastic transition이 심하게 일어나는 환경에서는 잘 적용되지 않을 것으로 예상된다.
- Exploration
  - Montezum's Revenge, pitfall 등 탐험이 중요한 환경에서 0점을 획득하였다.

## References

1. Schrittwieser et al. , 2020, Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model
2. Julian Schrittwieser - [youtube.com/watch?v=vt5jOSy7cz8](https://youtube.com/watch?v=vt5jOSy7cz8)

감사합니다