DMQA Seminar 20240705

Agent57 Lineage

Outperforming Humans with Reinforcement Learning Agents in Atari Games

일반대학원 산업경영공학과
김재훈

고려대학교
KOREA UNIVERSITY

Introduction

- 발표자 소개

- 이름: 김재훈
- 학력
  - ✓ 2020.03 – 현재 | 석박사통합과정 | 고려대학교 산업경영공학과 (지도교수: 김성범)
- 연구분야
  - ✓ Self-supervised learning
  - ✓ Reinforcement learning
- e-mail : jhoon0418@korea.ac.kr

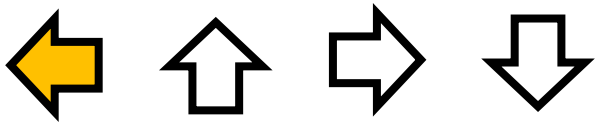
Deep Reinforcement Learning

Objective of RL

❖ 강화학습의 목적

- 강화학습은 **순차적인 의사결정 문제**에서 **누적 보상을 최대화**하기 위해 시행착오를 거쳐서 상황에 따른 행동 정책을 학습
- 강화학습은 에이전트가 속한 상태(state), 선택한 행동(action), 행동에 따른 보상(reward)으로 구성됨

$State$ $Action$ $Reward$



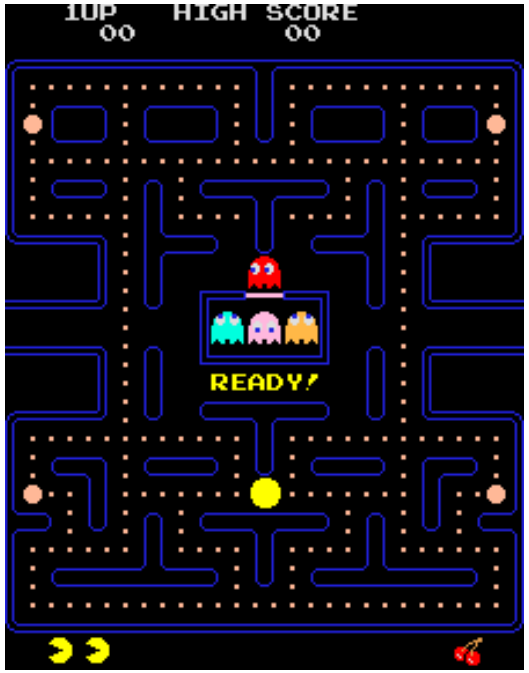
-10 / 0 / 1

Deep Reinforcement Learning

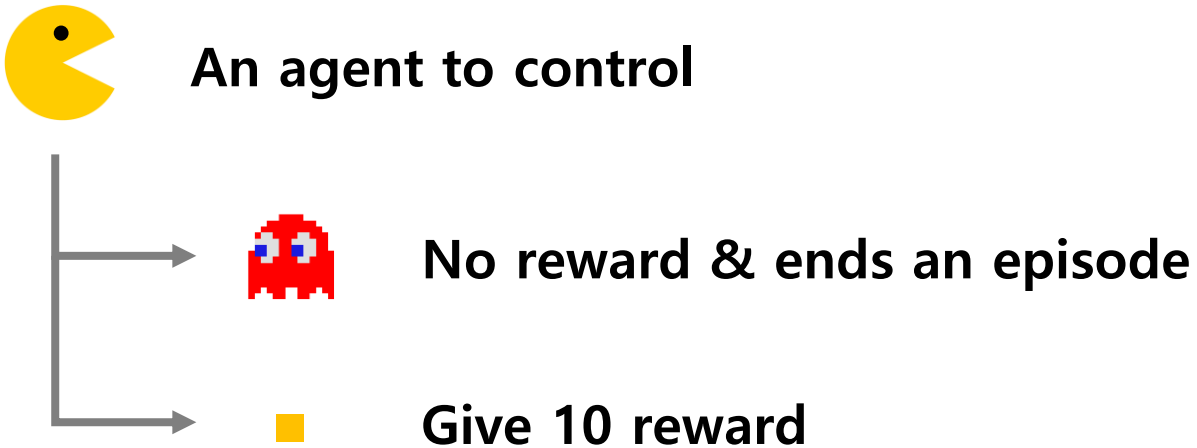
Objective of RL

❖ 예시: PAC-MAN

- 주어진 상태에서 팩맨을 조종하여 유령은 피하되 노란 점을 최대한 많이 수집하는 것이 목표



PAC-MAN

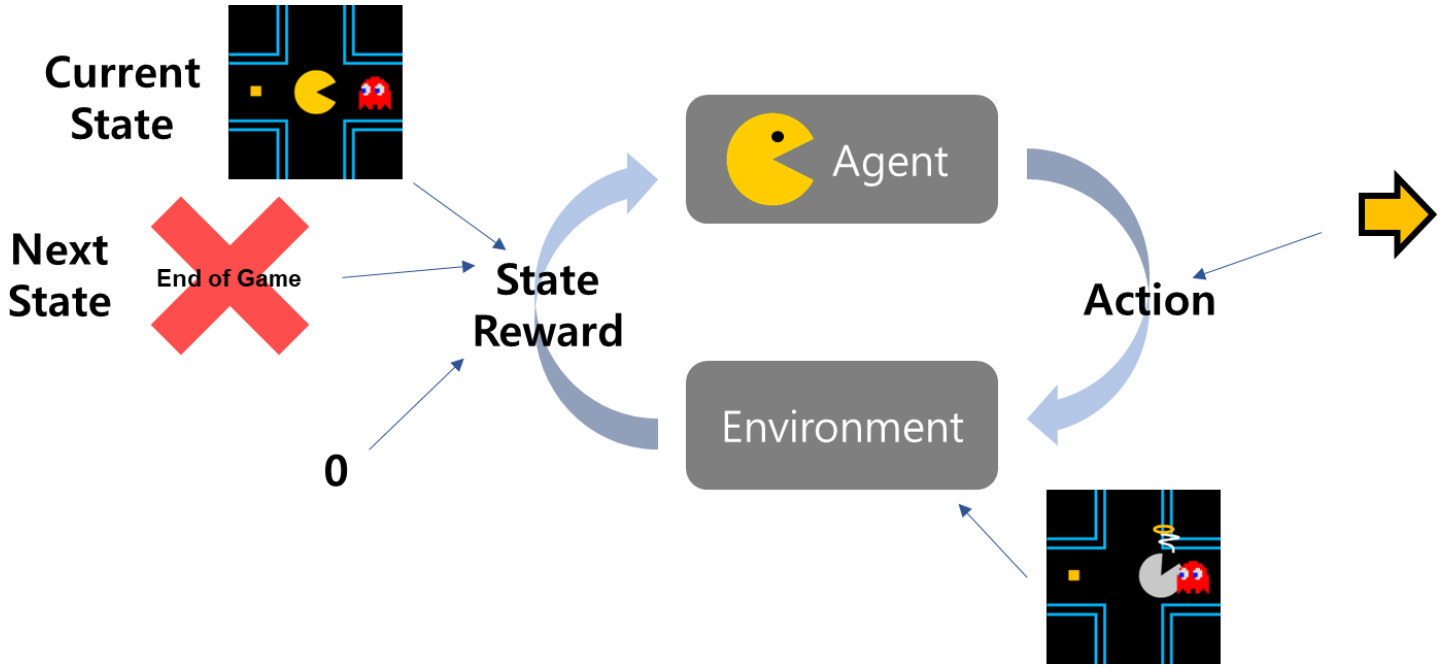


Deep Reinforcement Learning

Objective of RL

❖ 예시: PAC-MAN

- 에이전트를 어떻게 조종하는가에 따라서 달성할 수 있는 최종 점수가 달라짐
- 따라서 에이전트가 좋은 행동 정책을 습득하는 것이 매우 중요함

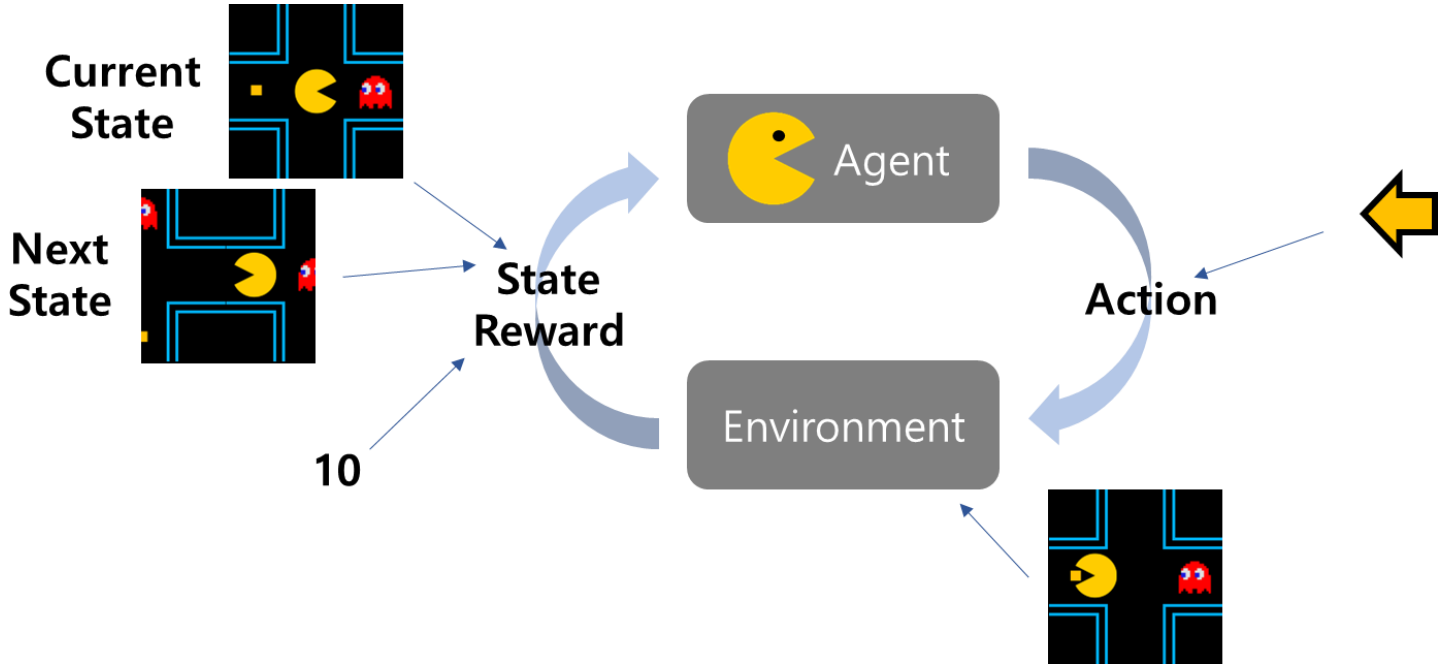


Deep Reinforcement Learning

Objective of RL

❖ 예시: PAC-MAN

- 에이전트를 어떻게 조종하는가에 따라서 달성할 수 있는 최종 점수가 달라짐
- 따라서 에이전트가 좋은 행동 정책을 습득하는 것이 매우 중요함



Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ **The Arcade Learning Environment: An Evaluation Platform for General Agents (Journal of Artificial Intelligence 2013, 3523회 인용)**

- 유명한 콘솔 게임인 Atari 2600에서 57개의 게임을 선정하여 만든 강화학습 벤치마크 환경
- 에이전트는 이미지 형식으로 게임 화면을 입력 받아 이산 행동 공간에서 행동을 선택

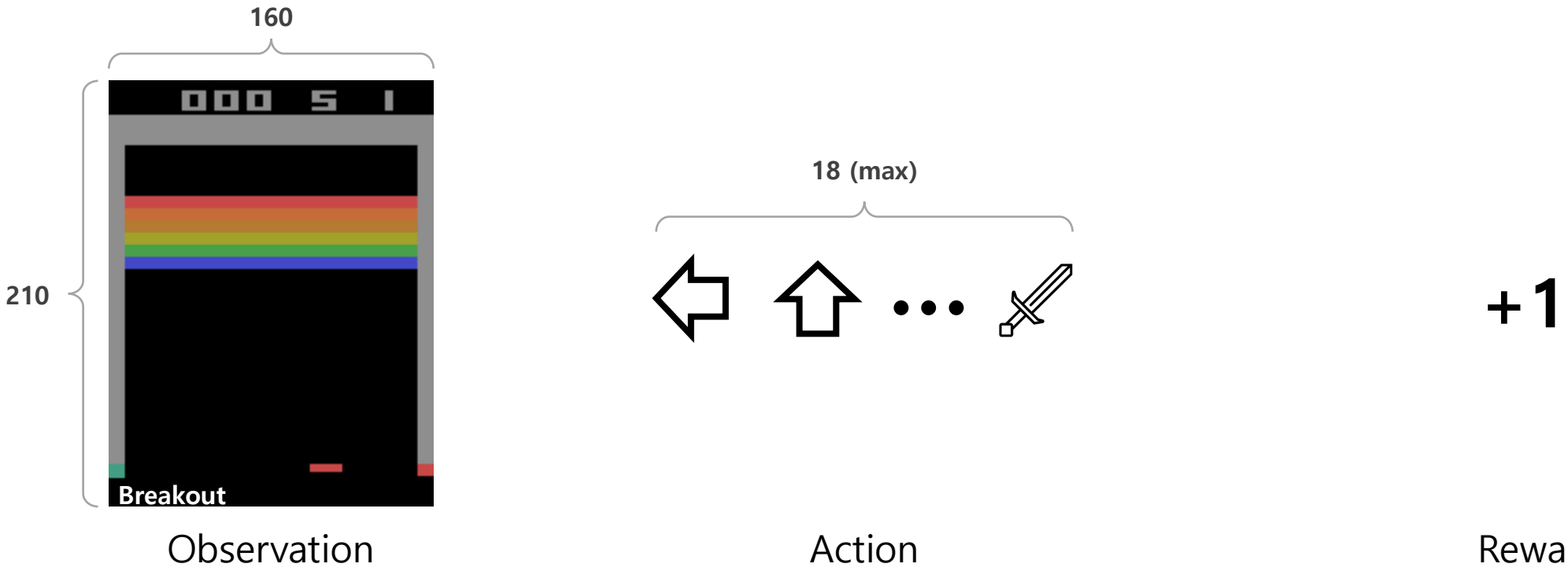
https://gymnasium.farama.org/main/environments/atari/

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ **The Arcade Learning Environment: An Evaluation Platform for General Agents (Journal of Artificial Intelligence 2013, 3523회 인용)**

- 모든 게임 화면 이미지의 크기는 너비 160 픽셀, 높이 210 픽셀로 고정되어 있음
- 게임 종류마다 행동의 종류는 달라질 수 있으며 최대 18가지로 구분됨
- 게임 종류마다 주어지는 보상의 크기와 시기 역시 모두 다름

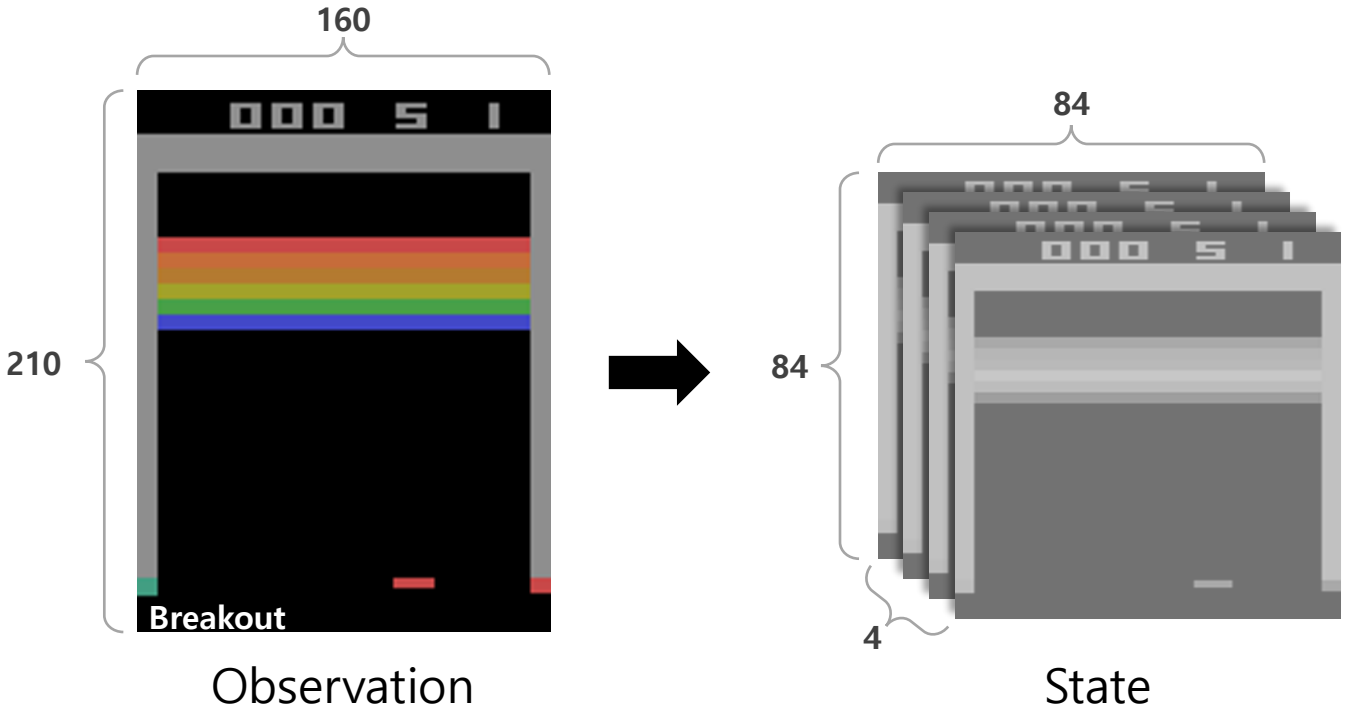


Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크에서 강화학습 실험 시 적용되는 전처리 및 입력 양식*

- 강화학습 모델에 입력될 때 이미지는 크기가 (210, 160)에서 (84, 84)로 축소되며 Y채널의 값만 추출해서 사용함
- 또한 연속하는 네 개의 프레임을 하나의 상태로서 취급* (또는 여러 프레임을 순환 신경망에 입력**하여 사용)
- 강화학습 모델이 결정한 행동은 네 번 반복해서 환경에 입력



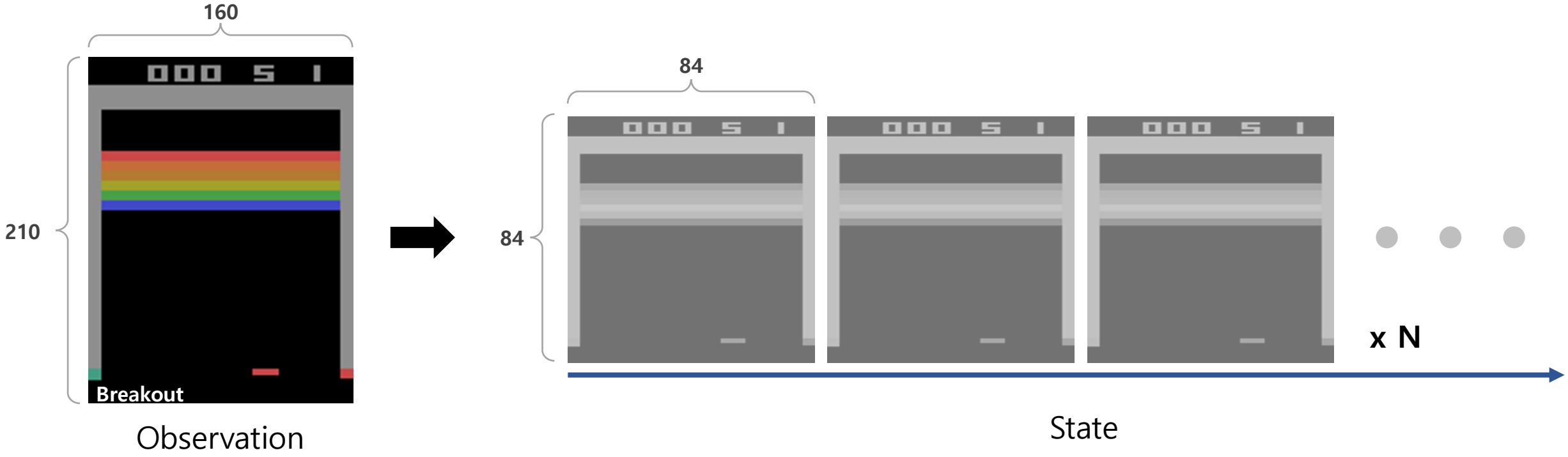
* Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.
** Hausknecht, M., & Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In 2015 aaai fall symposium series.

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크에서 강화학습 실험 시 적용되는 전처리 및 입력 양식 *

- 강화학습 모델에 입력될 때 이미지는 크기가 (210, 160)에서 (84, 84)로 축소되며 Y채널의 값만 추출해서 사용함
- 또한 연속하는 네 개의 프레임을 하나의 상태로서 취급 (또는 여러 프레임을 순환 신경망에 입력**하여 사용)
- 강화학습 모델이 결정한 행동은 네 번 반복해서 환경에 입력



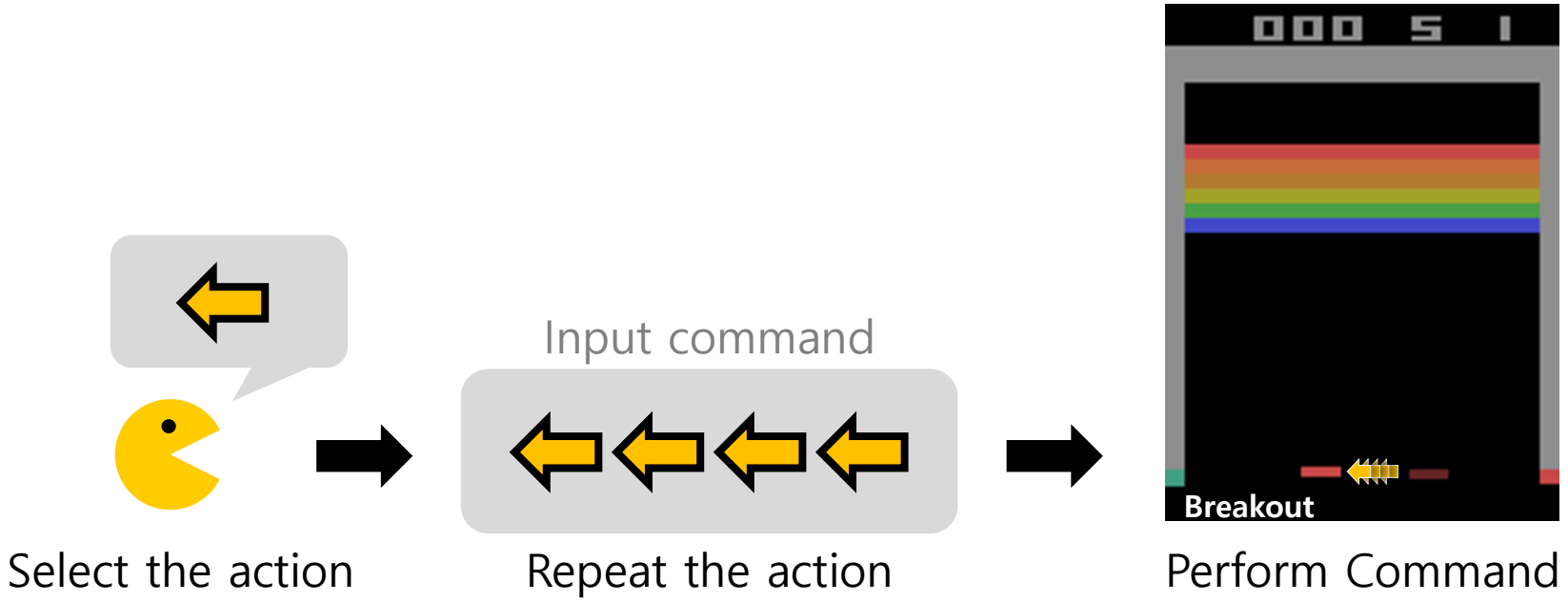
* Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.
** Hausknecht, M., & Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In 2015 aaai fall symposium series.

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크에서 강화학습 실험 시 적용되는 전처리 및 입력 양식 *

- 강화학습 모델에 입력될 때 이미지는 크기가 (210, 160)에서 (84, 84)로 축소되며 Y채널의 값만 추출해서 사용함
- 또한 연속하는 네 개의 프레임을 하나의 상태로서 취급 (또는 여러 프레임을 순환 신경망에 입력**하여 사용)
- 강화학습 모델이 결정한 행동은 네 번 반복해서 환경에 입력



* Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.
** Hausknecht, M., & Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In 2015 aaai fall symposium series.

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크를 통한 모델 간의 성능 비교 방법

- 성능 비교는 각 게임에서 얻은 점수를 일반화하여 진행
- 사람이 플레이 해서 얻은 점수와 랜덤하게 움직였을 때 얻은 점수를 기반으로 점수를 일반화 (human normalized score; HNS)

$$\text{HNS} = \frac{\text{Agent score} - \text{Random score}}{\text{Human score} - \text{Random score}}$$

Extended Data Table 2 | Comparison of games scores obtained by DQN agents with methods from the literature[12,15] and a professional human games tester

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%

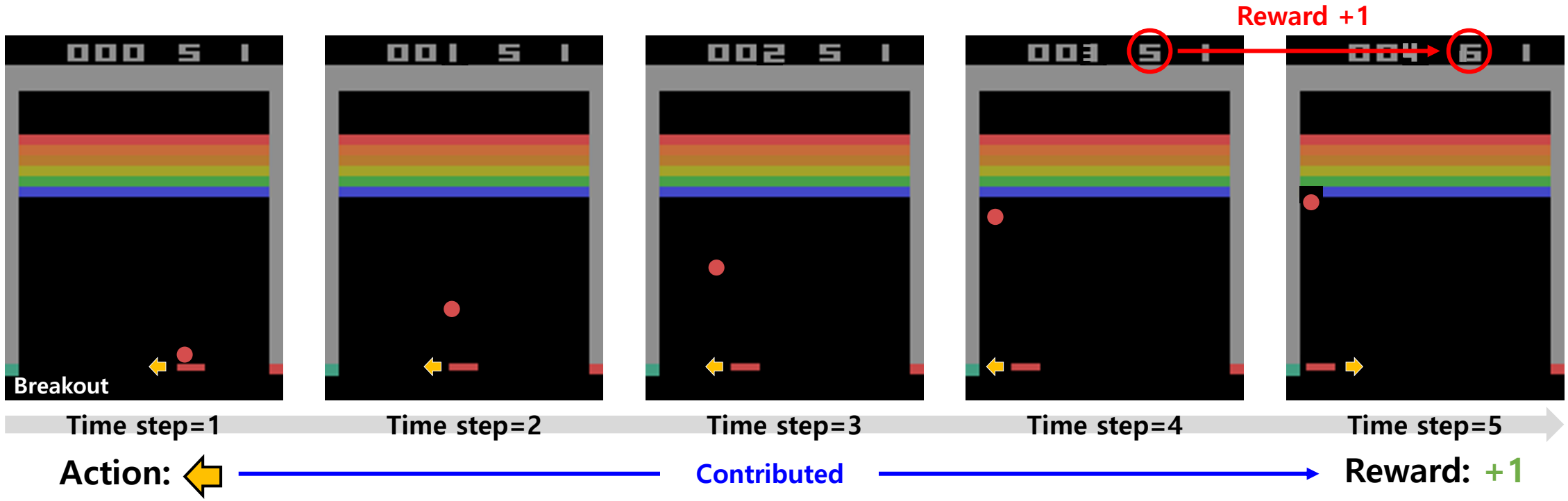
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). Human-level control through deep reinforcement learning. nature, 518(7540), 529-533.

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크의 주요 난제 – Long-term credit assignment

- 어떤 행동이 보상을 얻는 데에 기여를 크게 했는지 정하는 문제
- 에이전트가 취한 행동에 대한 보상이 바로 부여되지 않아 이에 기여한 행동을 추정하는 것이 필요

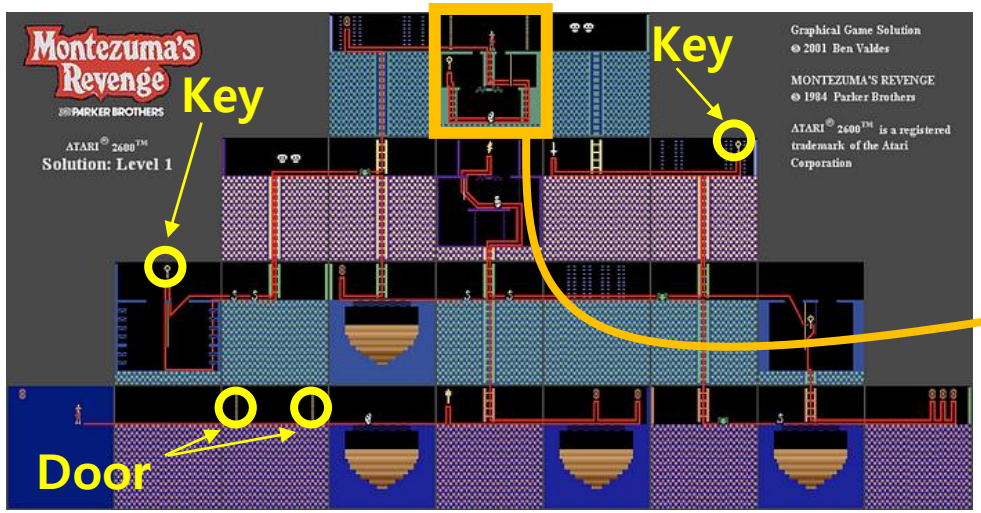


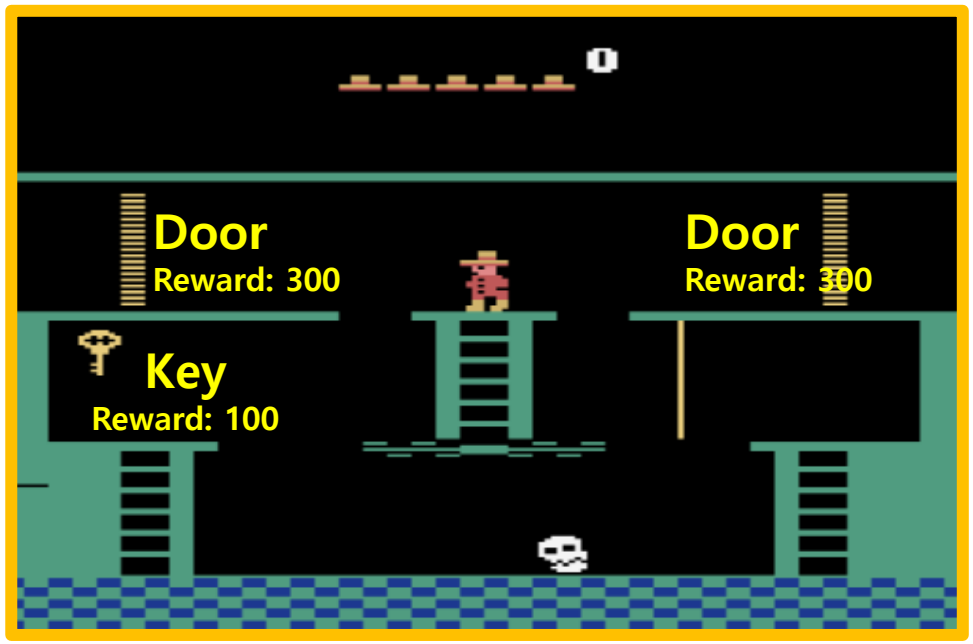
Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ Atari 벤치마크의 주요 난제 – Hard exploration

- 환경에서 주는 보상이 매우 희소하여 강화학습 모델이 적절한 피드백을 받기 어려운 상황
- 희소한 보상으로 인해 에이전트가 최적의 정책을 학습하기 위한 난이도가 높으며 시간이 매우 오래 걸림





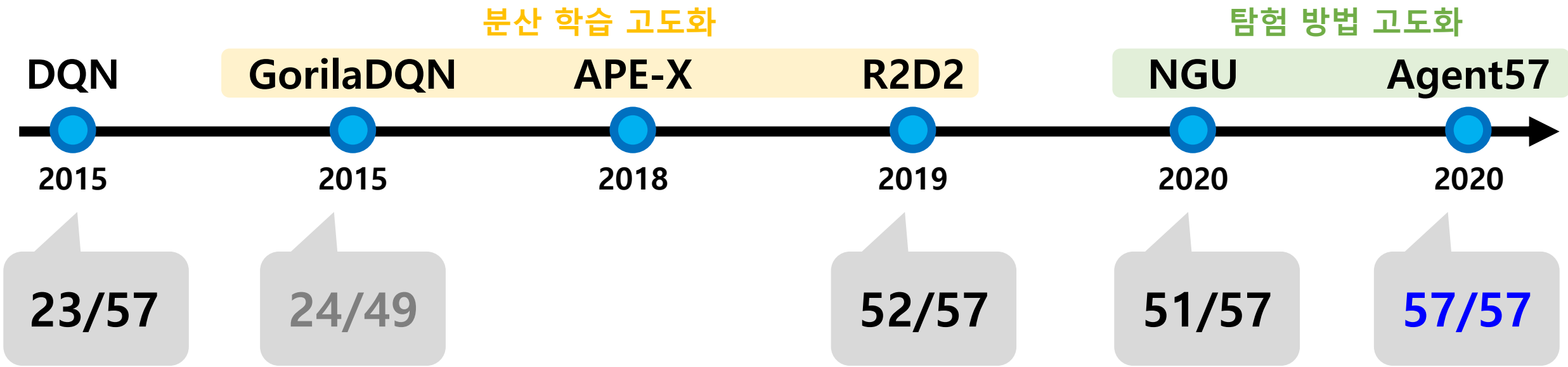
Montezuma's Revenge

Arcade Learning Environment

Atari2600 Reinforcement Learning Benchmark

❖ 시간 순서 별 Atari 벤치마크 최고 도달 성능

- Deep Q-network (DQN)를 시작으로 크게 분산 학습과 탐험 방법론을 보강하여 Agent57 모델이 모든 인간 점수를 뛰어넘는 성과를 달성함
- 분산 학습: 모델을 보다 빠르게 학습하기 위해서 병렬 구조로 학습하는 방법론
- 탐험 방법론: 보상이 희소하게 주어지는 환경을 잘 수행하기 위한 방법론



Agent57 Lineage

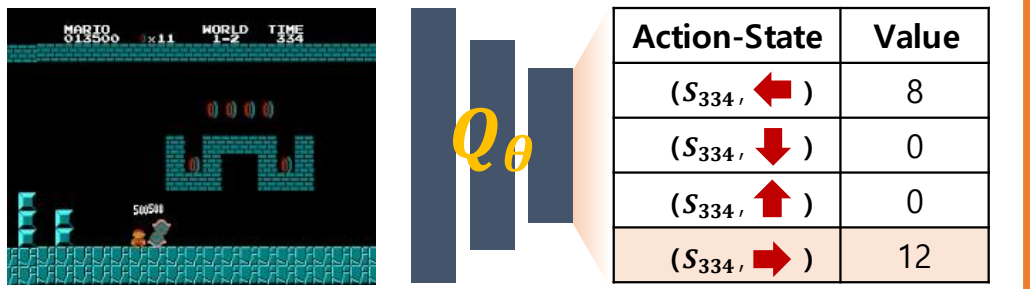
Deep Q-network (DQN)

❖ Human-level Control Through Deep Reinforcement Learning (Nature 2015, 30489회 인용)

- 가치기반 강화학습에 딥러닝을 적용한 연구 (deep Q-network; DQN)
- 행동-상태 가치(q)에 기반하여 행동을 선택하는 방법론으로 실제 가치를 추정할 가치함수(Q_θ)를 학습
- 별도의 명시된 정책함수가 없으며 탐험에 필요한 소수의 확률을 제외하고 가장 높은 가치를 갖는 행동을 선택함 (가치함수에 의존)
- 가치함수의 업데이트는 정답 가치와 추정 가치의 차이를 줄이는 방향으로 이루어 짐(gradient descent : $\theta' \leftarrow \theta - \alpha \nabla_\theta \big(L(\theta)\big)$)

정답 가치 / 추정 가치

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q_\theta(s', a') - Q_\theta(s, a)\right)^2\right]$$



State
(timestep =334)

Q_θ

Action-State	Value
$(S_{334}, \leftarrow)$	8
$(S_{334}, \downarrow)$	0
$(S_{334}, \uparrow)$	0
$(S_{334}, \rightarrow)$	12

State
(timestep =335)

Q_θ

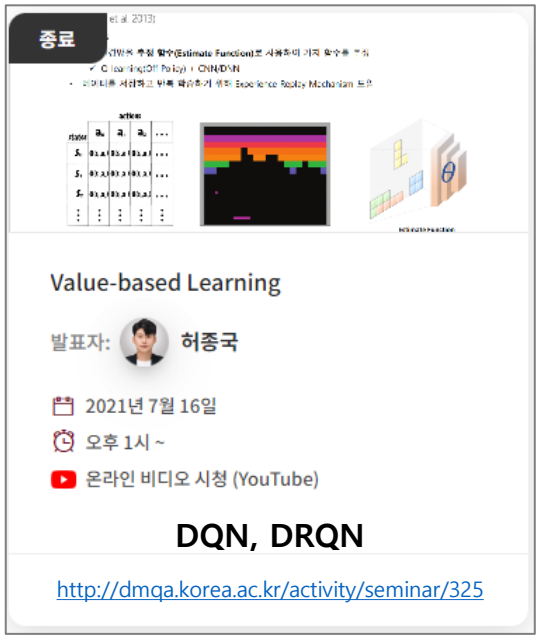
Action-State	Value
$(S_{335}, \leftarrow)$	3
$(S_{335}, \downarrow)$	0
$(S_{335}, \uparrow)$	0
$(S_{335}, \rightarrow)$	15

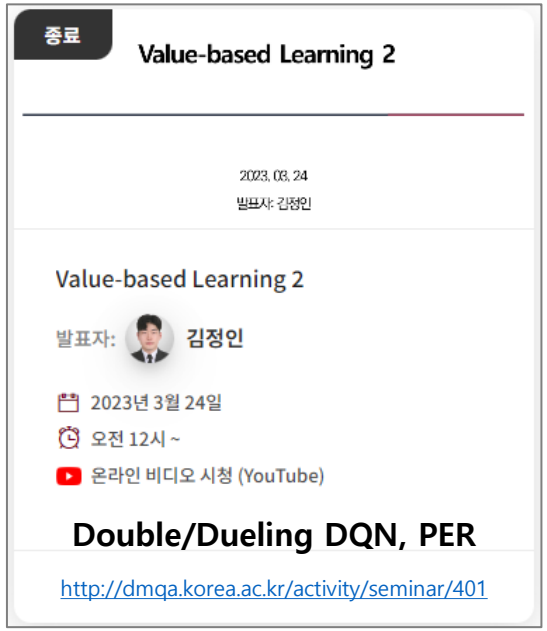
Agent57 Lineage

Deep Q-network (DQN)

❖ DQN를 고도화하는 다양한 연구가 진행됨

- DRQN: 환경의 일부분만 관찰할 수 있는 상황에서 DQN에 RNN을 추가하여 사용
- Double DQN: DQN에서 상태-행동 가치를 과대추정이 발생하는 것을 완화
- Dueling DQN: DQN에서 주어진 상태의 상대적인 가치를 추정하여 더 올바른 정책을 학습
- Prioritized Experience Replay (PER): 모델 학습에 도움이 되는 경험을 더 자주 학습하도록 우선 순위를 부여





Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ **Massively Parallel Methods for Deep Reinforcement Learning (ICML 2015, 620회 인용)**

- 기존의 강화학습은 단일 연산 자원(ex. CPU x 1ea)에서만 진행이 되어 학습 시간이 오래 걸림
- 따라서 다수의 연산 자원을 활용한 병렬 처리 방식으로 학습 속도를 향상시키고자 함

Training on
a single machine

Training on
multiple machines

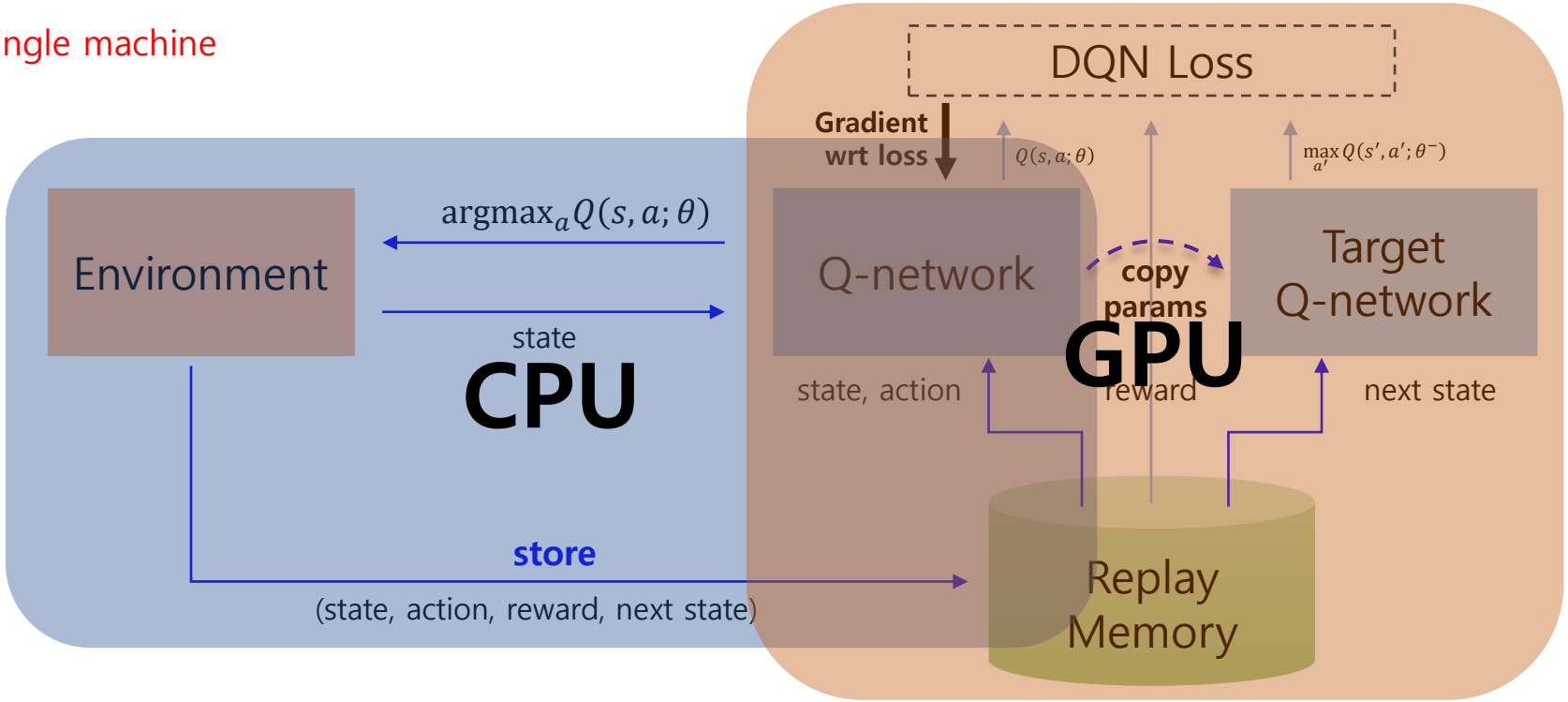
Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ Massively Parallel Methods for Deep Reinforcement Learning (ICML 2015, 620회 인용)

- 기존의 강화학습은 단일 연산 자원(ex. CPU x 1ea)에서만 진행이 되어 학습 시간이 오래 걸림
- 따라서 다수의 연산 자원을 활용한 병렬 처리 방식으로 학습 속도를 향상시키고자 함

** Running DQN on a single machine



Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., ... & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

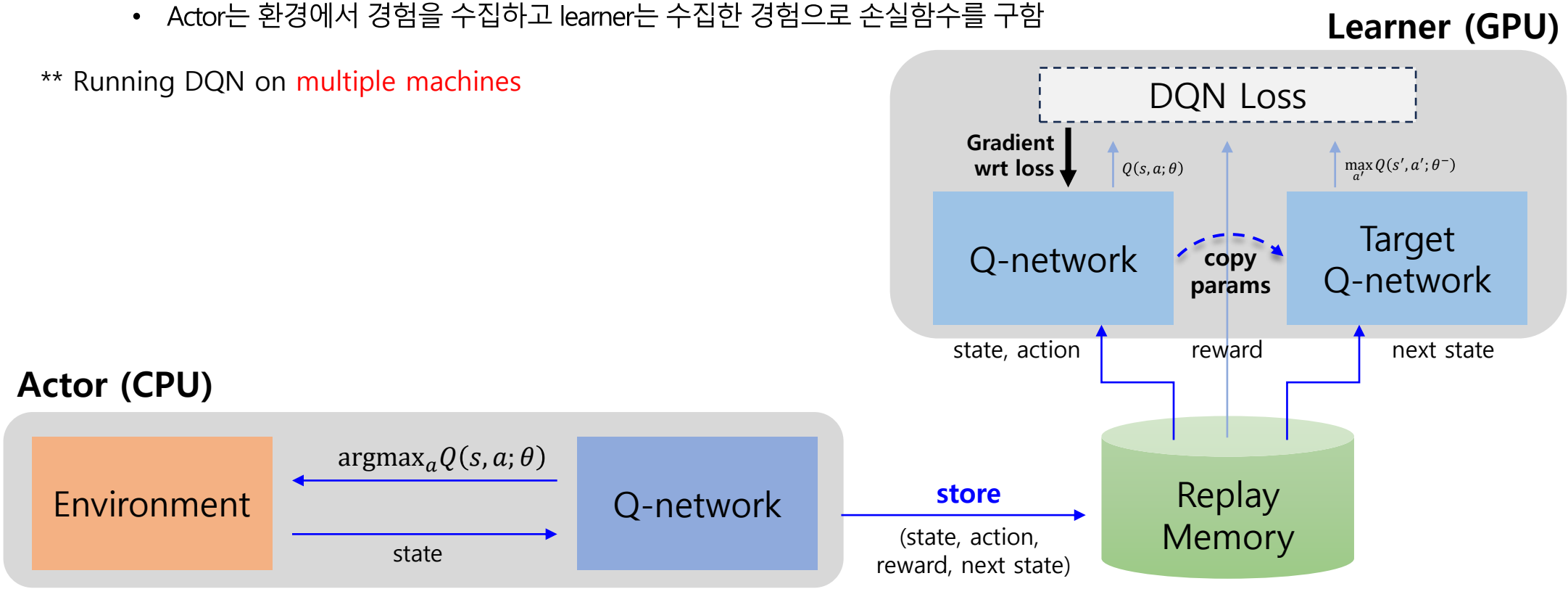
Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ Massively Parallel Methods for Deep Reinforcement Learning (ICML 2015, 620회 인용)

- 역할을 actor와 learner로 분리하여 학습을 수행 (이때 actor는 CPU, learner는 GPU로 구동)
- Actor는 환경에서 경험을 수집하고 learner는 수집한 경험으로 손실함수를 구함

** Running DQN on multiple machines



Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., ... & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

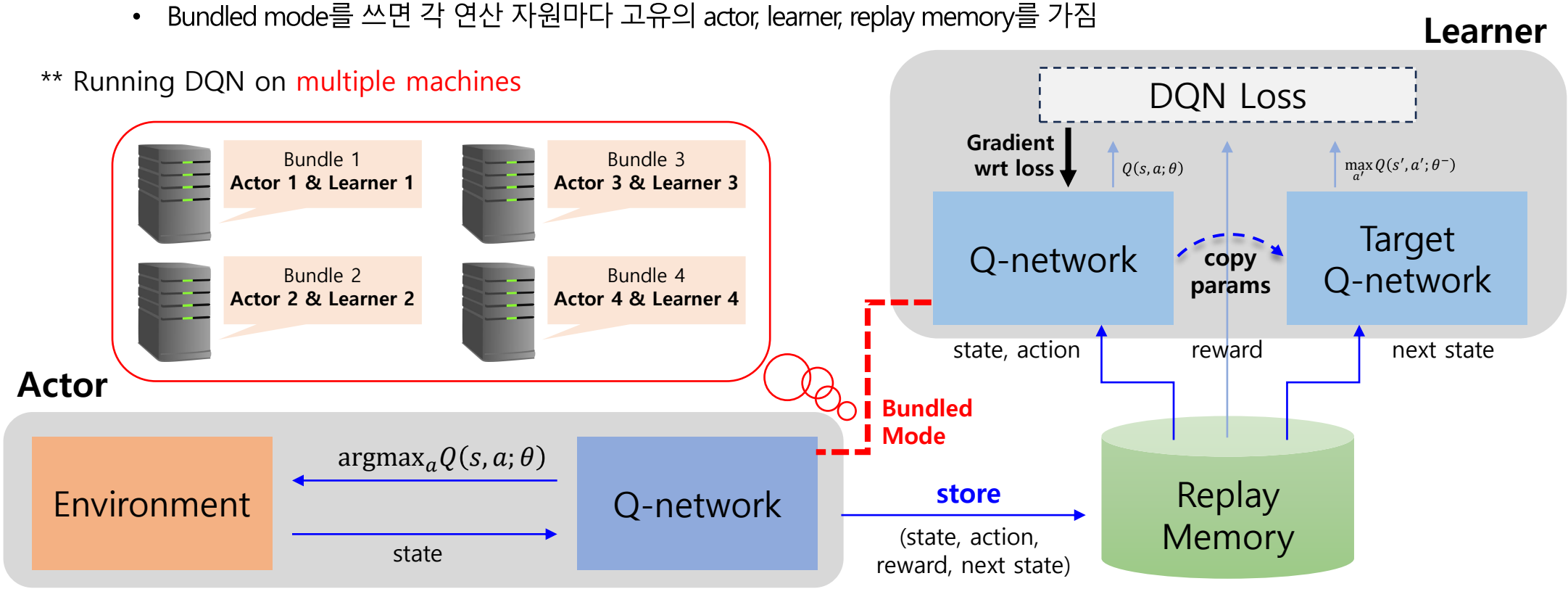
Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ Massively Parallel Methods for Deep Reinforcement Learning (ICML 2015, 620회 인용)

- Actor와 learner의 Q-network는 동일한 파라미터를 가짐
- Bundled mode를 쓰면 각 연산 자원마다 고유의 actor, learner, replay memory를 가짐

** Running DQN on multiple machines



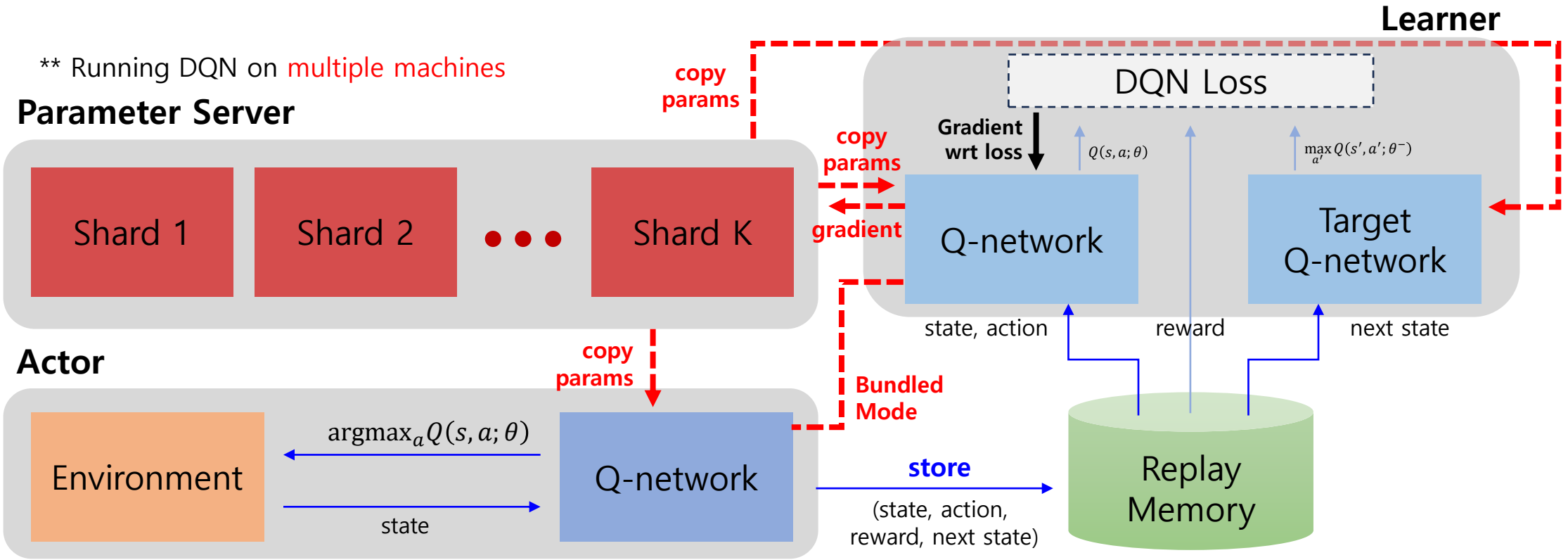
Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., ... & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ Massively Parallel Methods for Deep Reinforcement Learning (ICML 2015, 620회 인용)

- Parameter server는 각 bundle로부터 연산된 기울기를 수집하고 업데이트한 Q-network 파라미터를 각 bundle에 배포함



Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., ... & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

Agent57 Lineage

General Reinforcement Learning Architecture (Gorila)

❖ Experiment (performance comparison)

- Single GPU DQN은 최대 14일 동안 게임 49개를 학습
- GorilaDQN은 4일만에 Single DQN보다 41개 게임에서 더 좋은 성능을 달성
- GorilaDQN으로 사람보다 좋은 성능을 49개 게임 중 24개에서 달성

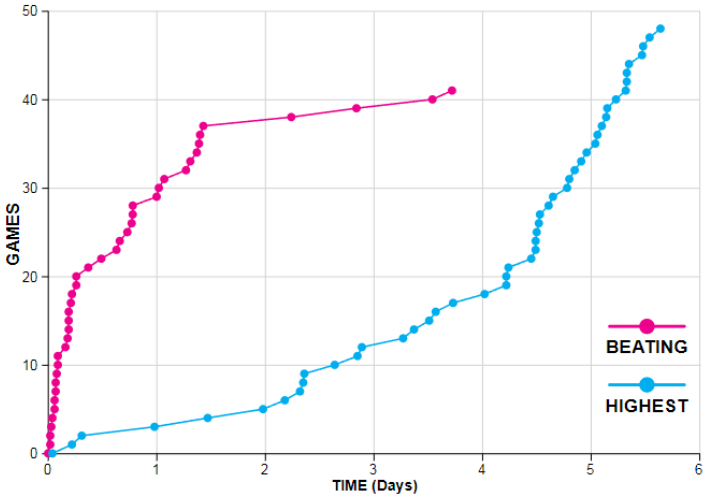


Figure 5. The time required by Gorila DQN to surpass single DQN performance (red curve) and to reach its peak performance (blue curve).

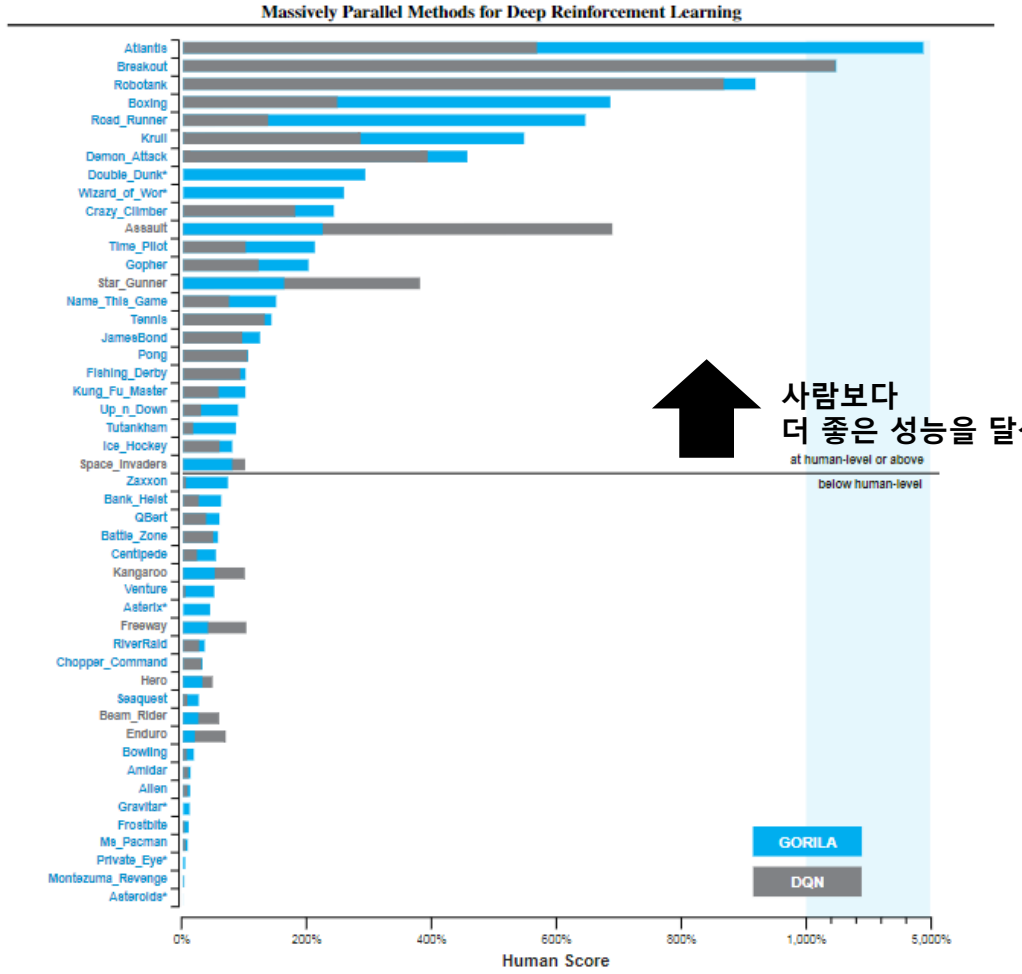


Figure 3. Performance of the Gorila agent on 49 Atari games with human starts evaluation compared with DQN (Mnih et al., 2015) performance with scores normalized to expert human performance. Font color indicates which method has the higher score. *Not showing DQN scores for Asterix, Asteroids, Double Dunk, Private Eye, Wizard Of Wor and Gravitar because the DQN human starts scores are less than the random agent baselines. Also not showing Video Pinball because the human expert scores are less than the random agent scores.

Nair, A., Srinivasan, P., Blackwell, S., Alcicek, C., Fearon, R., De Maria, A., ... & Silver, D. (2015). Massively parallel methods for deep reinforcement learning. arXiv preprint arXiv:1507.04296.

Agent57 Lineage

Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- 기존 연구에서는 분산 학습을 통해 기울기 연산을 병렬로 진행하는 것에 주로 초점이 맞춰져 있었음
- 이번 연구에서는 분산 학습을 통해 경험을 수집하고 저장하는 부분을 개선하는데 초점을 맞춤

** 기존의 분산 학습 방식과의 비교

	Gorila (Nair. et al., 2015)	Ape-X (Horgan. et al., 2018)
Architecture for distributed learning	Yes (same number of actors & learners)	No (single learner & multiple actors)
Assign different ϵ to each actors to collect diverse experience	No	Yes
Calculating gradients	Parallel	Non-parallel
Experience replay strategy	Local buffer & uniform sampling	Global buffer & prioritized sampling

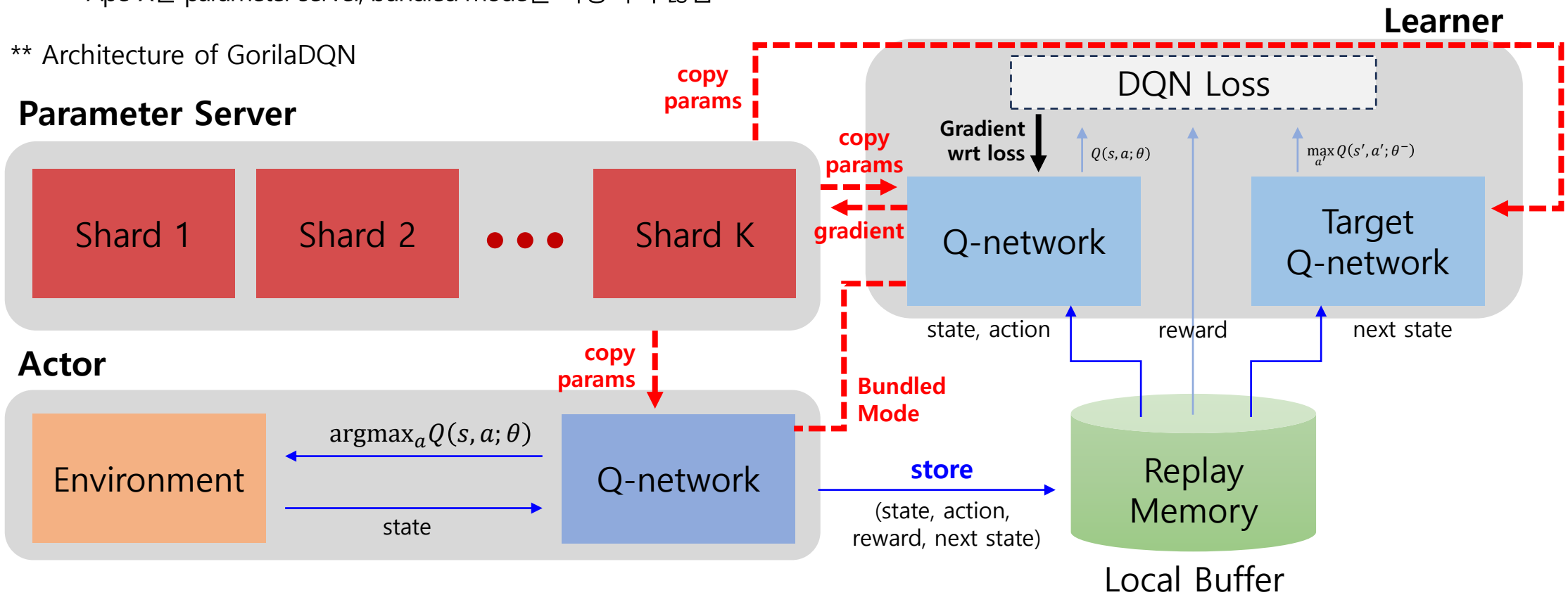
Agent57 Lineage

Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- 기존의 Gorila는 여러 묶음의 actor와 learner를 사용하였으며 메인 모델 파라미터를 별도의 서버에서 업데이트하는 방식을 사용
- Ape-X는 parameter server, bundled mode를 사용하지 않음

** Architecture of GorilaDQN



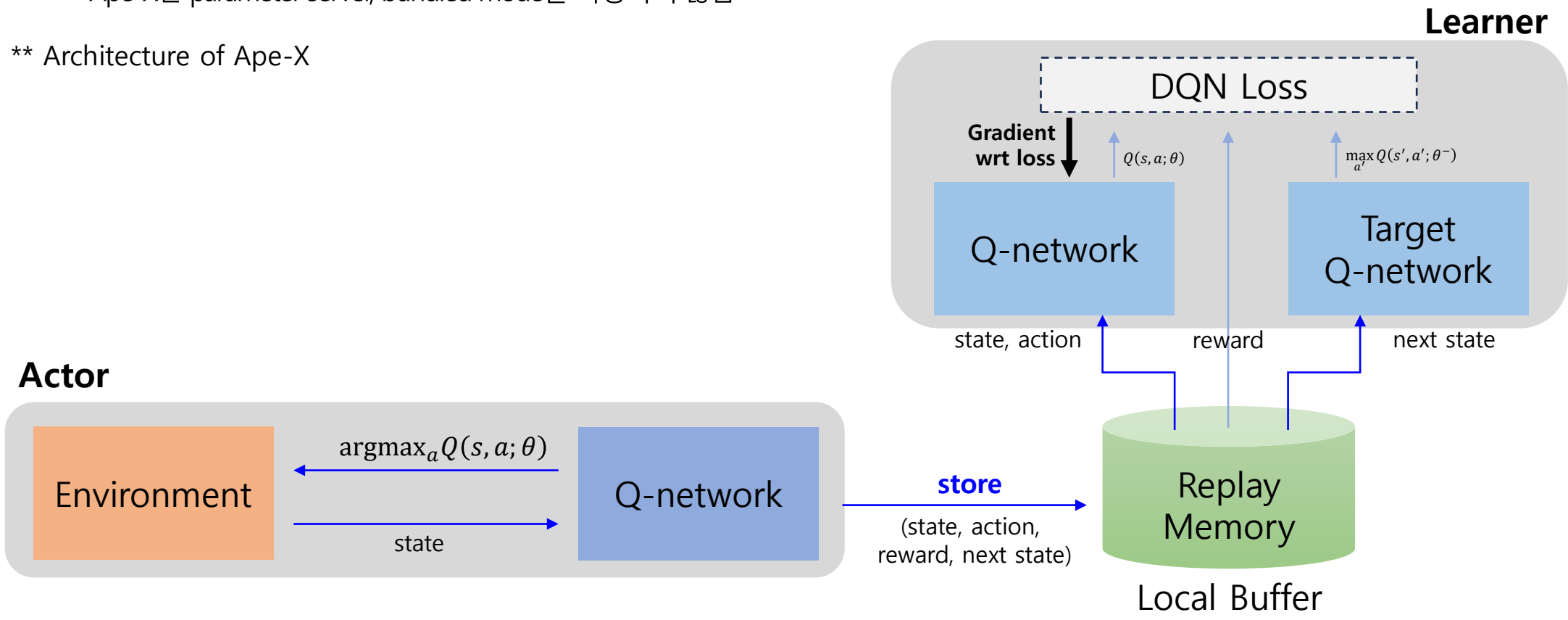
Agent57 Lineage

Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- 기존의 Gorila는 여러 묶음의 actor와 learner를 사용하였으며 메인 모델 파라미터를 별도의 서버에서 업데이트하는 방식을 사용
- Ape-X는 parameter server, bundled mode를 사용하지 않음

** Architecture of Ape-X



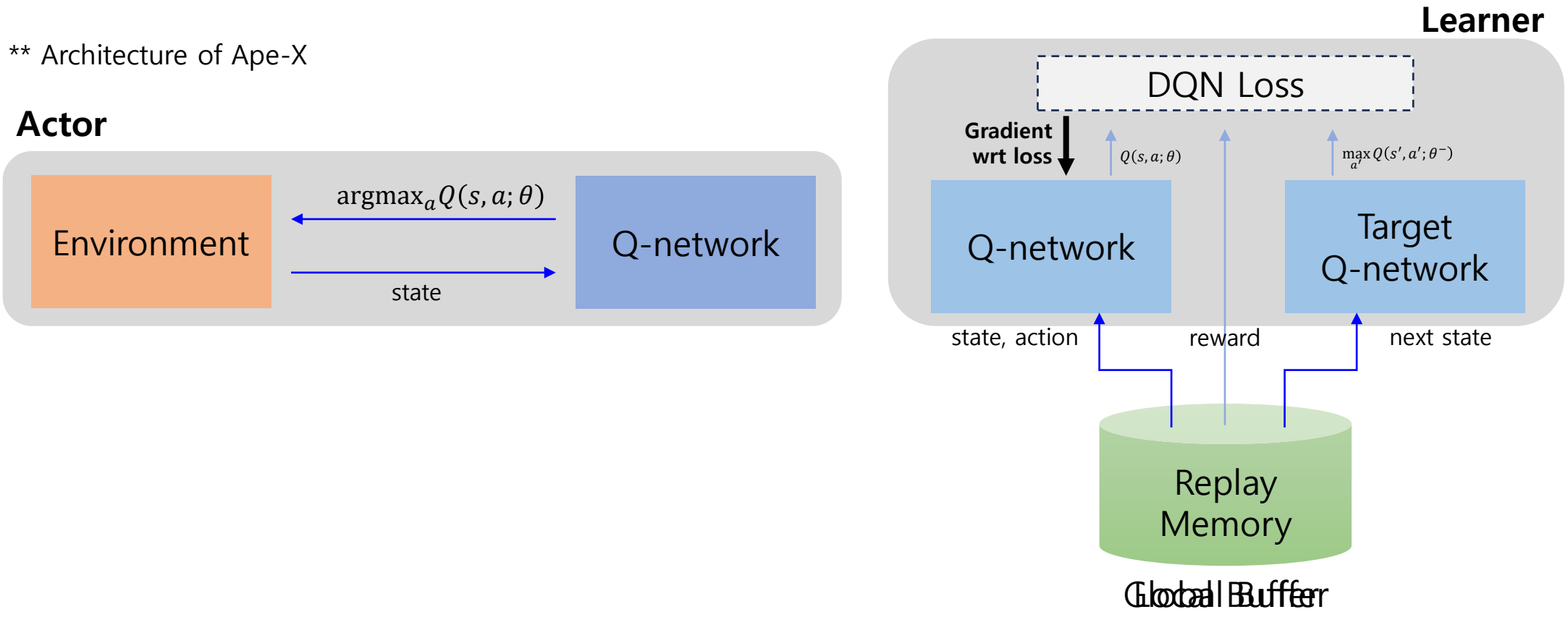
Agent57 Lineage

Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- Local / global buffer를 모두 사용하며 actor가 수집한 정보는 우선 local buffer에 저장

** Architecture of Ape-X



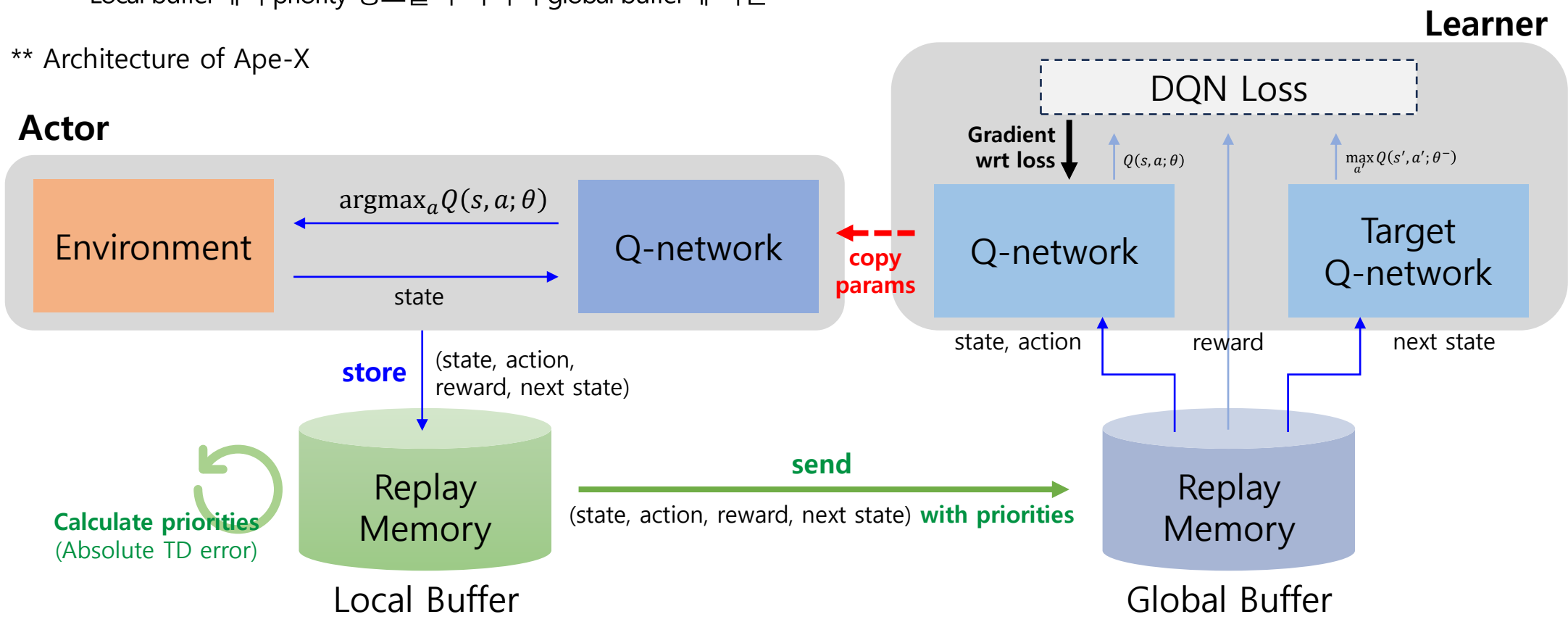
Agent57 Lineage

Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- Local / global buffer를 모두 사용하며 actor가 수집한 정보는 우선 local buffer에 저장
- Local buffer에서 priority 정보를 추가하여 global buffer에 이전

** Architecture of Ape-X



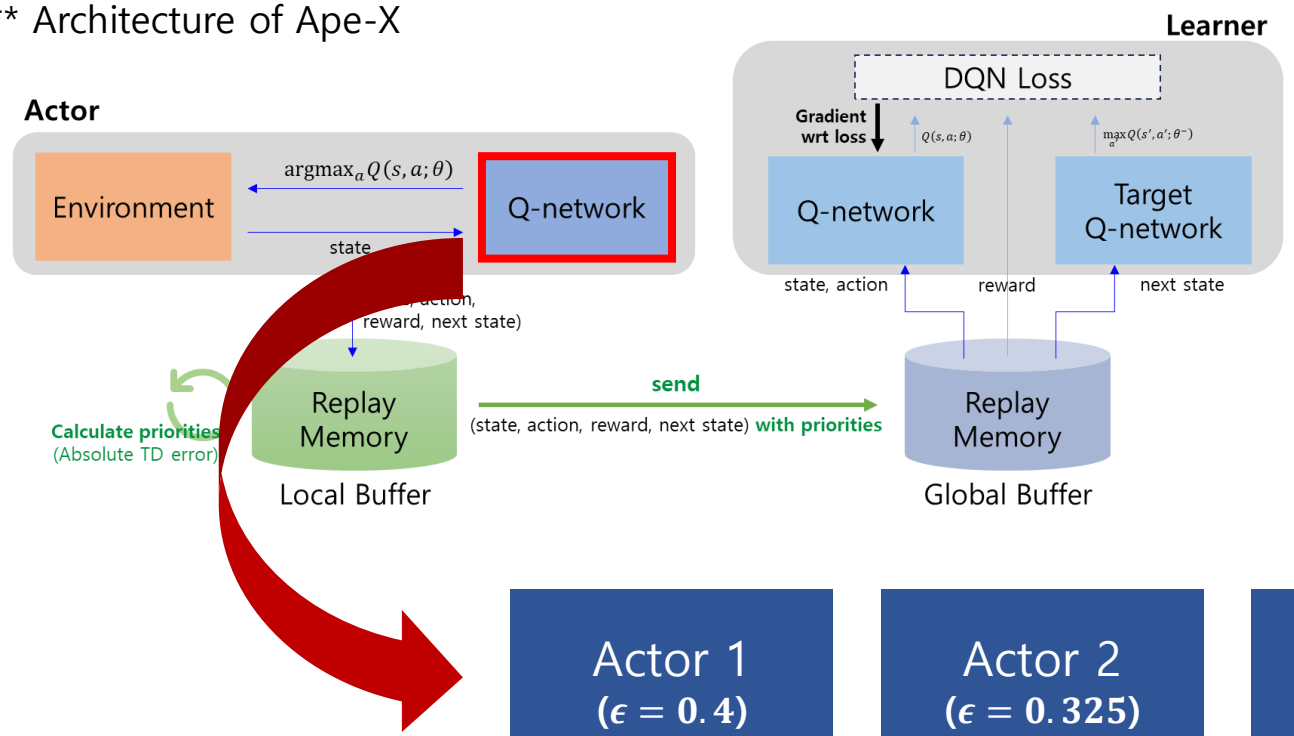
Agent57 Lineage

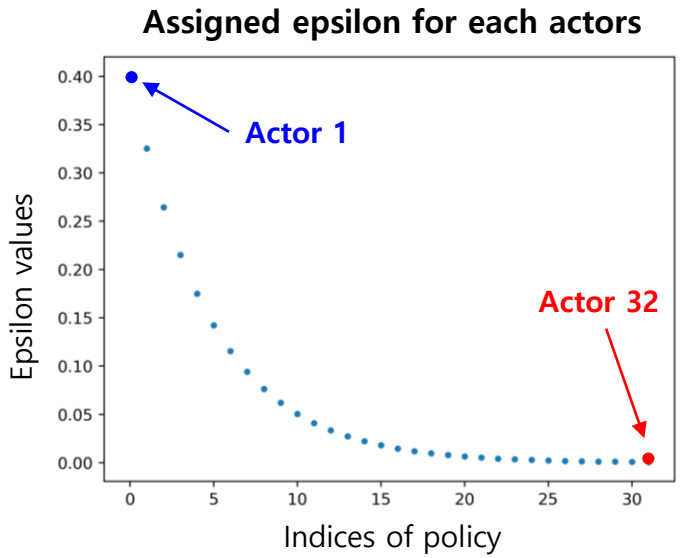
Ape-X

❖ Distributed Prioritized Experience Replay (ICLR 2018, 861회 인용)

- Actor마다 탐험 정도(ϵ)를 다양하게 부여하여 더 다양한 경험을 수집하고자 함

** Architecture of Ape-X





Actor 1	Actor 2	Actor 3	...	Actor 32
$(\epsilon = 0.4)$	$(\epsilon = 0.325)$	$(\epsilon = 0.264)$		$(\epsilon = 0.0007)$

Agent57 Lineage

Ape-X

❖ Experiment (performance comparison)

- 학습 속도와 성능 모두 뛰어나게 향상된 것을 확인
- 한정된 시간 내에서 actor 수가 늘어남에 따라 더 높은 점수를 빠르게 달성

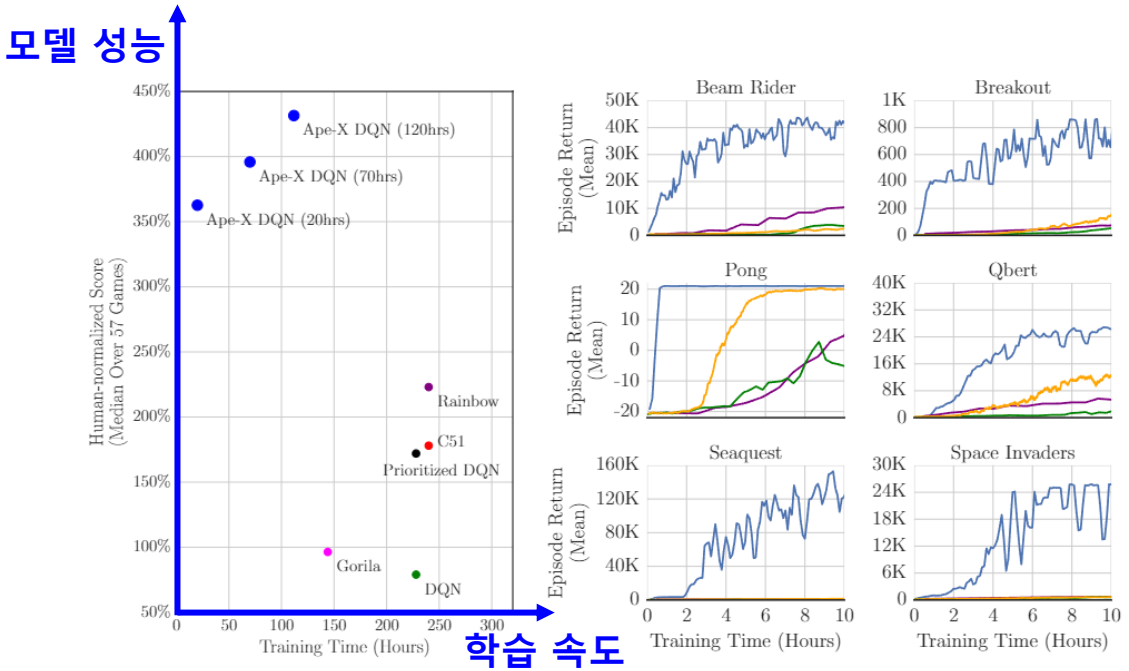


Figure 2: Left: Atari results aggregated across 57 games, evaluated from random no-op starts. Right: Atari training curves for selected games, against baselines. Blue: Ape-X DQN with 360 actors; Orange: A3C; Purple: Rainbow; Green: DQN. See appendix for longer runs over all games.

다른 방법론과의 성능 비교

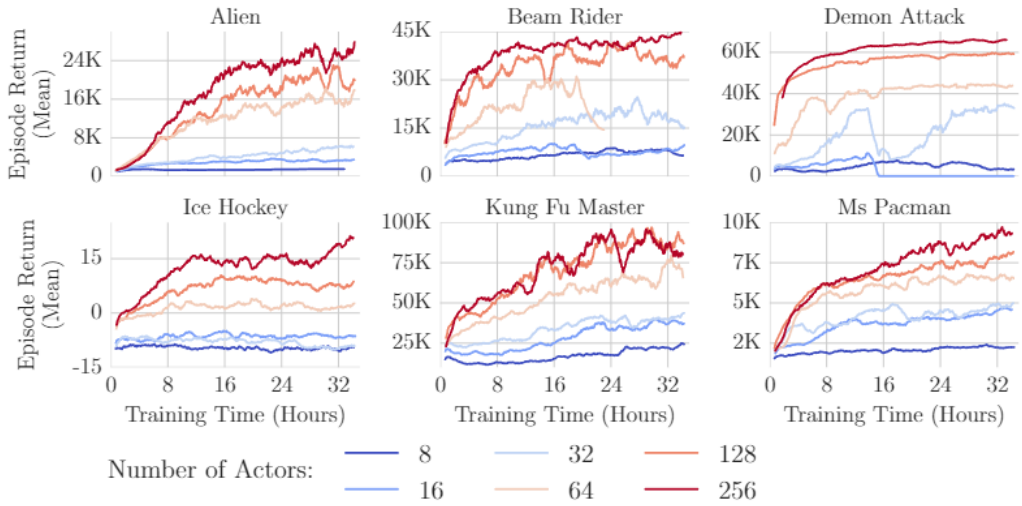


Figure 4: Scaling the number of actors. Performance consistently improves as we scale the number of actors from 8 to 256, note that the number of learning updates performed does not depend on the number of actors.

Actor 수에 따른 성능 비교

Agent57 Lineage

Ape-X

❖ Experiment (performance comparison detail)

- Ape-X에서 actor는 360 core를 사용 (1 core = 1 actor), learner는 1 GPU + 16 core를 사용
- 다른 단일 학습 그리고 분산 학습 방법론에 비해서 더 시간 효율적으로 높은 성능을 달성하는 것을 확인

Algorithm	Training Time	Environment Frames	Resources (per game)	Median (no-op starts)	Median (human starts)
Ape-X DQN	5 days	22800M	376 cores, 1 GPU [a]	434%	358%
Rainbow	10 days	200M	1 GPU	223%	153%
Distributional (C51)	10 days	200M	1 GPU	178%	125%
A3C	4 days	—	16 cores	—	117%
Prioritized Dueling	9.5 days	200M	1 GPU	172%	115%
DQN	9.5 days	200M	1 GPU	79%	68%
Gorila DQN [c]	∼4 days	—	unknown [b]	96%	78%
UNREAL [d]	—	250M	16 cores	331% [d]	250% [d]

Table 1: Median normalized scores across 57 Atari games. [a] Tesla P100. [b] >100 CPUs, with a mixed number of cores per CPU machine. [c] Only evaluated on 49 games. [d] Hyper-parameters were tuned per game.

Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Recurrent Experience Replay In Distributed Reinforcement Learning (ICLR 2019, 533회 인용)

- 더 온전한 상태 표현을 위해서 long-short term memory (LSTM)를 통해 과거의 데이터까지 활용하고자 함
  - ✓ 해당 논문은 Atari 벤치마크 환경을 "almost fully observable"로 표현함 (partially observable MDP로 인식)
  - ✓ 4-stacked frame을 쓰는 이유도 하나의 프레임에서는 객체의 속도나 움직임을 완벽하게 파악할 수 없기 때문
  - ✓ 여기서 제안하는 방법론(R2D2)는 Ape-X와 거의 같은 구조를 사용하며 Q-network에 LSTM을 추가한 것이 가장 다른 점

(Atari benchmark)	**Ape-X (Horgan. et al., 2018)**	**R2D2 (Kapturowski. et al., 2019)**
N-step return	Yes	Yes (n=5)
Double DQN	Yes	Yes
Dueling DQN	Yes	Yes
Prioritized Experience Replay	Yes	Yes
Priority calculation	Absolute TD-error	Mixture of max & mean of absolute TD-error
# of frame stack	4	4
Using recurrent neural networks	No	Yes (LSTM)

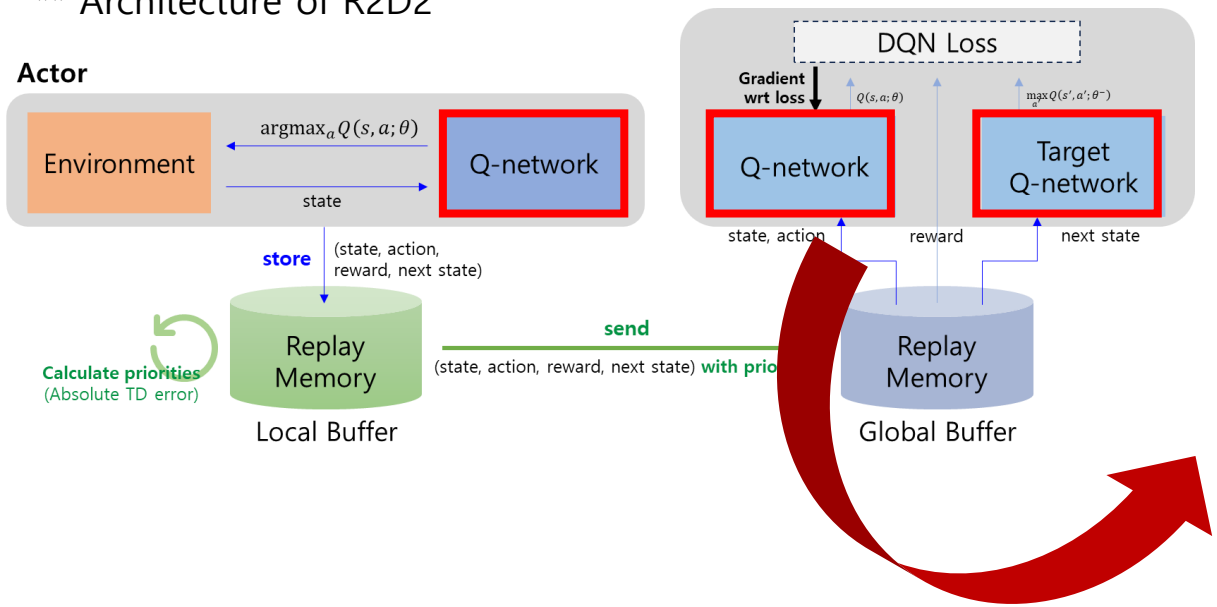
Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

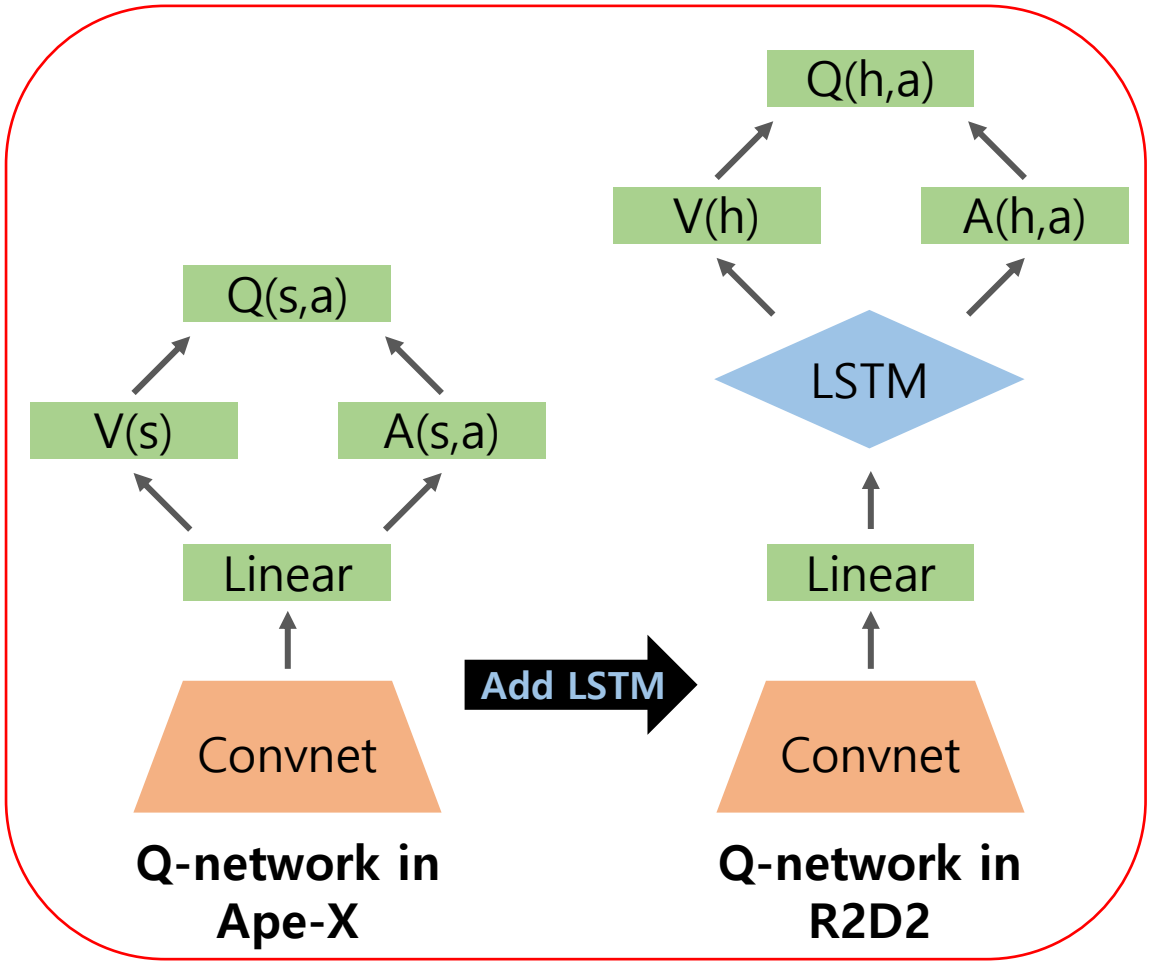
❖ 모델 구조 측면에서 Ape-X와 다른 점

- Q-network를 DRQN (Hausknecht & Stone, 2015)과 비슷한 구조로 변경

** Architecture of R2D2



Q-network에 LSTM을 추가



Hausknecht, M., & Stone, P. (2015, September). Deep recurrent q-learning for partially observable mdps. In 2015 aaai fall symposium series.

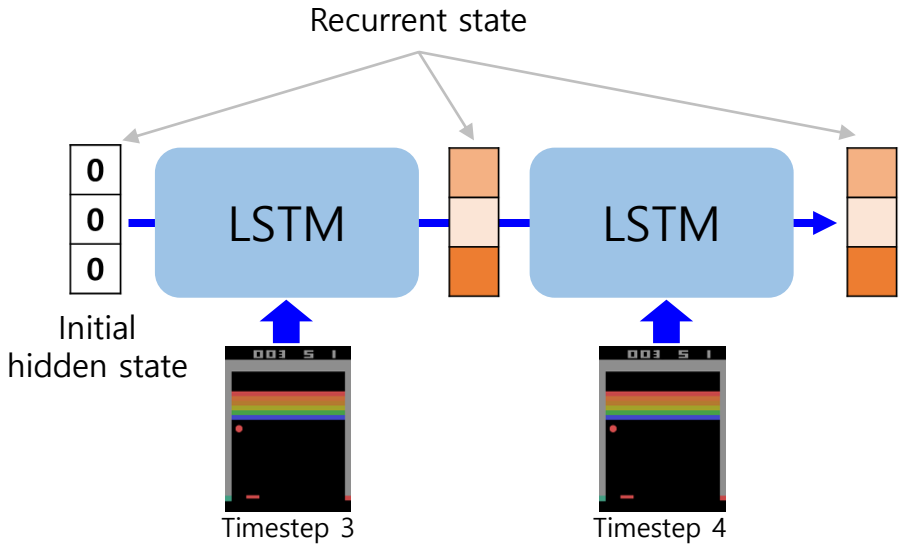
Gruslys, A., Dabney, W., Azar, M. G., Piot, B., Bellemare, M., & Munos, R. (2018, February). The Reactor: A fast and sample-efficient Actor-Critic agent for Reinforcement Learning. In International Conference on Learning Representations.

Agent57 Lineage

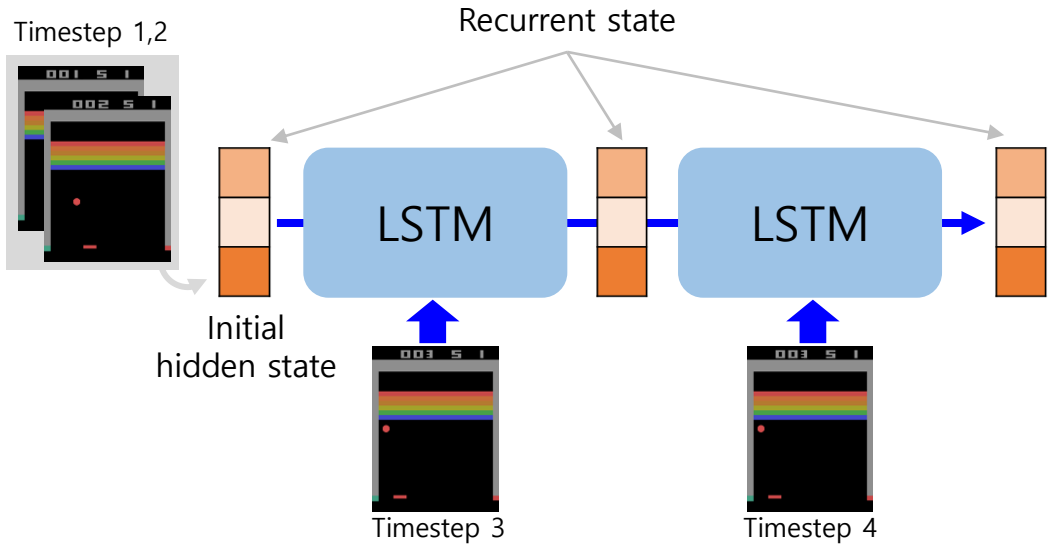
Recurrent Replay Distributed DQN (R2D2)

❖ R2D2가 기존에 쓰던 LSTM 활용 방식에서 개선하려는 부분: Stored state

- DRQN에서는 replay memory로 LSTM을 학습할 때 initial hidden state로서 zero vector를 사용
  - ➢ 하지만 실제로 시작할 때의 recurrent state와 다르기 때문에 시계열적 특성을 충분히 사용하기 어려움
  - ➢ 다만 충분히 긴 시퀀스가 입력되는 경우 recurrent state가 회복되는 양상을 보임
- R2D2는 따라서 LSTM을 사용할 때 initial hidden state에 실제 입력되었던 상태 정보가 들어있는 recurrent state를 사용
  - ➢ LSTM 통해 얻은 recurrent state를 buffer에 저장 (**stored state**)



LSTM in DQRN



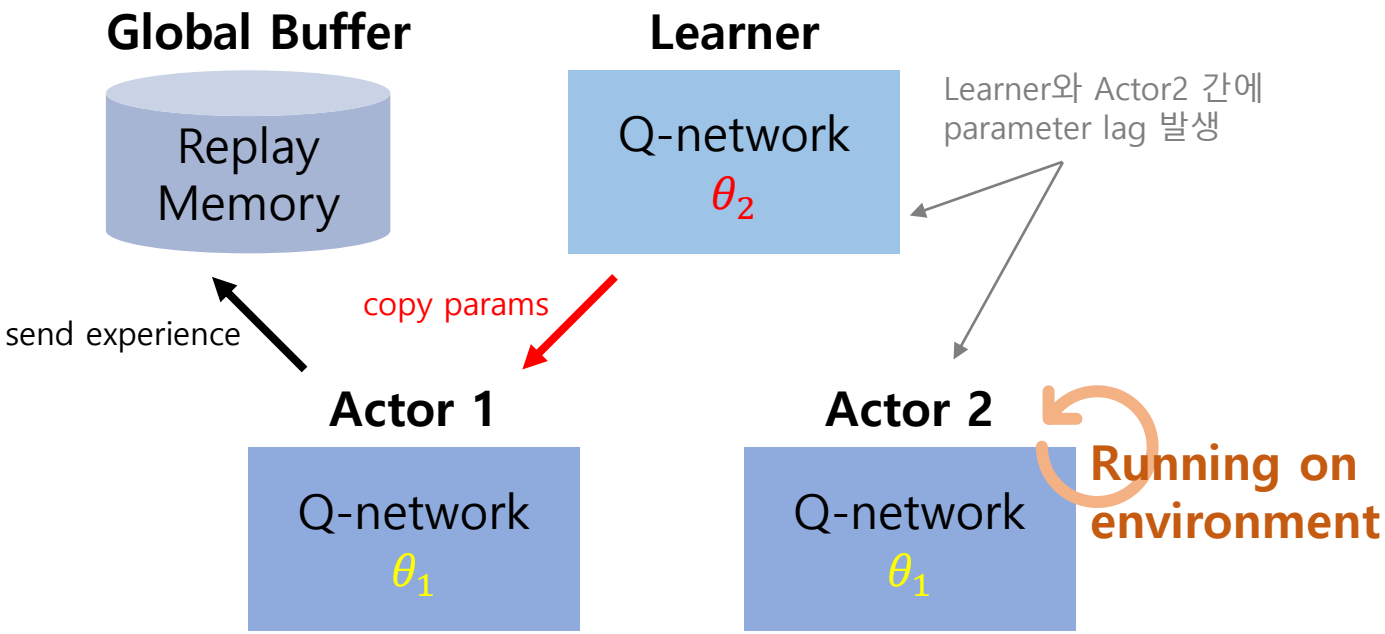
LSTM in R2D2

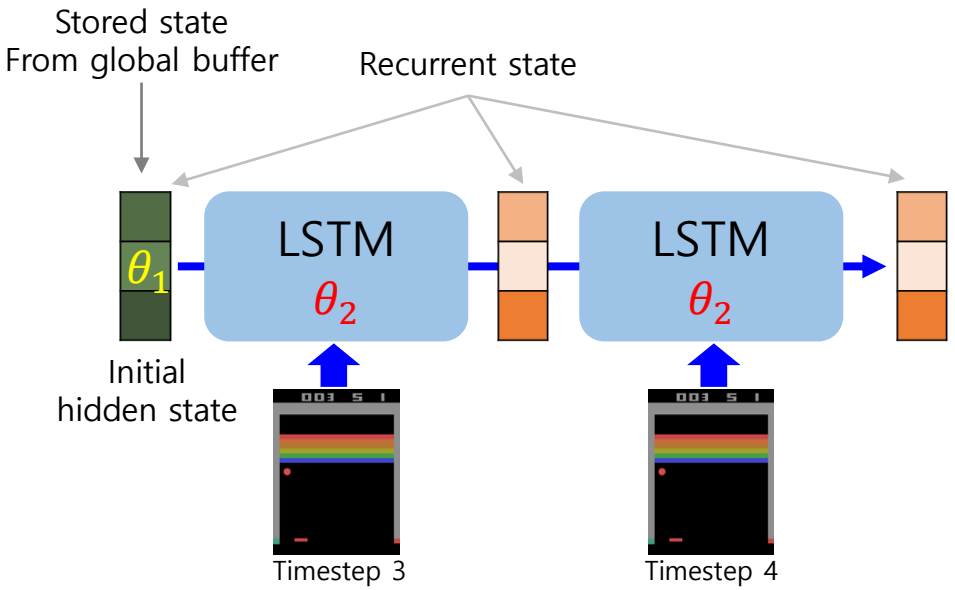
Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ LSTM을 사용하면서 발생하는 문제점

- Representation shift에 따라 stored state가 본래의 역할을 못 함(**recurrent state staleness**)
  - ➢ 분산 학습 시 leaner와 actor는 비동기적으로 파라미터가 업데이트 되기 때문에 파라미터 차이(parameter lag)가 발생
  - ➢ Parameter lag가 발생하면 당연히 같은 데이터에 대한 표현도 차이(representational drift)가 발생





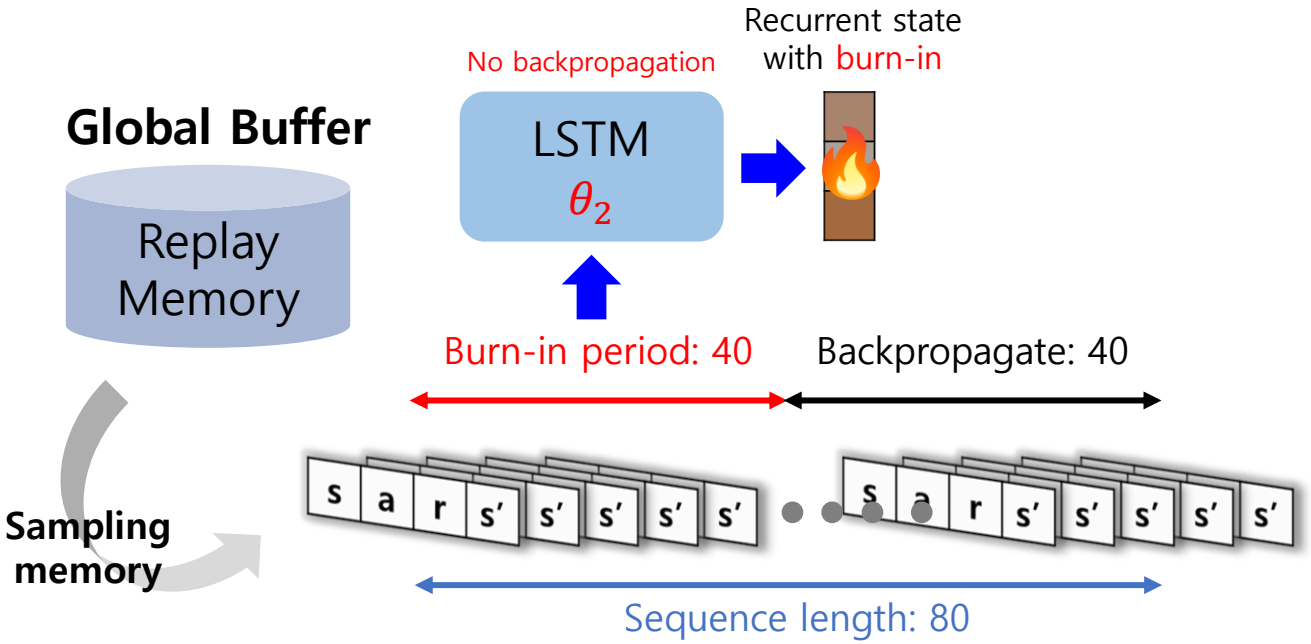
LSTM in R2D2

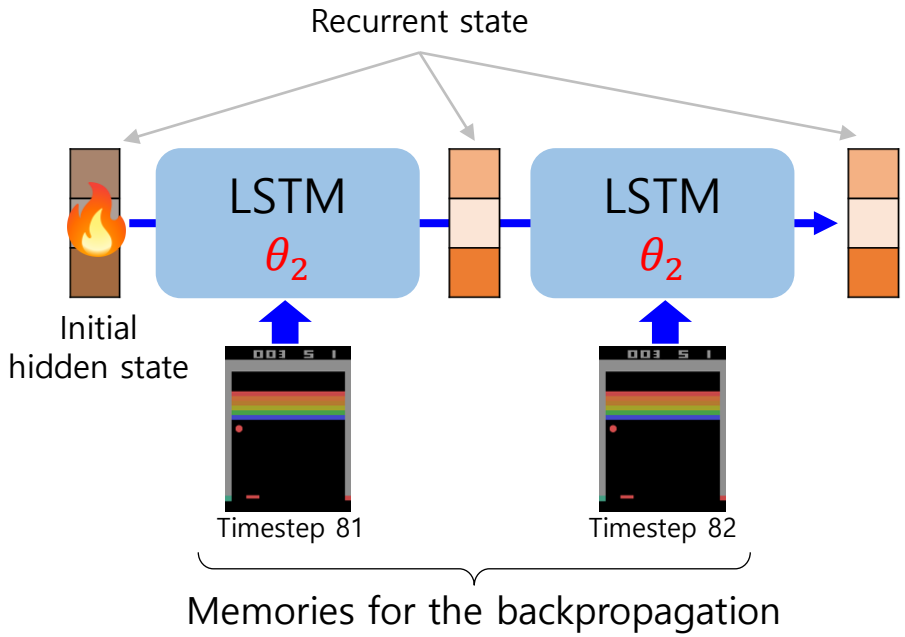
Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ **Recurrent state staleness 해결 방안: Burn-in period**

- 저장한 시퀀스의 앞부분을 initial hidden state를 만드는데 사용
- 충분히 긴 시퀀스가 입력될 때는 recurrent state가 복구되는 경향이 있고 사용할 네트워크의 파라미터로 진행하므로 representation drift를 회피해서 적합한 initial hidden state를 만들 수 있음





Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Experiment (performance comparison)

- Ape-X와의 주요한 차이는 사실상 LSTM의 사용 여부 하나였으나 매우 큰 차이를 보임
- R2D2가 human score를 이긴 게임은 57개 중 52개

	Atari-57		DMLab-30	
Human-Normalized Score	**Median**	**Mean**	**Median**	**Mean-Capped**
Ape-X (Horgan et al., 2018)	434.1%	1695.6%	–	–
Reactor (Gruslys et al., 2018)	187.0%	–	–	–
IMPALA, deep (Espeholt et al., 2018)	191.8%	957.6%	49.0%	45.8%
IMPALA, shallow (re-run)	–	–	89.7%	73.6%
IMPALA, deep (re-run)	–	–	**107.5%**	85.1%
R2D2+	–	–	99.5%	**85.7%**
R2D2, feed-forward	589.2%	1974.4%	–	–
R2D2	**1920.6%**	**4024.9%**	96.9%	78.3%

Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Experiment (performance comparison)

- Atari 벤치마크에서 매우 우수한 성능을 보여주었으나 탐험이 어려운 환경에서는 여전히 낮은 성능을 달성
  - ✓ Montezuma's revenge
  - ✓ Pitfall!

Hard exploration games

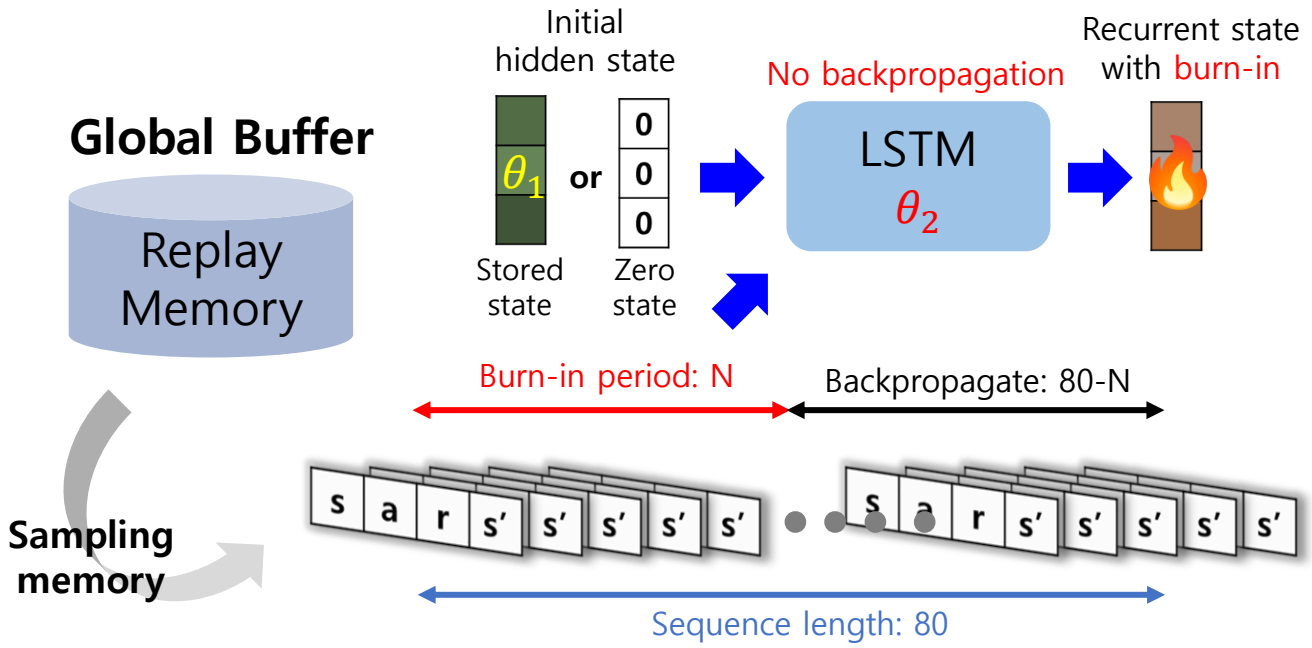
GAMES	HUMAN	REACTOR	IMPALA(S/D)	APE-X	R2D2
montezuma_revenge	**4753.3**	2643.5	0.0/0.0	2500.0	2061.3
ms_pacman	6951.6	2724.3	6501.7/7342.3	11255.2	**42281.7**
name_this_game	8049.0	9907.1	6049.6/21537.2	25783.3	**58182.7**
phoenix	7242.6	40092.3	33068.2/210996.5	224491.1	**864020.0**
pitfall	**6463.7**	-3.5	-11.1/-1.7	-0.6	0.0

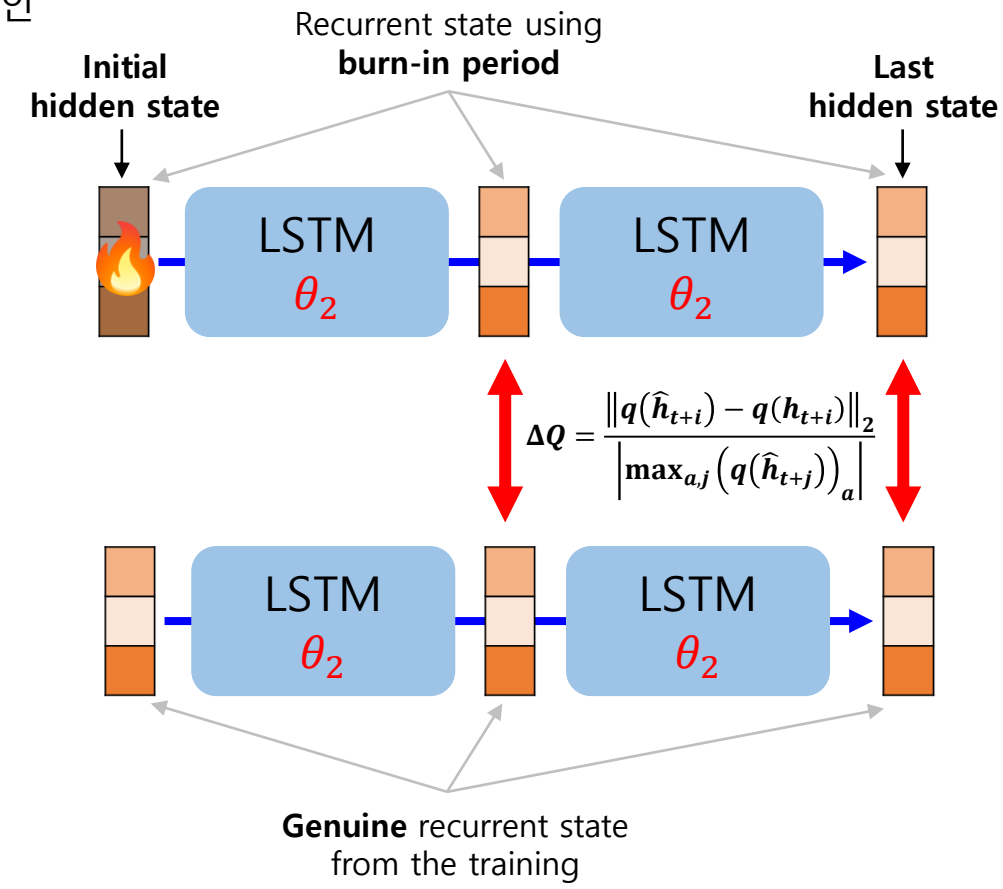
Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Experiment (Q-value discrepancy; ΔQ)

- Burn-in 방법론의 효과를 검증
  - ✓ Burn-in 방법론이 recurrent state staleness를 얼마나 완화하는지 확인
  - ✓ 실제 학습에서 가져온 recurrent state와 burn-in 과정으로 만든 recurrent state 간의 차이를 구함(ΔQ)
  - ✓ Burn-in 시 stored state 또는 zero state를 사용



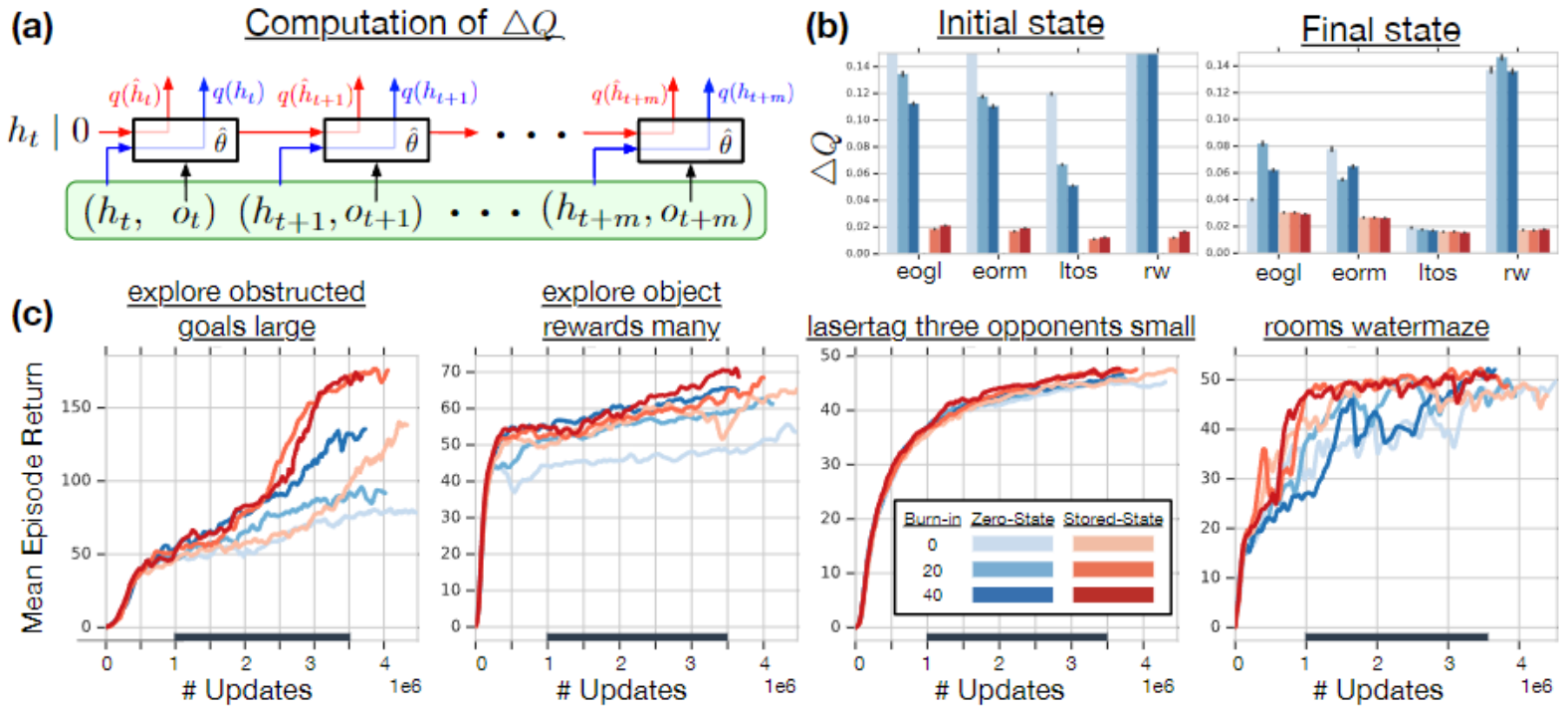


Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Experiment (Q-value discrepancy; ΔQ)

- Burn-in 방법론의 효과를 검증
  - ✓ Burn-in 길이 간의 차이는 크지 않으나 stored-state를 사용하지 않았을 때 $\boldsymbol{\Delta Q}$가 커지는 것을 확인
  - ✓ Q-value discrepancy가 작으므로 부정확한 초기 값으로 인해 학습이 망가지는(destructive updates) 것을 방지



Agent57 Lineage

Recurrent Replay Distributed DQN (R2D2)

❖ Experiment (measuring the effectiveness of using LSTM)

- LSTM의 효과를 검증
  - ✓ LSTM 대신에 feed-forward를 사용하여 학습 및 검증
  - ✓ Feed-forward를 썼을 때 성능이 급락하는 것을 확인

Figure 1

Human-Normalized Score	Atari-57 Median	Atari-57 Mean
Ape-X (Horgan et al., 2018)	434.1%	1695.6%
Reactor (Gruslys et al., 2018)	187.0%	–
IMPALA, deep (Espeholt et al., 2018)	191.8%	957.6%
IMPALA, shallow (re-run)	–	–
IMPALA, deep (re-run)	–	–
R2D2+	–	–
R2D2, feed-forward	589.2%	1974.4%
R2D2	**1920.6%**	**4024.9%**

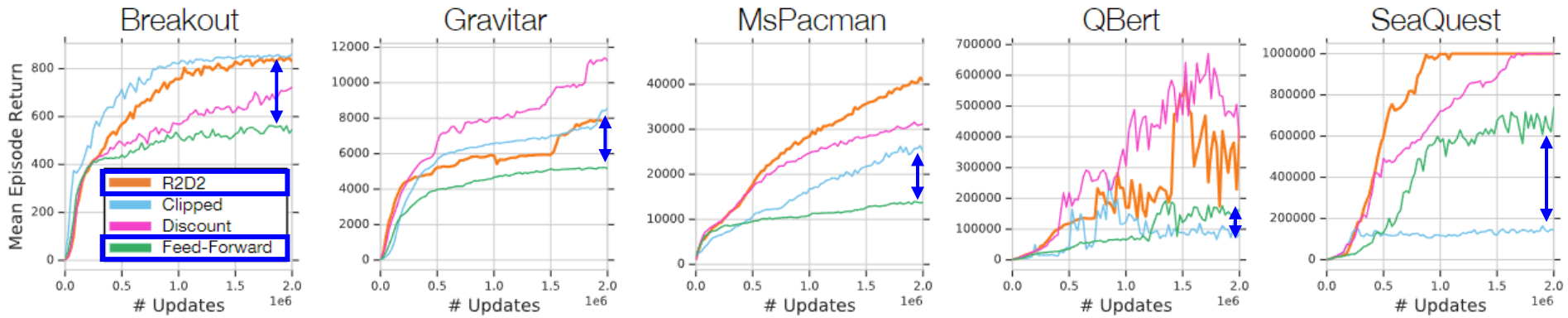


Figure 4: Ablations with reward clipping instead of value function rescaling (**Clipped**), smaller discount factor of $\gamma = 0.99$ (**Discount**), and feed-forward (**Feed-Forward**) variants of R2D2.

Agent57 Lineage

Never give up (NGU)

❖ Never give up: Learning directed exploration strategies (ICLR 2020, 324회 인용)

- NGU는 R2D2에서 부족했던 탐험 성능을 보완하였음
- Episodic novelty module를 통해서 에피소드 내에서의 새로운 상태에 대한 호기심을 측정
- Life-long novelty module을 통해서 에피소드 간에서의 호기심을 측정

(Atari benchmark)	R2D2 (Kapturowski. et al., 2019)	NGU (Badia. et al., 2020)
# of frame stack	4	1
Using recurrent neural networks	Yes (LSTM)	Yes (LSTM)
Using diverse pairs of reward weight (β) & discount factor (γ)	No	Yes
Using dynamics modeling	No	Yes
Using intrinsic reward	No	Yes

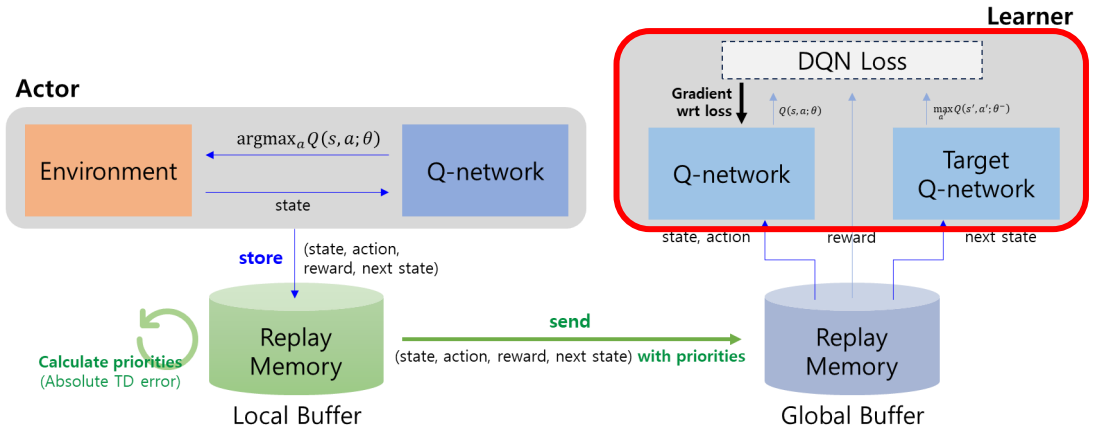
Agent57 Lineage

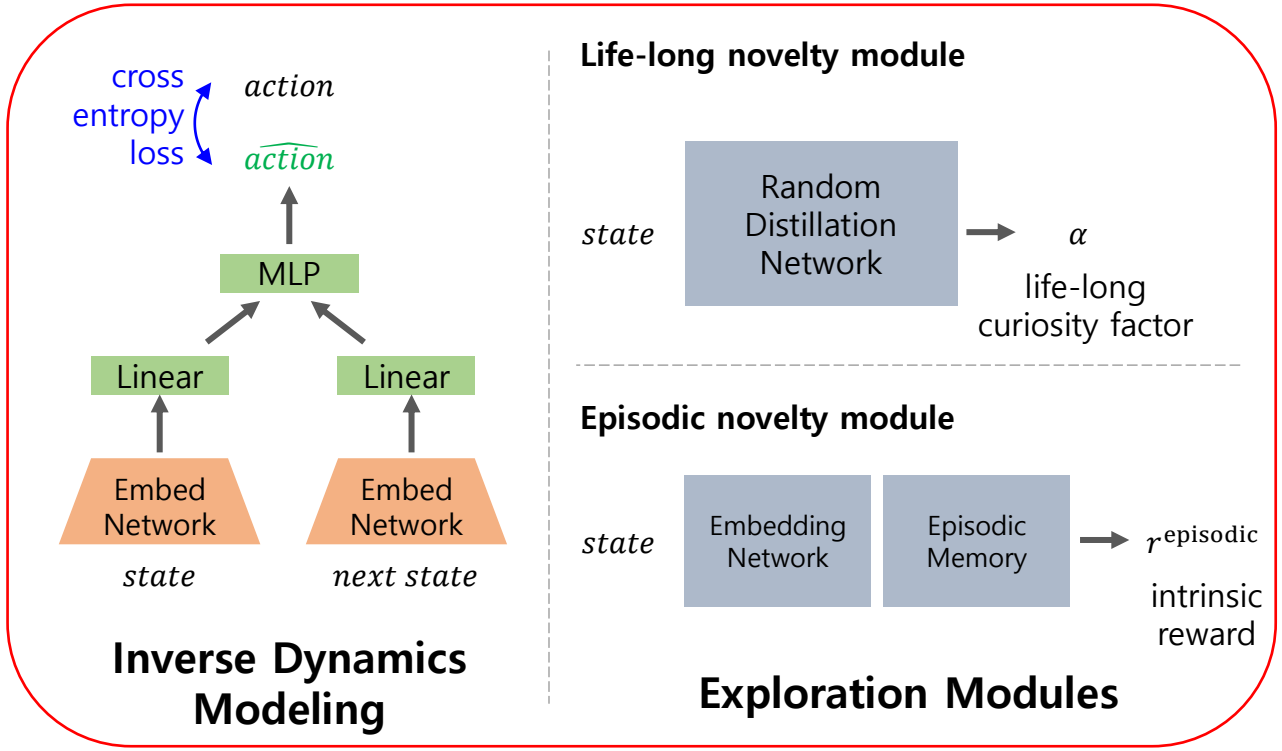
Never give up (NGU)

❖ **Never give up: Learning directed exploration strategies (ICLR 2020, 324회 인용)**

- R2D2는 Atari 벤치마크에서 뛰어난 성능을 보여주었지만 탐험이 어려운 환경(Pitfall! , Montezuma's revenge)에서 낮은 성능을 보임
- NGU에서는 R2D2에 탐험을 강화하는 모듈을 추가하여 탐험이 어려운 환경에서의 성능을 향상

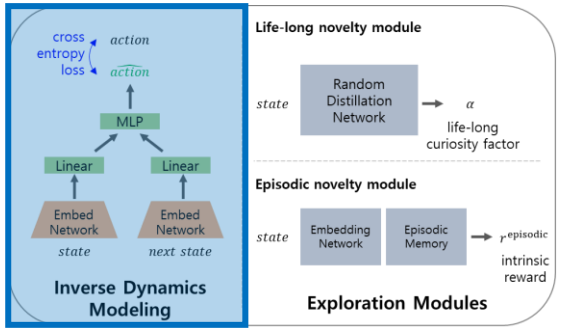
** Architecture of NGU (based on R2D2)





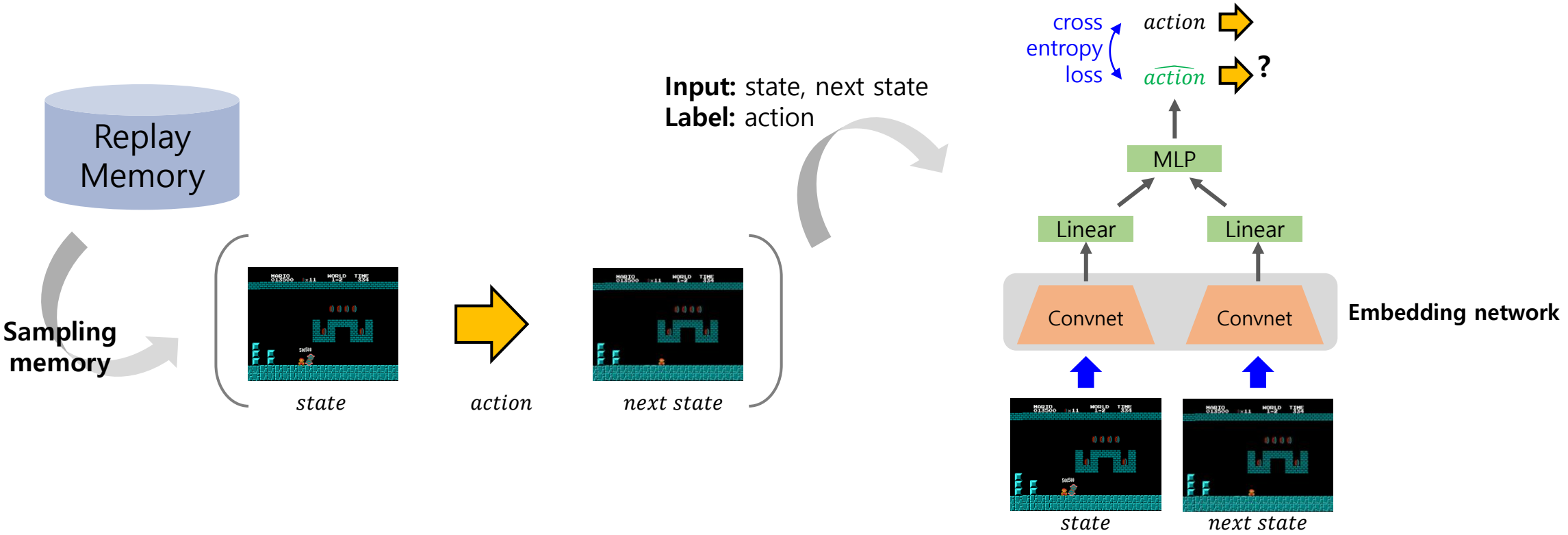
Agent57 Lineage

Never give up (NGU)



❖ Overview of Inverse Dynamics Modeling

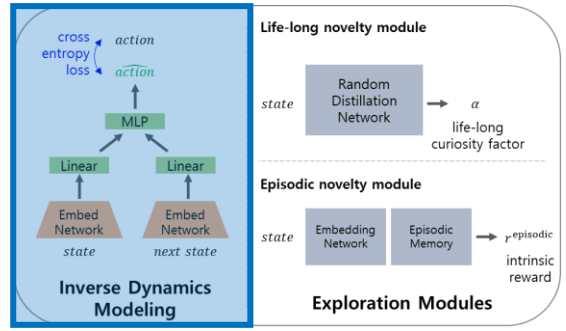
- Inverse dynamics modeling은 현재와 다음에 발생하는 상태 값으로 현재의 행동을 예측하는 문제



Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." *International conference on machine learning*. PMLR, 2017.

Agent57 Lineage

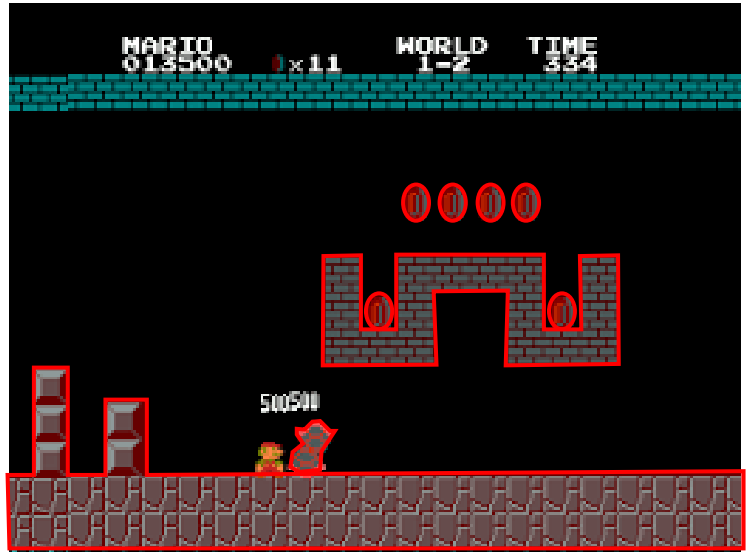
Never give up (NGU)



❖ Overview of Inverse Dynamics Modeling

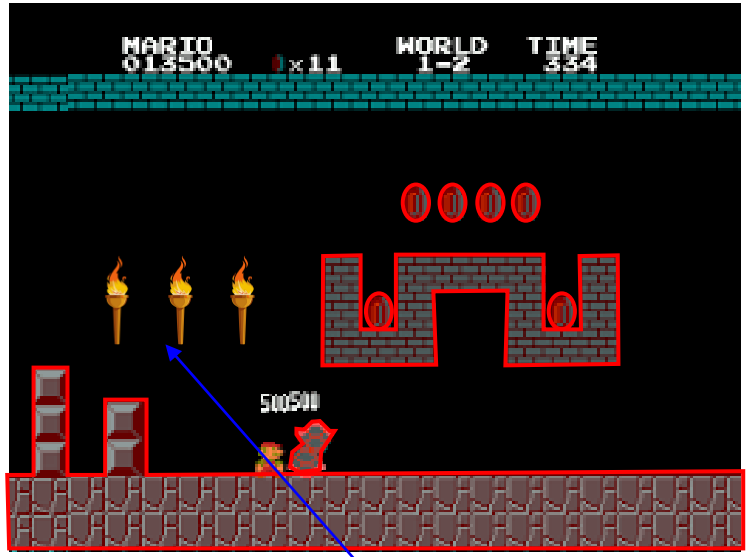
- Inverse dynamics modeling으로 학습된 embedding network는 에이전트에 영향을 미치지 않는 정보는 무시하고 특징을 추출

State No.165



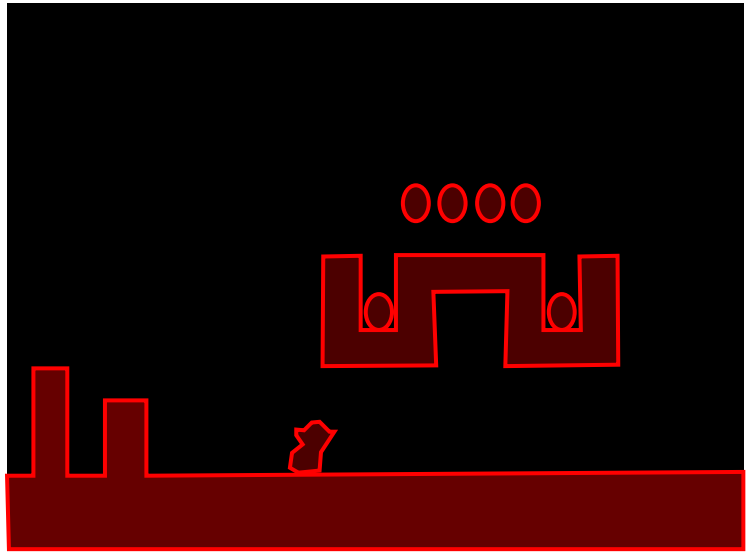
에이전트에게 영향을 미치는 객체를 파악함

State No.276



단순 배경은 에이전트에 영향을 안 미침 (ex. 횃불)

Controllable State



State no.165, 276은 같은 controllable state를 가짐

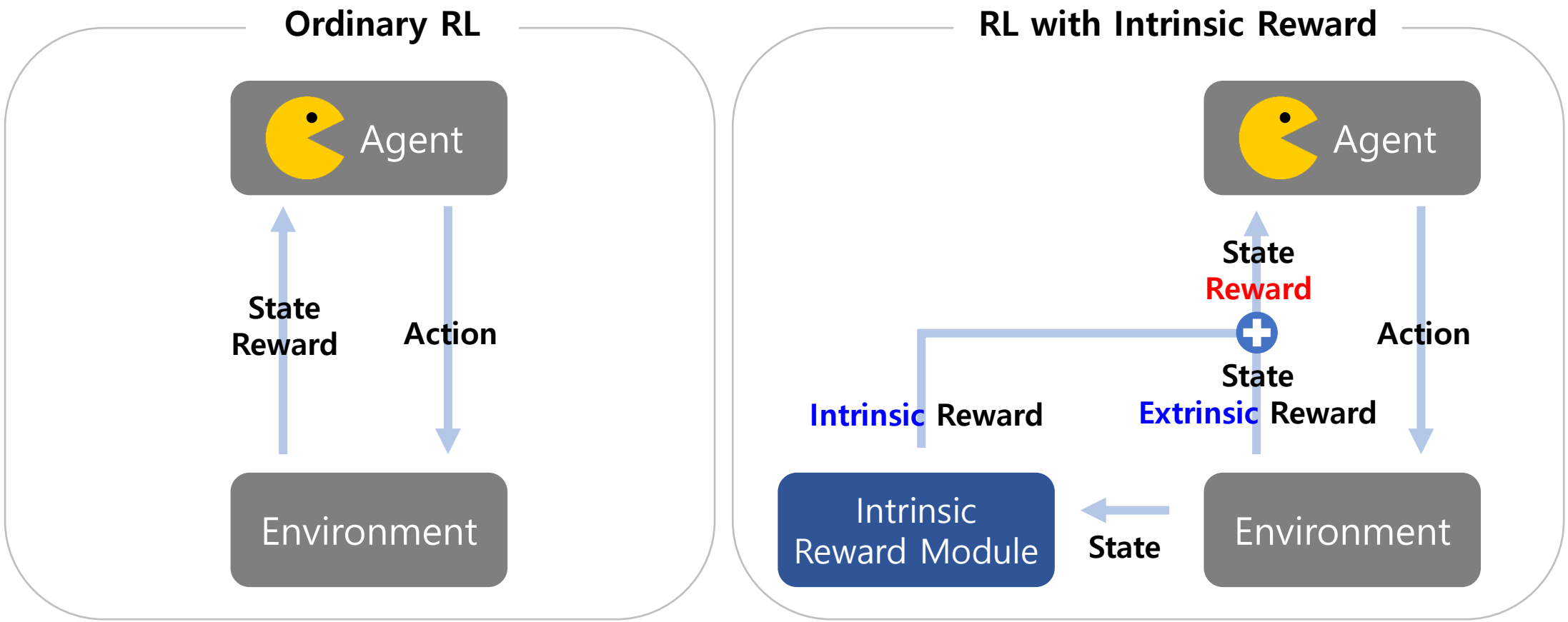
Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." *International conference on machine learning*. PMLR, 2017.

Agent57 Lineage

Never give up (NGU)

❖ Overview of using intrinsic reward

- 환경에서 제공하는 보상(extrinsic reward)과 모델에서 제공되는 보상(intrinsic reward)을 더하여 정책을 업데이트



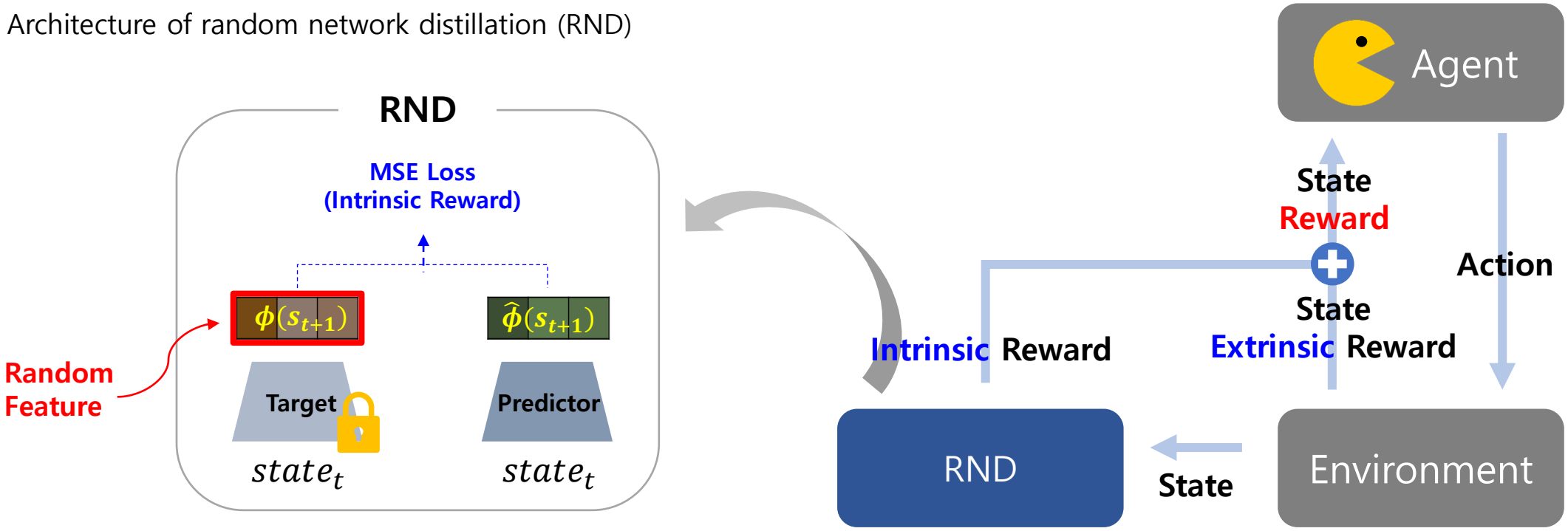
Agent57 Lineage

Never give up (NGU)

❖ Curiosity as an intrinsic reward

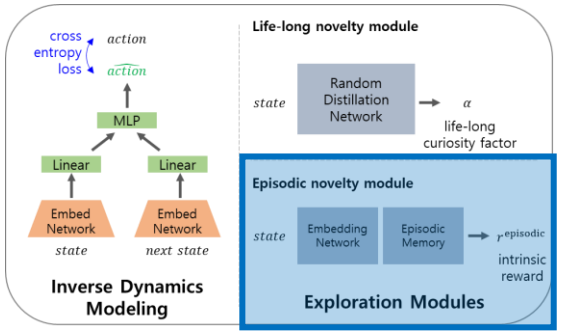
- 호기심(curiosity)은 상태에 대해 불완전한 정보를 가지고 있을 수록 값이 커지는 지표로써 이 값이 큰 상태 위주로 탐험하도록 모델을 유인
- 예측 모델의 예측 오차 혹은 출력 값의 분산을 기반으로 호기심을 계산하며 intrinsic reward로 활용

** Architecture of random network distillation (RND)



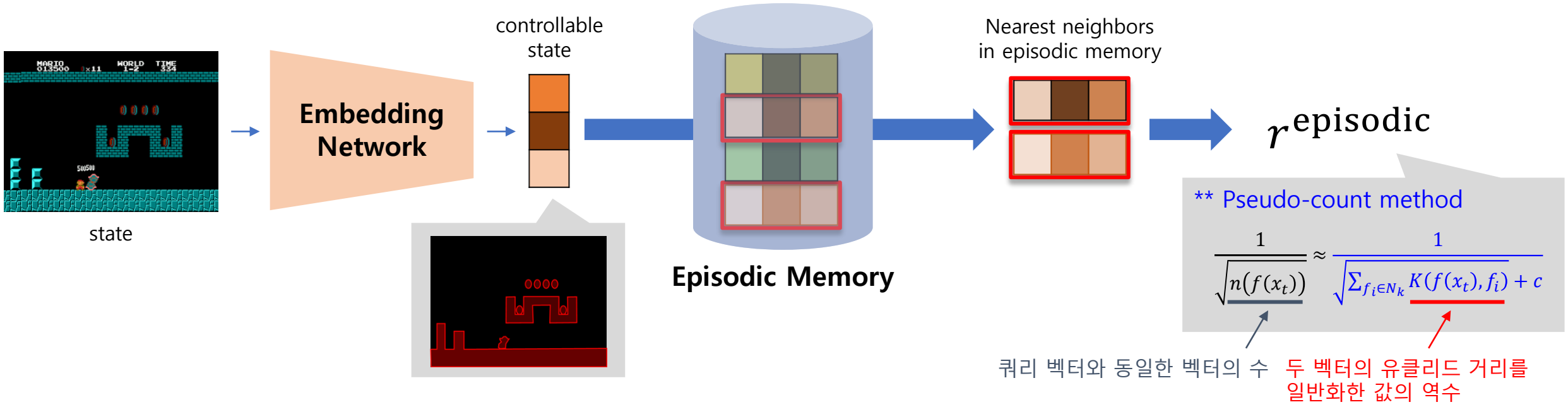
Agent57 Lineage

Never give up (NGU)



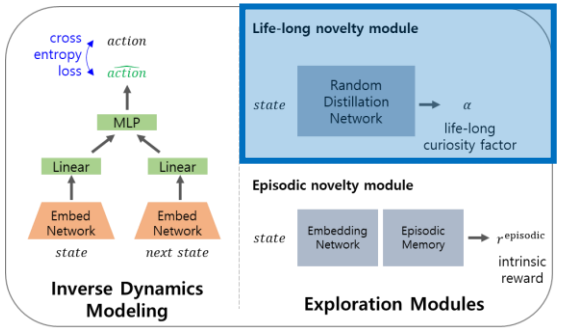
❖ Exploration module: Episodic novelty module

- 매 상호작용마다 임베딩된 controllable state를 episodic memory에 저장
- 이 때 episodic memory는 하나의 에피소드에서 생성된 controllable state에 한정되며 에피소드 내에서의 호기심을 정량화($r^{episodic}$) 함
- 입력된 controllable state와 유사한 memory를 일정 개수 불러온 뒤 유클리드 거리를 기반으로 $r^{episodic}$ 을 구함



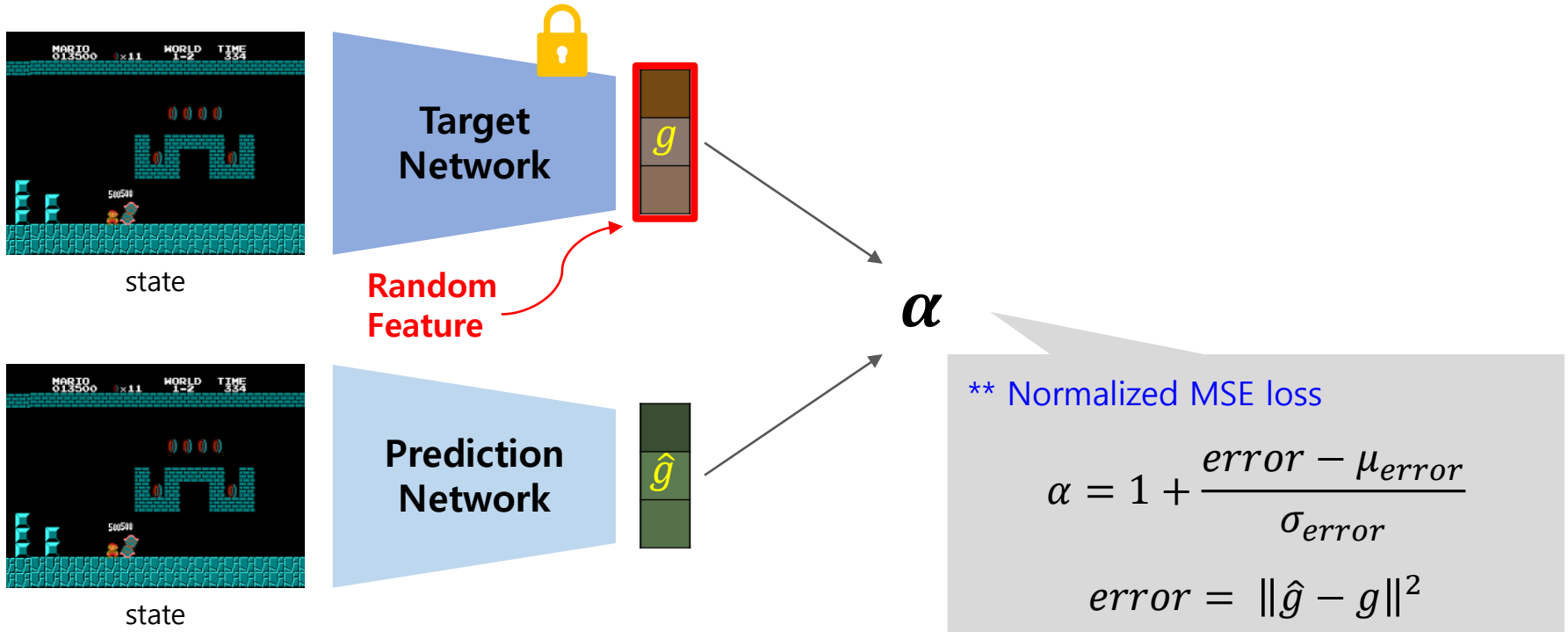
Agent57 Lineage

Never give up (NGU)



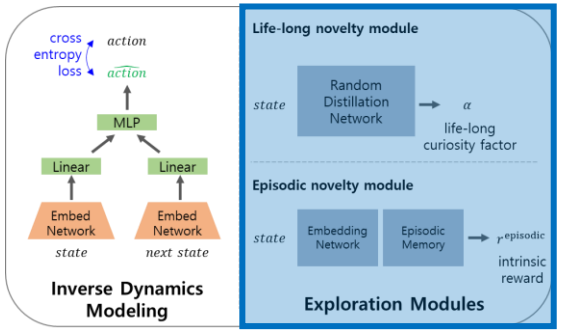
❖ Exploration module: Life-long novelty module

- RND를 수행하여 얻은 mean squared error 값을 일반화하여 **life-long curiosity factor (α)**로 사용
- 모든 에피소드의 데이터로 네트워크를 학습하기 때문에 α를 구할 때 다른 에피소드에서 학습한 내역의 영향을 받음
- 따라서 다른 에피소드에서도 자주 발생했던 경험일수록 α 값이 작아짐



Agent57 Lineage

Never give up (NGU)



❖ Exploration module: Final intrinsic reward

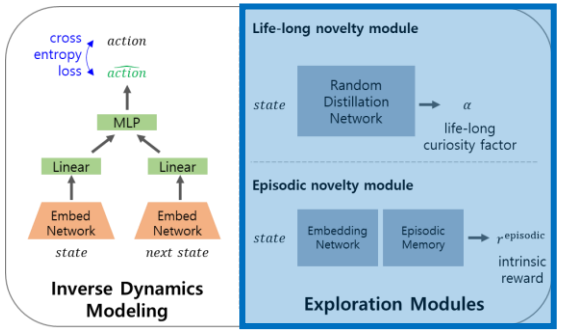
- Episodic intrinsic reward (r^{episodic})와 life-long curiosity factor (α) 를 가중하여 최종 intrinsic reward를 생성
- Life-long curiosity factor는 최소 1, 최대 5의 값을 가짐

$$r^{\text{instinsic}} = r^{\text{episodic}} \cdot \min\{\max\{\alpha, 1\}, \text{L}\}$$

* L은 최대 보상 조정 값(max reward scaling)으로 모든 실험에서 5로 고정

Agent57 Lineage

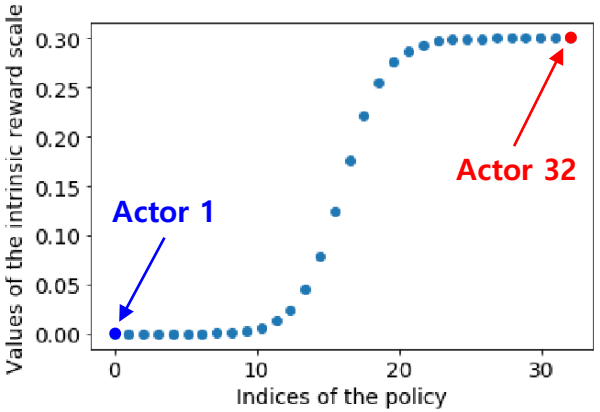
Never give up (NGU)



❖ NGU: Total reward (extrinsic reward + intrinsic reward)

- β를 통해서 intrinsic reward의 비중을 조절
- 각 actor 마다 서로 다른 β_i 을 가짐으로써 다양한 경험을 수집할 수 있음 ($\beta_i = \beta$인 경우 exploratory한 정책, $\beta_i = 0$인 경우 exploitative한 정책)
- β_i 는 최소 0, 최대 β 의 값을 가짐

$$r^{\beta_i} = r^{\text{extrinsic}} + \left(\beta_i * r^{\text{intrinsic}}\right)$$



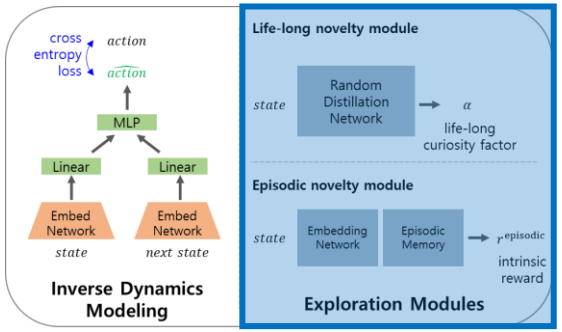
(a) Values taken by the $\{\beta_i\}_{i=0}^{N-1}$

$$\beta_i = \begin{cases} 0 & \text{if } i = 0 \\ \beta & \text{if } i = N-1 \\ \beta \cdot \sigma(10\frac{2i-(N-2)}{N-2}) & otherwise \end{cases}$$

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., ... & Blundell, C. Never Give Up: Learning Directed Exploration Strategies. In International Conference on Learning Representations.

Agent57 Lineage

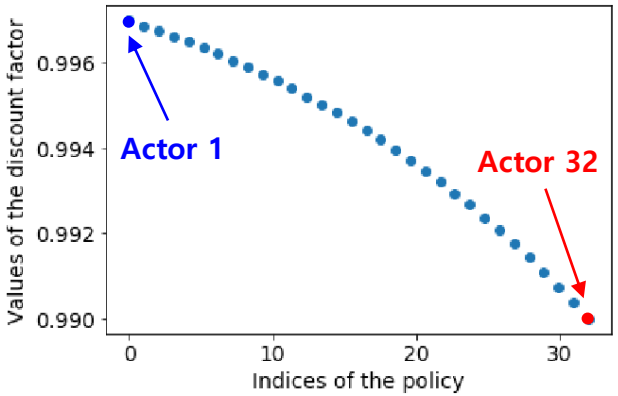
Never give up (NGU)



❖ NGU: RL loss

- γ을 통해서 return에 대한 discount factor를 설정 (시점 별 가중치에 대한 중요도 조절)
- Exploration rate에 따른 정책의 특징에 맞추어 discount factor를 적용
  - ✓ Exploratory policy: 보상이 촘촘하게 주어지며 값의 범위가 넓지 않음
  - ✓ Exploitative policy: 최대한 먼 시점의 보상까지도 고려하여 정책을 학습해야 함

$$L(\theta) = \mathbb{E}\left[\left(r + \gamma_i \max_{a'} Q_\theta(s', a') - Q_\theta(s, a)\right)^2\right]$$



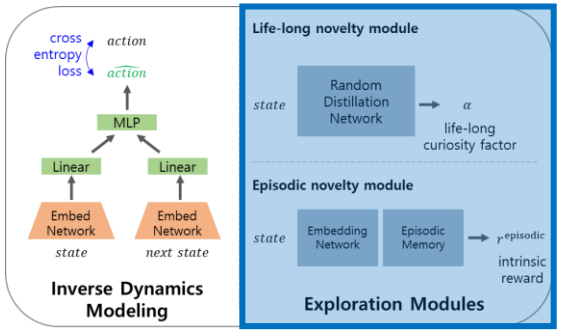
(b) Values taken by the $\{\gamma_i\}_{i=0}^{N-1}$

$$\gamma_i = 1 - \exp\left(\frac{(N-1-i)\log(1-\gamma_{\max}) + i\log(1-\gamma_{\min})}{N-1}\right)$$

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., ... & Blundell, C. Never Give Up: Learning Directed Exploration Strategies. In International Conference on Learning Representations.

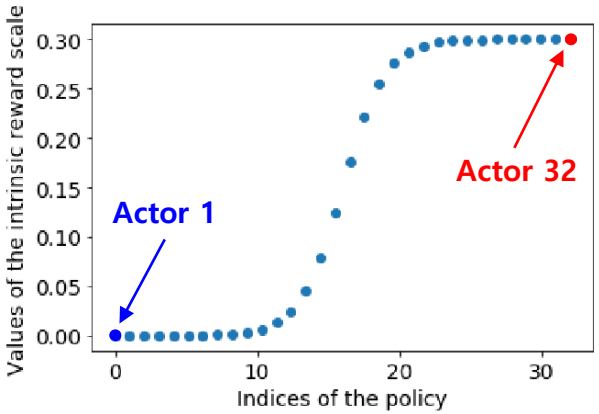
Agent57 Lineage

Never give up (NGU)

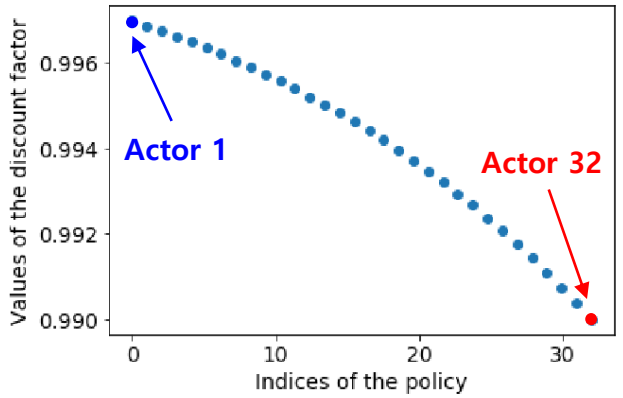


❖ NGU: Pairs of β and γ

- Actor 1은 exploitative하고 미래의 보상을 좀 더 고려하는 정책 (하지만 epsilon은 높음)
- Actor 32(N번 째)는 exploratory하고 현재의 보상을 좀 더 고려하는 정책 (하지만 epsilon은 낮음)



(a) Values taken by the $\{\beta_i\}_{i=0}^{N-1}$



(b) Values taken by the $\{\gamma_i\}_{i=0}^{N-1}$

Actor	1	2	3	...	32
β	0	0.00002	0.00005	...	0.3
γ	0.997	0.99688	0.99676	...	0.99

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., ... & Blundell, C. Never Give Up: Learning Directed Exploration Strategies. In International Conference on Learning Representations.

Agent57 Lineage

Never give up (NGU)

❖ Experiment (performance comparison – Hard exploration)

- 탐험이 어려운 일부 환경(Pitfall, Montezuma's revenge)에서 R2D2를 압도하는 경우를 보임
- Life-long curiosity factor가 제거될 경우 위의 환경에서 성능이 급격하게 하락하는 것을 확인

Algorithm	Gravitar	MR	Pitfall!	PrivateEye	Solaris	Venture
Human	3.4k	4.8k	6.5k	69.6k	**12.3k**	1.2k
Best baseline	**15.7k**	**11.6k**	0.0	11k	5.5k	2.0k
RND	3.9k	10.1k	-3	8.7k	3.3k	1.9k
R2D2+RND	15.6k±0.6k	10.4k±1.2k	-0.5±0.3	19.5k±3.5k	4.3k±0.6k	**2.7k±0.0k**
R2D2(Retrace)	13.3k±0.6k	2.3k±0.4k	-3.5±1.2	32.5k±4.7k	6.0k±1.1k	2.0k±0.0k
NGU(N=1)-RND	12.4k±0.8k	3.0k±0.0k	**15.2k±9.4k**	40.6k±0.0k	5.7k±1.8k	46.4±37.9
NGU(N=1)	11.0k±0.7k	8.7k±1.2k	9.4k±2.2k	60.6k±16.3k	5.9k±1.6k	876.3±114.5
NGU(N=32)	14.1k±0.5k	10.4k±1.6k	8.4k±4.5k	**100.0k±0.4k**	4.9k±0.3k	1.7k±0.1k

Table 1: Results against exploration algorithm baselines. Best baseline takes the best result among R2D2 (Kapturowski et al., 2019), DQN + PixelCNN (Ostrovski et al., 2017), DQN + CTS (Bellemare et al., 2016), RND (Burda et al., 2018b), and PPO + CoEx (Choi et al., 2018) for each game.

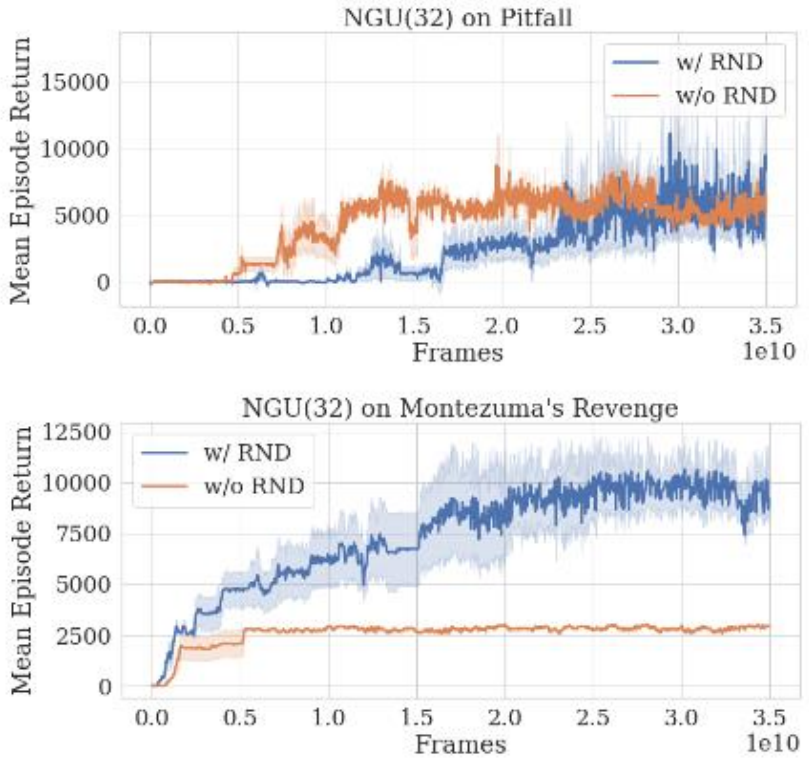


Figure 5: Mean episodic return for agents trained (left) *Pitfall!* and (right) *Montezuma's Revenge*.

Agent57 Lineage

Never give up (NGU)

❖ Experiment (performance comparison – Total)

- 하지만 전반적인 성능을 비교했을 때 R2D2가 많이 우월함
- 탐험 방법론이 추가 되었을 때 기존 성능을 떨어뜨리는 고질적인 문제점 → Agent57에서 해결

Game	R2D2(Retrace)	NGU(32) eval beta=0.0	NGU(32) eval beta=0.3
alien	189.1k±15.6k	**248.1k±22.4k**	225.5k±36.9k
asteroids	**338.5k±4.5k**	230.5k±4.0k	181.6k±2.8k
time pilot	**446.8k±3.9k**	344.7k±31.0k	336.1k±32.9k
tutankham	**452.2±20.0**	191.1±1.2	125.7±5.2
up n down	**678.8k±1.3k**	620.1k±13.7k	575.2k±10.4k
venture	**2.0k±0.0k**	1.7k±0.1k	779.9±188.7
video pinball	948.6k±5.2k	**965.3k±12.8k**	596.7k±120.0k
wizard of wor	**120.2k±7.6k**	106.2k±7.0k	85.1k±12.3k
atlantis	**1654.9k±1.5k**	1653.6k±2.3k	1638.0k±4.6k
yars revenge	990.4k±2.9k	986.0k±3.2k	**993.8k±1.5k**
zaxxon	94.8k±23.7k	111.1k±11.8k	**190.1k±6.9k**
bank heist	15.0k±5.7k	**17.4k±12.0k**	1.4k±0.0k
battle zone	**733.1k±45.2k**	691.7k±22.6k	571.5k±45.4k
beam rider	**103.9k±1.1k**	63.6k±8.6k	31.9k±0.6k
berzerk	**69.3k±5.8k**	36.2k±4.7k	27.3k±0.8k
bowling	**253.0±1.2**	211.9±8.4	161.8±4.9
boxing	**100.0±0.0**	99.7±0.0	3.8±1.4
breakout	**839.7±6.0**	559.2±29.0	387.6±51.0
centipede	**700.2k±19.8k**	577.8k±3.1k	574.6k±7.0k
chopper command	**999.9k±0.0k**	999.9k±0.0k	974.1k±14.7k
amidar	**28.2k±0.2k**	17.8k±1.2k	8.5k±0.4k
crazy climber	294.2k±25.7k	313.4k±7.7k	**357.0k±8.3k**
defender	**675.6k±3.2k**	664.1k±1.8k	656.3k±7.2k
demon attack	**143.9k±0.0k**	143.5k±0.0k	136.6k±3.3k
double dunk	**24.0±0.0**	-14.1±2.4	-21.5±0.7
enduro	**2.4k±0.0k**	2.0k±0.1k	682.5±24.0
fishing derby	**86.8±0.9**	32.0±1.9	-29.3±4.1
freeway	**33.2±0.1**	28.5±1.1	21.2±0.5
frostbite	10.5k±3.8k	**206.4k±150.0k**	67.9k±45.6k
gopher	**117.8k±0.9k**	113.4k±0.6k	86.6k±7.4k
gravitar	12.9k±0.6k	**14.2k±0.5k**	13.2k±0.3k
assault	**36.7k±3.0k**	34.8k±5.4k	1.9k±0.6k
hero	54.5k±3.0k	**69.4k±5.7k**	64.2k±7.0k
ice hockey	**85.1±0.6**	-4.1±0.3	-0.5±0.2
jamesbond	**30.8k±2.3k**	26.6k±2.6k	22.4k±0.3k
kangaroo	14.7k±0.1k	**35.1k±2.1k**	16.9k±2.0k
krull	**131.1k±12.7k**	127.4k±17.1k	26.7k±2.2k
kung fu master	**220.7k±3.9k**	212.1k±11.2k	203.2k±10.8k
montezuma revenge	2.3k±0.4k	10.4k±1.5k	**16.8k±6.8k**
ms pacman	40.3k±1.1k	**40.8k±1.1k**	38.1k±1.6k
name this game	**70.6k±8.3k**	23.9k±0.5k	15.6k±0.3k
phoenix	935.2k±13.5k	**959.1k±2.7k**	933.3k±4.7k
asterix	**994.3k±0.9k**	950.7k±23.1k	953.1k±21.7k
pitfall	-5.2±1.7	**7.8k±4.0k**	1.3k±0.9k
pong	**20.9±0.0**	19.6±0.2	-9.7±1.4
private eye	31.5k±5.4k	**100.0k±0.4k**	65.6k±14.3k
qbert	398.6k±46.4k	**451.9k±82.1k**	449.0k±108.2k
riverraid	38.9k±0.6k	36.7k±2.3k	**42.6k±4.0k**
road runner	105.2k±16.8k	**128.6k±28.7k**	103.9k±7.1k
robotank	**142.4±0.8**	9.1±0.7	24.9±2.5
seaquest	**1000.0k±0.0k**	1000.0k±0.0k	1000.0k±0.0k
skiing	**-11081.7±97.7**	-22977.9±5059.9	-21907.4±3647.5
solaris	**5.6k±1.1k**	4.7k±0.3k	1.3k±0.1k
space invaders	33.4k±12.2k	**43.4k±1.3k**	12.0k±1.9k
star gunner	412.8k±4.1k	414.6k±66.8k	**452.5k±53.0k**
surround	**9.8±0.0**	-9.6±0.2	-7.3±0.4
tennis	**24.0±0.0**	10.2±3.5	-3.5±5.7

Table 10: Scores over all 57 Atari games.

Agent57 Lineage

Agent57

❖ Agent57: Outperforming the Atari Human Benchmark (ICML 2020, 617회 인용)

- NGU의 저자들이 그대로 참여해 NGU의 단점을 분석하고 이를 보완하는 모델(Agent57)을 만든 연구
- 다양한 특성을 가진 **모든** Atari 게임에서 사람보다 뛰어난 성능을 달성한 최초의 모델
- NGU를 기반으로 탐험 방식과 long-term credit assignment 문제를 보완

(Atari benchmark)	**NGU (Badia. et al., 2020)**	**Agent57 (Badia. et al., 2020)**
# of frame stack	1	1
Separated Q-learning head	No	Yes
Use meta-controller to select the policy	No	Yes
discount factor (γ) range	0.99 ~ 0.997	0.99 ~ 0.9999
reward weight (β)	0.3	0.3

Agent57 Lineage

Agent57

❖ NGU의 학습이 불안정하거나 낮은 성능이 나오는 결과 분석

- NGU에서 낮은 성능을 달성할 때의 원인으로 각 actor마다 부여하는 서로 다른 discount factor (γ)와 exploration rate (β) 방식이 지목됨

✓ 원인 1: 모델의 학습 진행 상황과 무관하게 각 actor가 동일한 양의 경험을 수집

✓ 제안 1: 각 게임 또는 상황에 맞춰 exploration rate를 조정

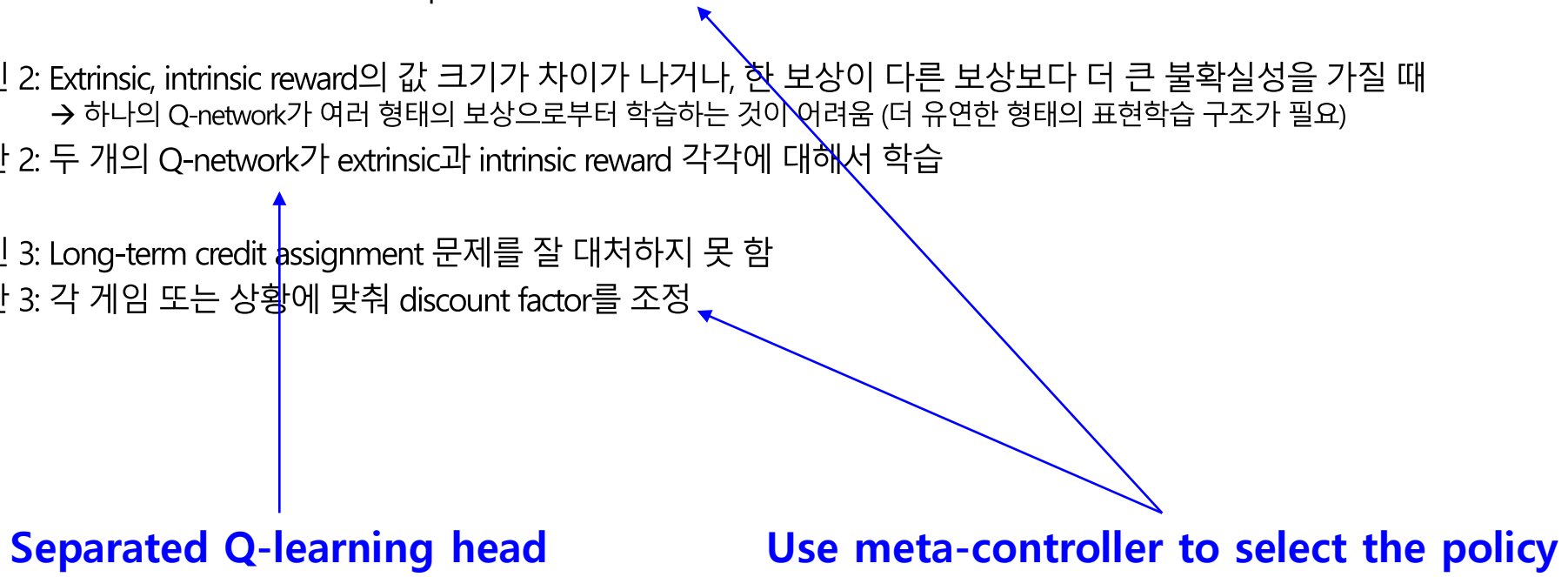
✓ 원인 2: Extrinsic, intrinsic reward의 값 크기가 차이가 나거나, 한 보상이 다른 보상보다 더 큰 불확실성을 가질 때

→ 하나의 Q-network가 여러 형태의 보상으로부터 학습하는 것이 어려움 (더 유연한 형태의 표현학습 구조가 필요)

✓ 제안 2: 두 개의 Q-network가 extrinsic과 intrinsic reward 각각에 대해서 학습

✓ 원인 3: Long-term credit assignment 문제를 잘 대처하지 못 함

✓ 제안 3: 각 게임 또는 상황에 맞춰 discount factor를 조정



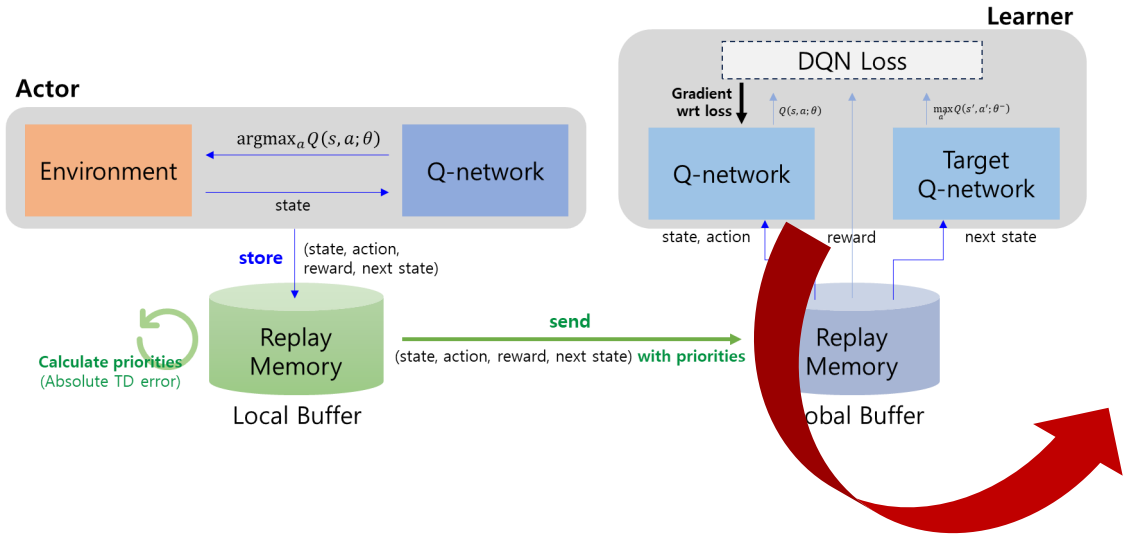
Agent57 Lineage

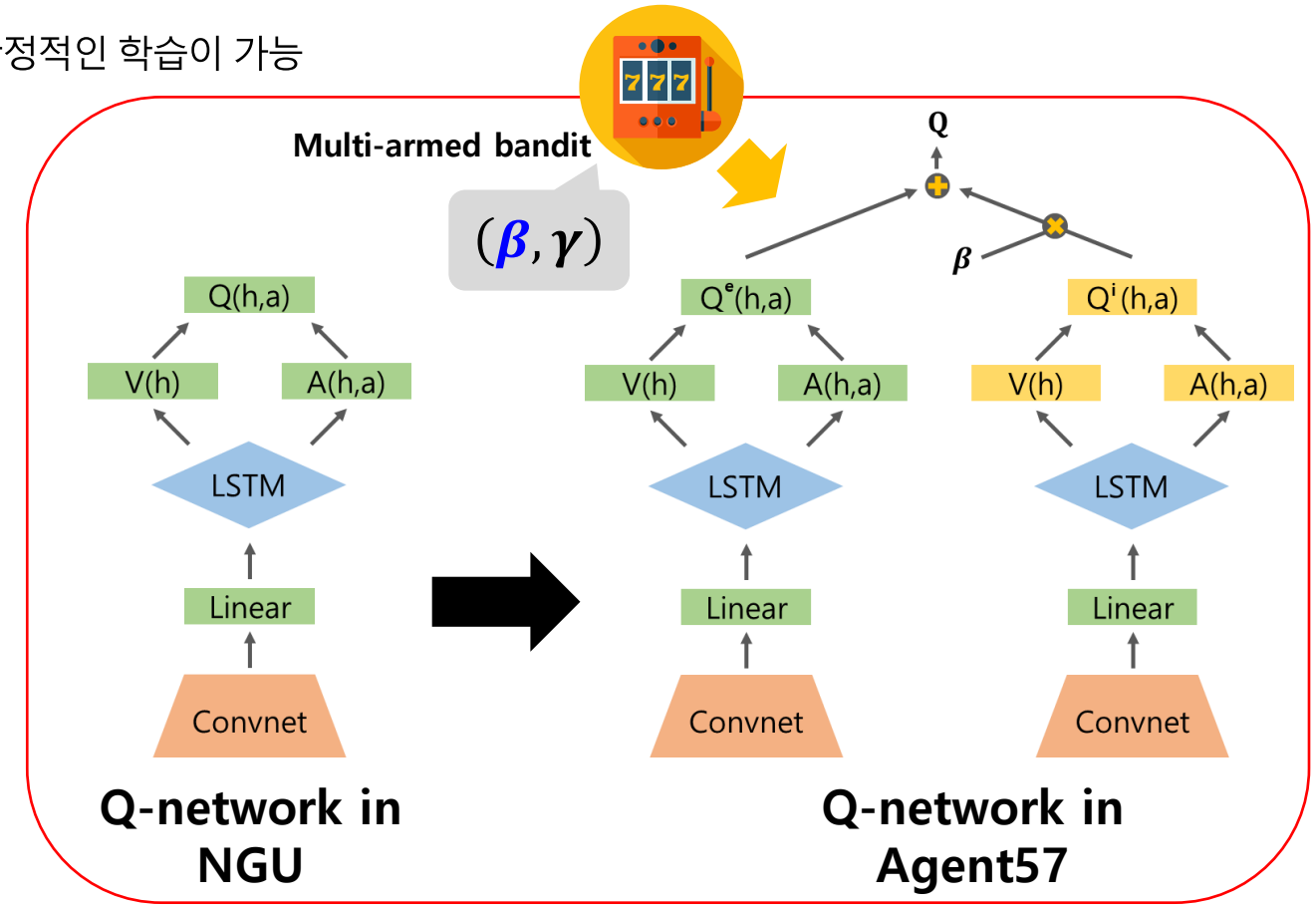
Agent57

❖ **Extrinsic / intrinsic 보상을 학습하도록 서로 다른 파라미터를 갖는 두 개의 Q-network를 사용**

- 서로 다른 네트워크로 각각의 보상을 학습하기 때문에 더 안정적인 학습이 가능
- Exploration rate를 통해서 상황에 따라 정책의 성향을 조정 (exploitative policy vs. exploratory policy)

** Architecture of Agent57 (based on NGU)





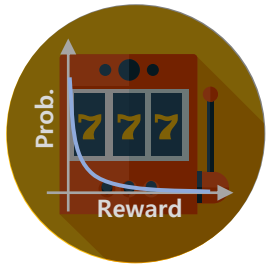
Extrinsic/intrinsic reward로 업데이트하는 각각의 Q-learning head를 사용

Agent57 Lineage

Agent57

❖ Multi-armed bandit (MAB)을 이용하여 상황에 알맞은 정책을 선택

- 제한된 기회 안에서 여러 arm들을 시도해보면서 평균적으로 높은 점수를 주는 arm을 찾는 방법
  - ✓ 우선 모든 arm을 모두 시도하여 얻게 되는 보상을 기록
  - ✓ 가장 높은 가치를 주었던 arm을 여러 번 선택하여 정확한 보상 분포를 파악하거나 다른 arm을 시도하여 얻게 되는 보상을 확인
- 일반적인 MAB는 모든 시도를 기록하지만 **sliding window MAB**는 일정 window에서 수행한 시도의 보상만으로 arm의 가치를 산정

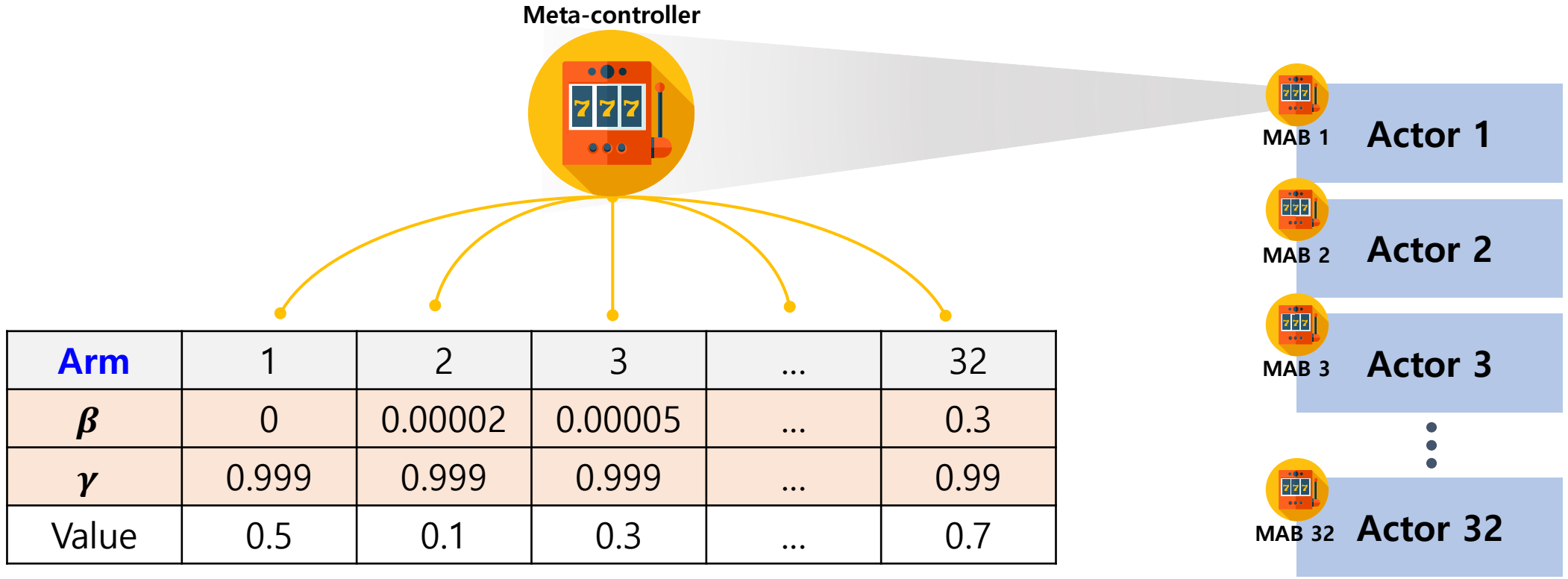
Times selected	1	2	3	1	1
Value	0.5	0.11	0.745	0.3	0.4
True reward distribution	 **Bandit (arm 1)**	 **Bandit (arm 2)**	 **Bandit (arm 3)**	 **Bandit (arm 4)**	 **Bandit (arm 5)**

Agent57 Lineage

Agent57

❖ **Meta-controller (sliding window MAB)를 이용하여 상황에 알맞은 정책을 선택**

- 한 에피소드가 시작할 때 MAB의 각 arm은 β와 γ의 여러 조합 중 하나의 쌍을 선택
- 각각의 actor는 고유의 MAB를 가지며 한 에피소드에서 나온 extrinsic reward의 평균으로 arm의 가치를 업데이트



Arm	1	2	3	…	32
β	0	0.00002	0.00005	…	0.3
γ	0.999	0.999	0.999	…	0.99
Value	0.5	0.1	0.3	…	0.7

Agent57 Lineage

Agent57

❖ Experiment (performance comparison)

- Atari 벤치마크에서 제공하는 57개의 게임 중 57개의 게임에서 사람보다 높은 점수를 달성 (figure 1)
- 개선한 점을 통해서 마지막에 hard exploration과 hard long-term credit assignment 문제를 가진 환경도 정복 (figure 3)

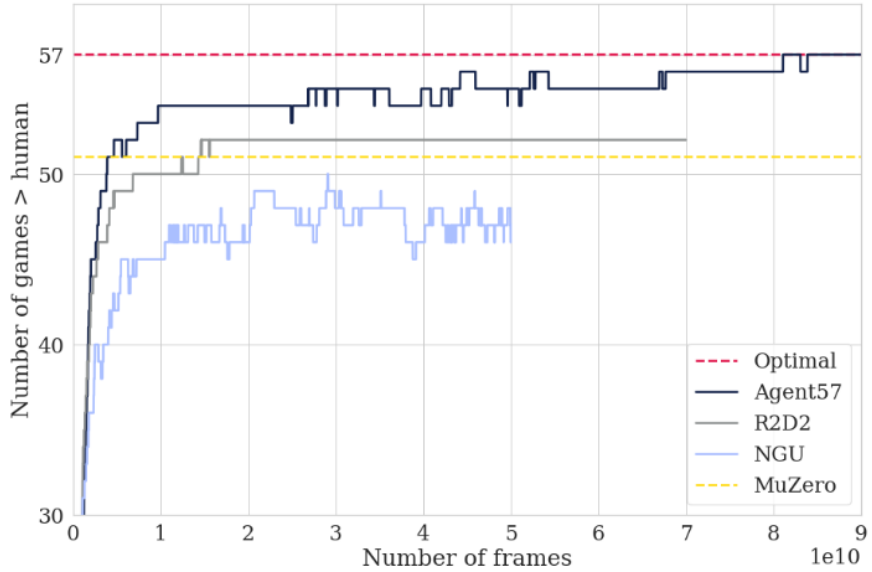


Figure 1. Number of games where algorithms are better than the human benchmark throughout training for Agent57 and state-of-the-art baselines on the 57 Atari games.

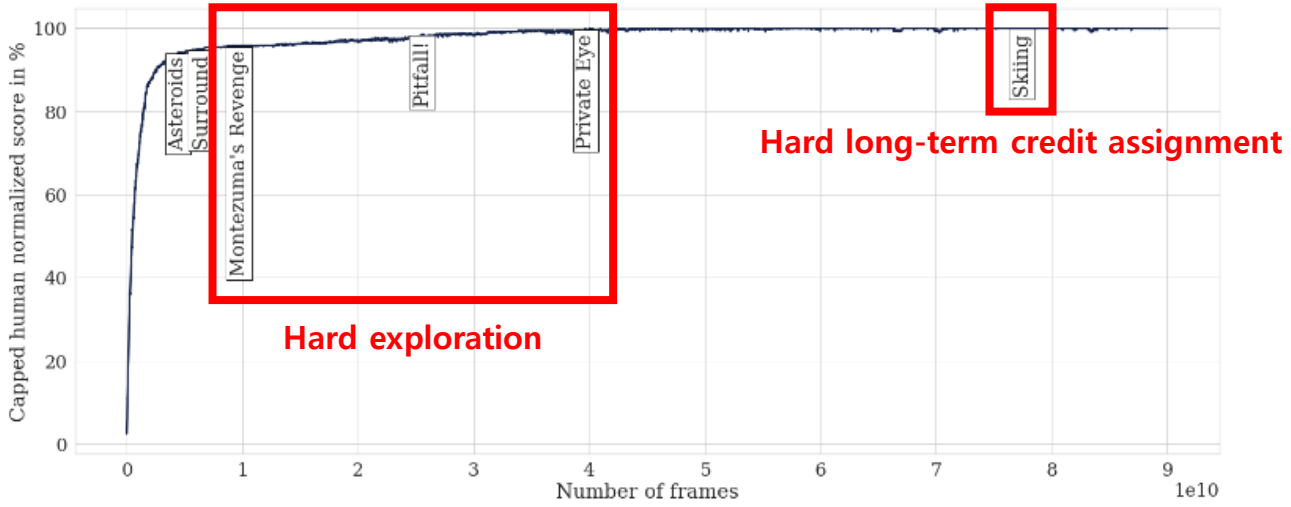


Figure 3. Capped human normalized score where we observe at which point the agent surpasses the human benchmark on the last 6 games.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020, November). Agent57: Outperforming the atari human benchmark. In International conference on machine learning (pp. 507-517). PMLR.

Agent57 Lineage

Agent57

❖ Experiment (meta-controller, separate network)

- Exploration rate와 discount factor가 게임, 학습 진행 상황에 따라 알맞게 선택되는 것을 확인 (figure 8)
- 네트워크를 나누어 학습했을 때 intrinsic reward의 비중이 커지더라도 exploitation policy의 성능이 강건하게 유지되는 것을 확인 (figure 5)

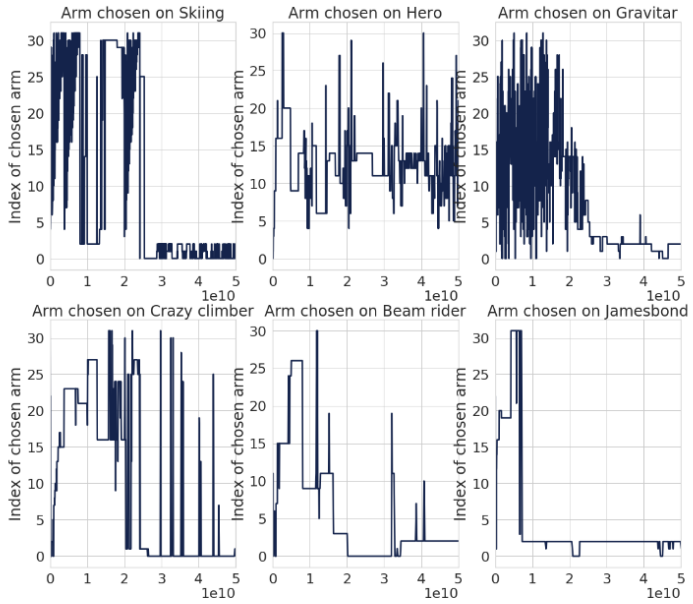


Figure 8. Best arm chosen by the evaluator of Agent57 over training for different games.

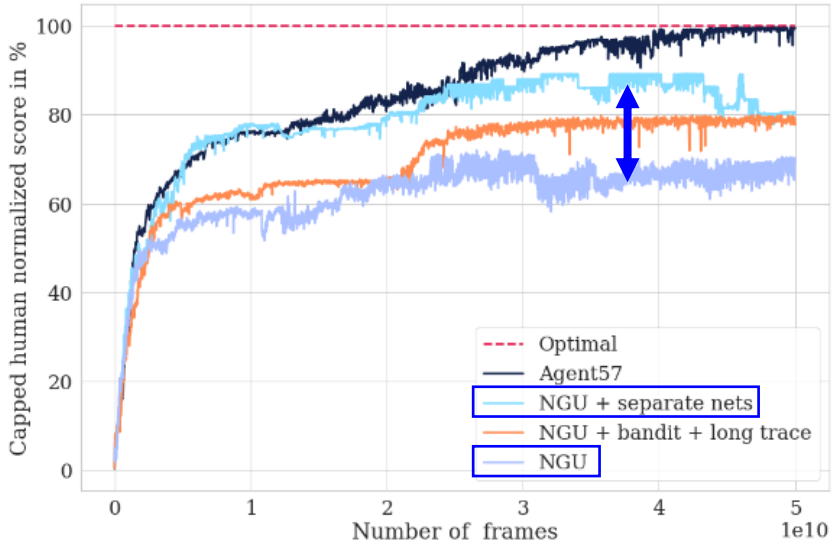


Figure 4. Performance progression on the 10-game *challenging set* obtained from incorporating each one of the improvements.

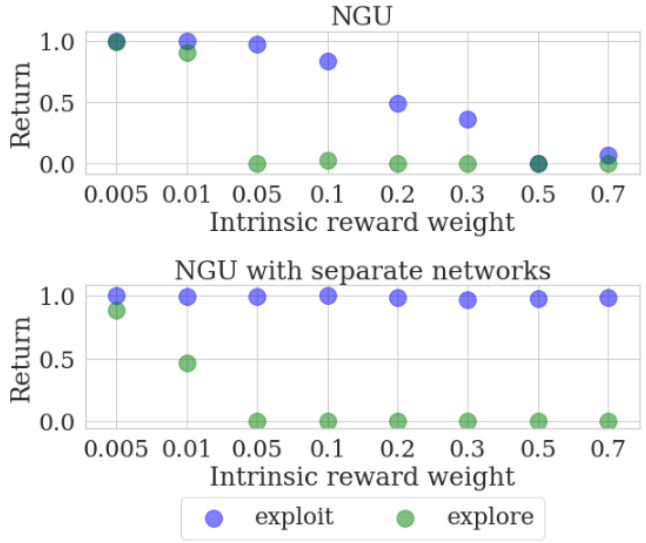


Figure 5. Extrinsic returns for the exploitative ($\beta_0 = 0$) and most exploratory ($\beta_{31} = \beta$) on "random coin" for different values of the intrinsic reward weight, β. *(Top)* NGU*(Bottom)* NGU with Separate networks for intrinsic and extrinsic values.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., & Blundell, C. (2020, November). Agent57: Outperforming the atari human benchmark. In International conference on machine learning (pp. 507-517). PMLR.

Summary

❖ Outperforming Humans with Reinforcement Learning Agents in Atari Games

- DQN이 Atari 벤치마크에서 57개 게임 중 23개에서 사람보다 좋은 성능을 기록 (DQN, 2013 ~ 2015년)
- DQN을 분산 학습으로 더 빠르게 좋은 성능을 갖도록 연구가 이루어짐 (GorilaDQN, Ape-X, R2D2, 2015 ~ 2019년)
- 분산 학습 기반의 환경에서 탐험 방법론을 추가하여 탐험이 어려운 게임에서도 좋은 성능을 달성할 수 있게 함 (NGU, 2020년)
- 탐험 방법론을 추가함으로써 발생하는 모델 성능 저하를 해결하고 하나의 아키텍처가 별도의 수정 없이 모든 게임에서 사람보다 좋은 성능을 달성할 수 있게 됨 (Agent57, 2020년)
- Atari 벤치마크가 나오고 정복되기까지 걸린 시간 - 7년 1개월 * 출간/발표일 기준

Citation

- Bellemare, Marc G., et al. "The arcade learning environment: An evaluation platform for general agents." Journal of Artificial Intelligence Research 47 (2013): 253-279.
- Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.
- Hausknecht, Matthew, and Peter Stone. "Deep recurrent q-learning for partially observable mdps." 2015 aaai fall symposium series. 2015.
- Nair, Arun, et al. "Massively parallel methods for deep reinforcement learning." arXiv preprint arXiv:1507.04296 (2015).
- Horgan, Dan, et al. "Distributed prioritized experience replay." arXiv preprint arXiv:1803.00933 (2018).
- Badia, Adrià Puigdomènech, et al. "Never give up: Learning directed exploration strategies." arXiv preprint arXiv:2002.06038 (2020).
- Kapturowski, Steven, et al. "Recurrent experience replay in distributed reinforcement learning." International conference on learning representations. 2018.
- Badia, Adrià Puigdomènech, et al. "Agent57: Outperforming the atari human benchmark." International conference on machine learning. PMLR, 2020
- Pathak, Deepak, et al. "Curiosity-driven exploration by self-supervised prediction." *International conference on machine learning*. PMLR, 2017.
- Burda, Yuri, et al. "Exploration by random network distillation." *International Conference on Learning Representations*. 2018.