State Representation Learning for Reinforcement Learning

2021. 04. 16

발표자: 이영재

발표자 소개

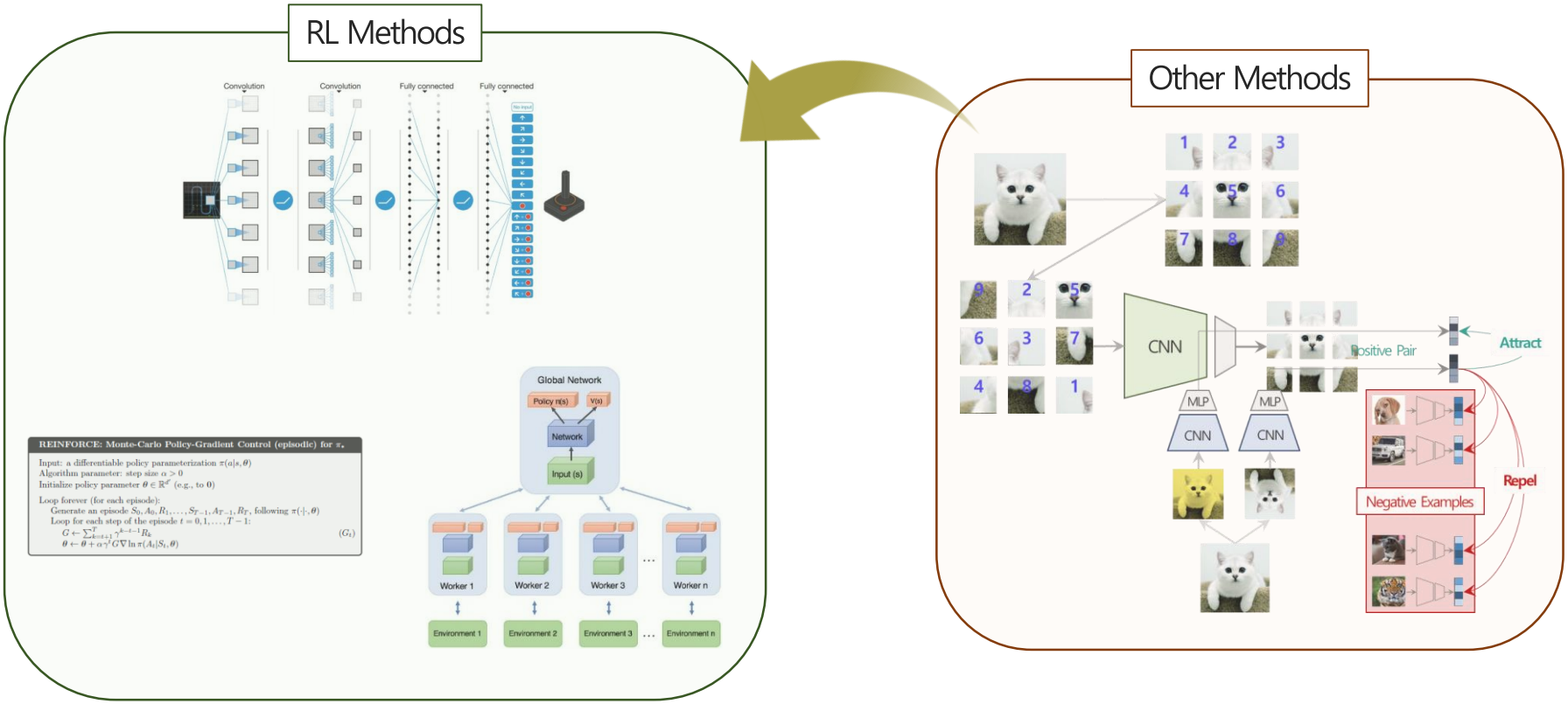
- 이름: 이영재 (Young Jae Lee)
  - Data Mining & Quality Analytics Lab
  - Ph.D. Student (2019.03 ~ Present)
  - 지도 교수: 김성범 교수님
- 연구 분야
  - Deep Reinforcement Learning
  - Multi-Agent Reinforcement Learning
  - Self-Supervised Learning
- 연락망
  - E-mail: jae601@korea.ac.kr

목차

- Representation Learning
- Reinforcement Learning
- Details on Reinforcement Learning
- Combination of Reinforcement Learning and Representation Learning
- Conclusion

❖ State Representation Learning for Reinforcement Learning 이란?

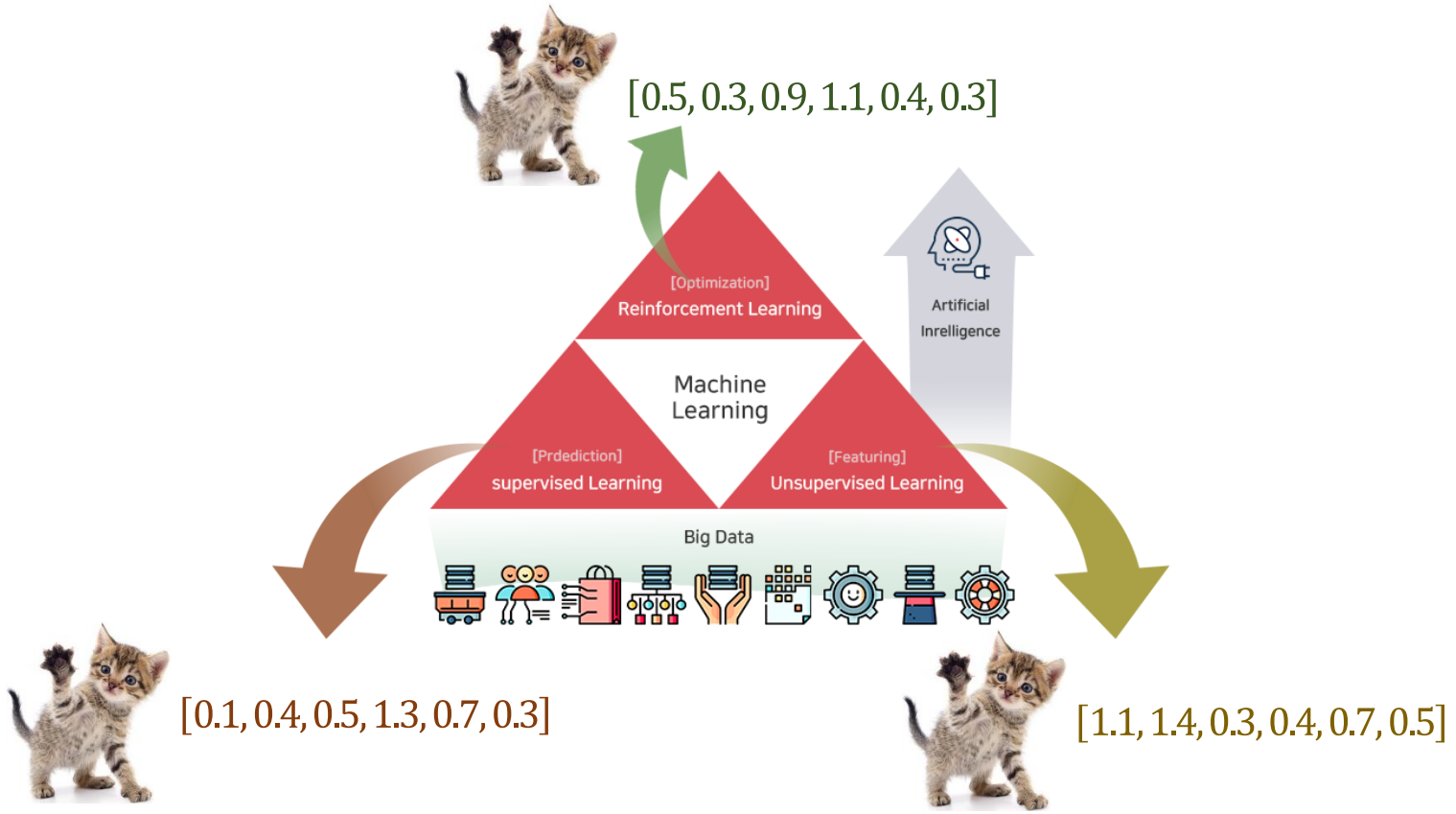
- 강화학습에서 발생하는 문제점을 보완하기 위해 다른 학습 기법을 결합



Representation Learning

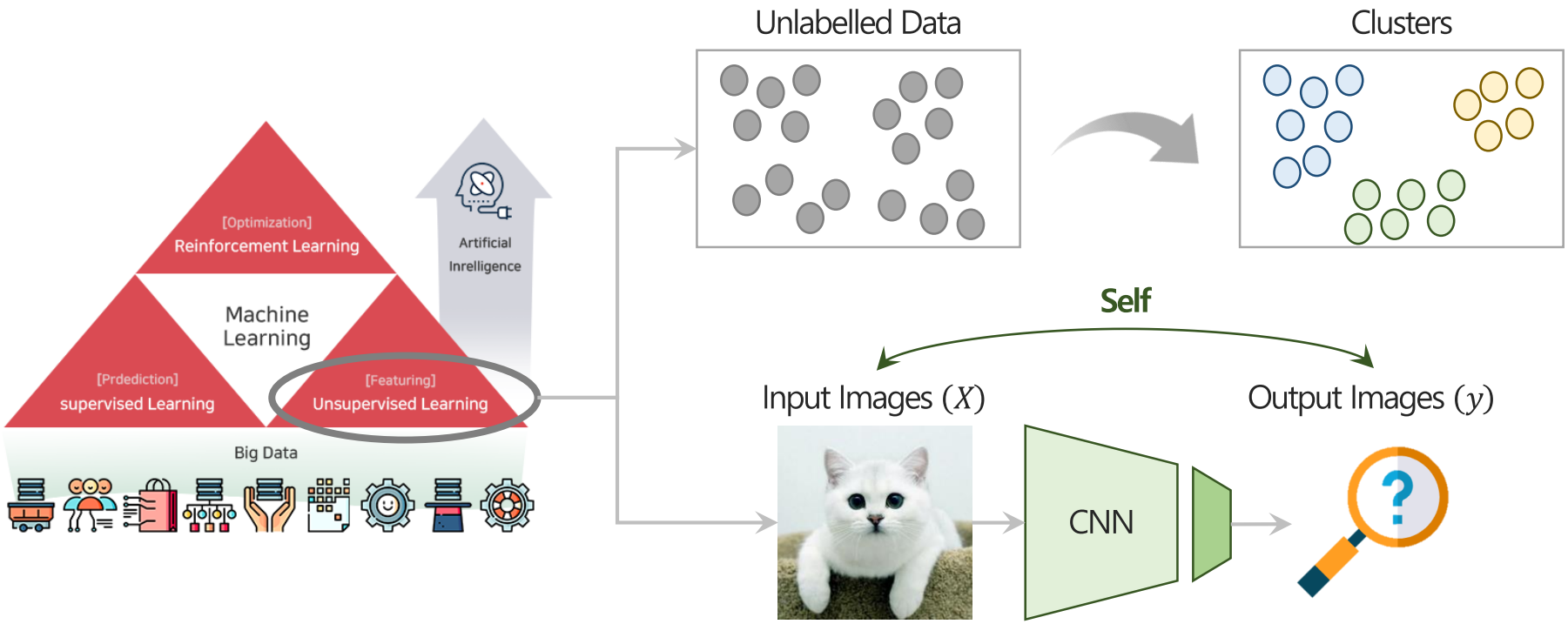
❖ Representation Learning?

- 이미지, 텍스트 등과 같은 데이터를 숫자 형태로 기술하도록 학습
  - ✓ 학습 방법에 따라 데이터를 표현하는 결과는 다를 수 있음



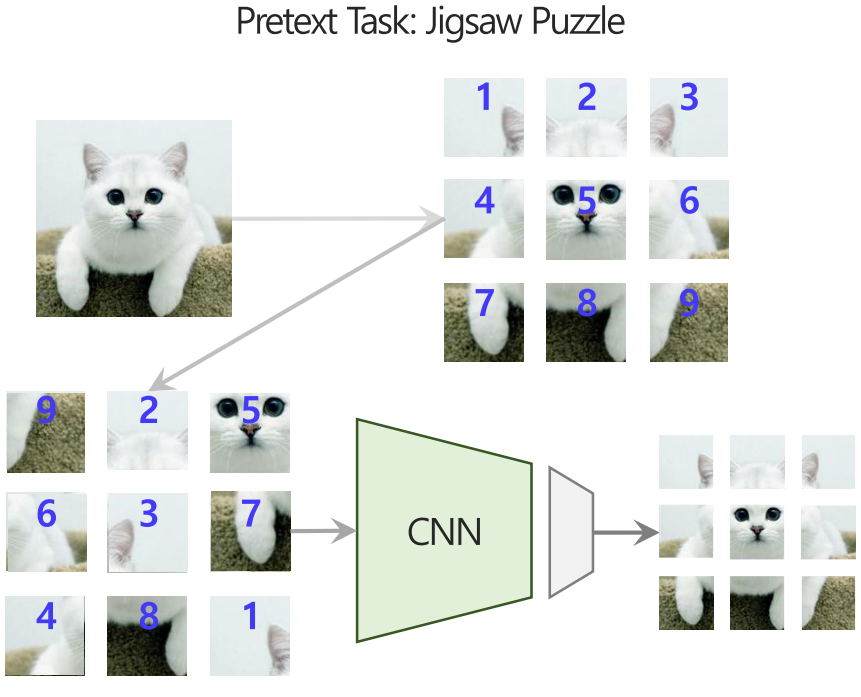
Representation Learning

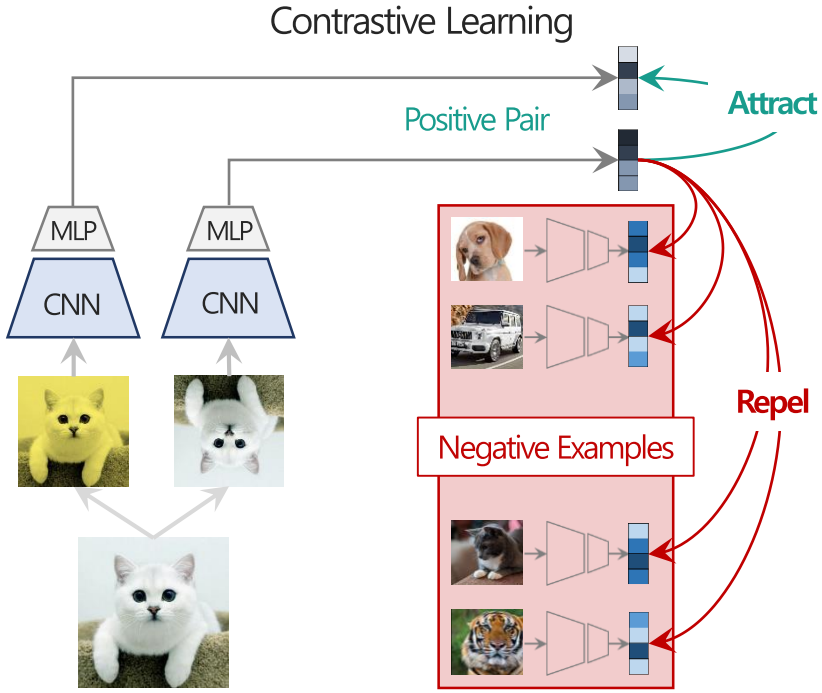
- Unsupervised Learning
  - 레이블 정보(y)가 없는 입력 데이터(X)만을 사용하여 데이터 자체의 좋은 표현을 학습
  - Ex: Clustering, Dimensionality Reduction, **Self-Supervised Learning**, …



Representation Learning

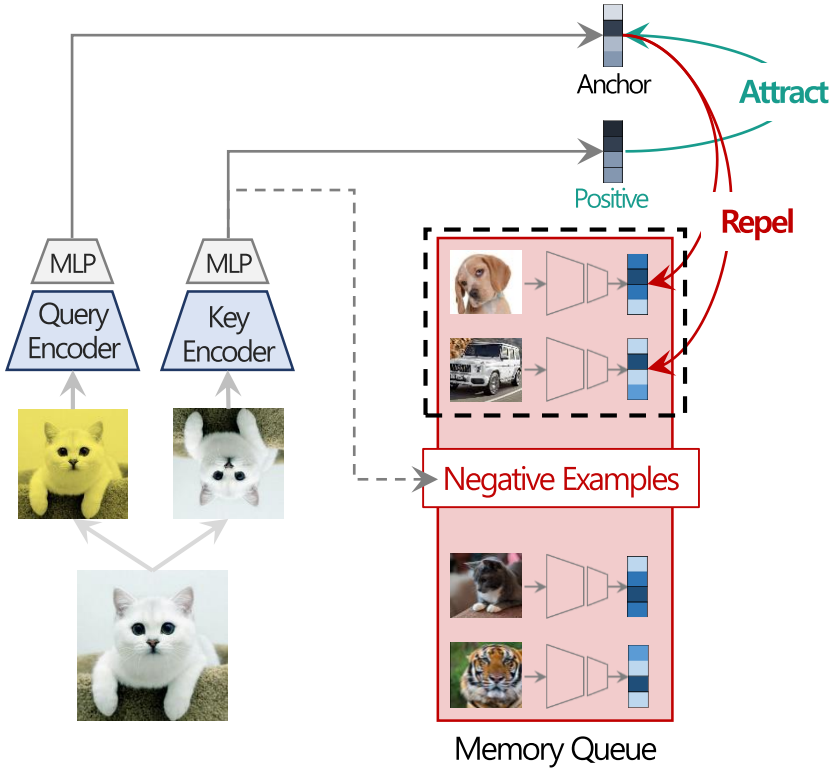
- Self-Supervised Learning
  - Pretext Tasks: Examplar, Context Prediction, Jigsaw Puzzle, Count, Rotation, …
  - Contrastive Learning: MoCo, SimCLR
  - Non-Contrastive Learning: BYOL





Representation Learning

- Self-Supervised Learning
  - Contrastive Learning: MoCo
  - Memory Queue (First In First Out), Momentum Update



$$\mathcal{L}_i = -log \frac{exp\left(\frac{i \cdot i^+}{\tau}\right)}{exp\left(\frac{i \cdot i^+}{\tau}\right) + \sum_{k \notin \{i, i^+\}} exp\left(\frac{i \cdot k}{\tau}\right)}$$

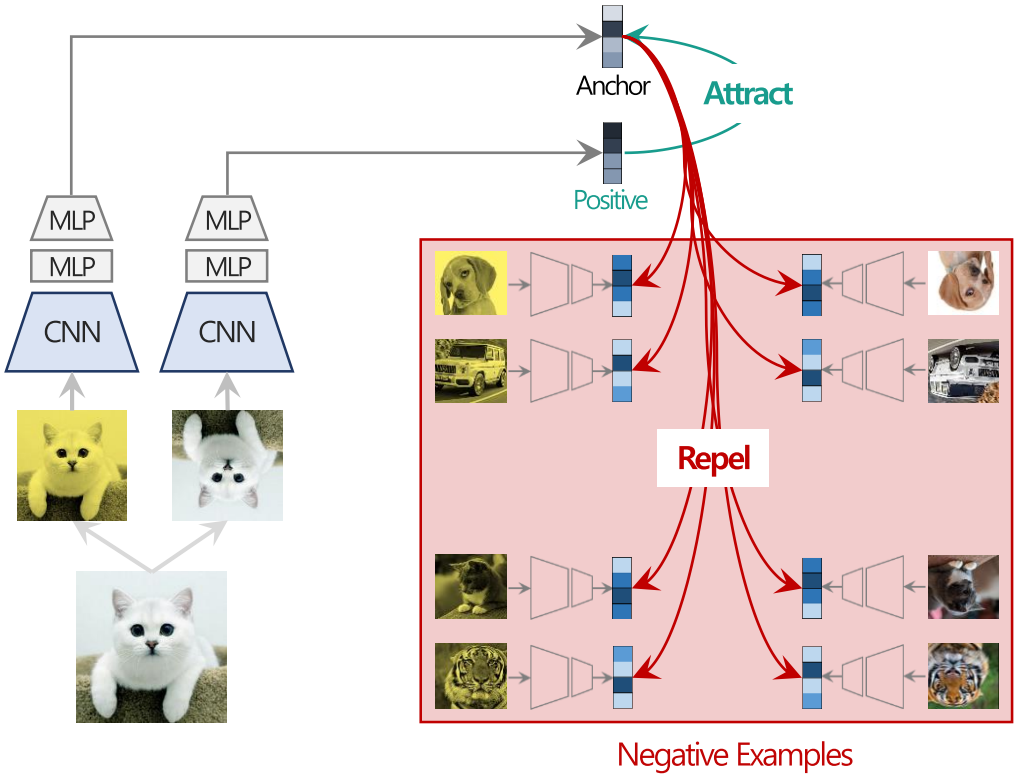
i: Anchor
i^+: Positive
k: Negative Examples

Momentum Update
(Key Encoder)

$$\theta_{key} \leftarrow m\theta_{key} + (1-m)\theta_{query}$$

Representation Learning

- Self-Supervised Learning
  - Contrastive Learning: SimCLR
  - Memory Queue → Batch Size, Data Augmentation, Deep Embedding Layer

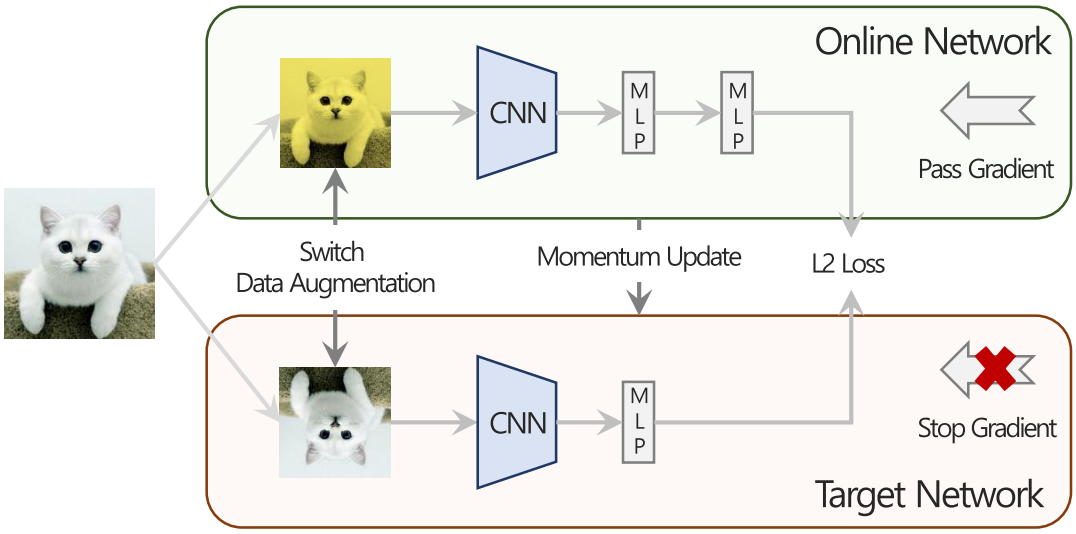


$$\mathcal{L}_i = -\log \frac{exp\left(\frac{i \cdot i^+}{\tau}\right)}{exp\left(\frac{i \cdot i^+}{\tau}\right) + \sum_{k \notin \{i, i^+\}} exp\left(\frac{i \cdot k}{\tau}\right)}$$

i: Anchor
i^+: Positive
k: Negative Examples

Representation Learning

- Self-Supervised Learning
  - Non-Contrastive Learning: BYOL
  - Positive pair (without Negative examples), Momentum Update, L2 Loss



$$\mathcal{L}_{\theta,\delta}^{BYOL} = \left|\left|\overline{q_\theta}(z_\theta) - \overline{z_\delta}'\right|\right|_2^2$$

$$where\ \overline{q_\theta}(z_\theta) \triangleq \frac{q_\theta(z_\theta)}{\left|\left|q_\theta(z_\theta)\right|\right|_2}, \overline{z_\delta}' \triangleq \frac{z_\delta'}{\left|\left|z_\delta'\right|\right|_2}$$

$$Total\ Loss = \mathcal{L}_{\theta,\delta}^{BYOL} + \tilde{\mathcal{L}}_{\theta,\delta}^{BYOL}$$

Momentum Update
(Target Network)

$$\delta_{target} \leftarrow \tau\delta_{target} + (1-\tau)\theta_{online}$$

Reinforcement Learning

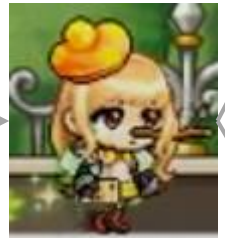
❖ Example of Reinforcement Learning

- 게임 환경: 장애물을 피하며 목적을 달성
- Agent 목표: 좌·우로 날아다니는 장애물을 피하며 목적지까지 도달하는 것

Game Environment: 장애물 피하기

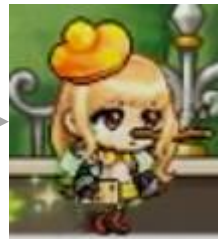
: 장애물

Environment



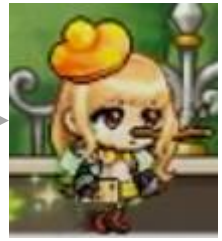
Agent

Action: ↑ + Jump



Reward: +1

Action: ← + Jump

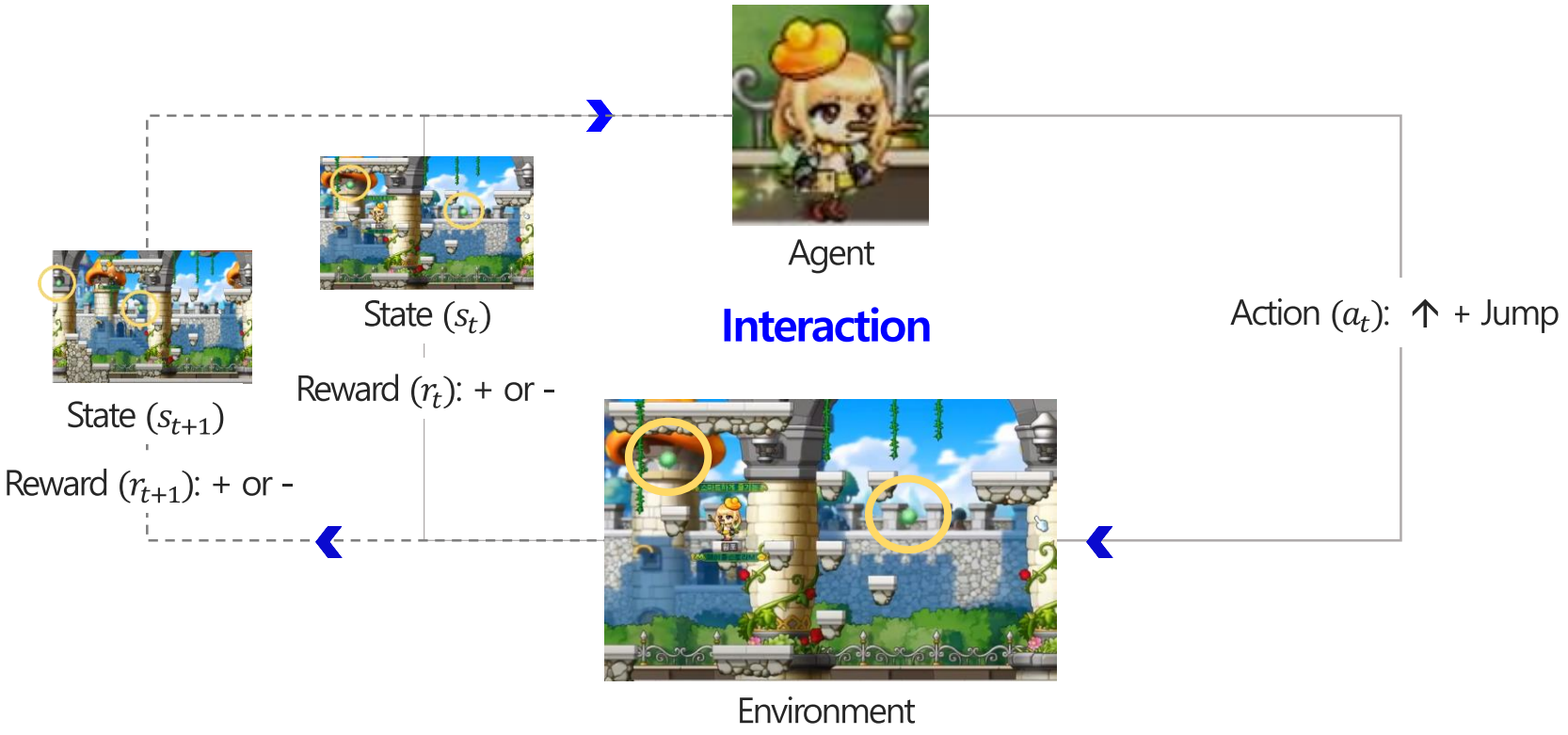


Reward: -1

Reinforcement Learning

❖ Definition of Reinforcement Learning

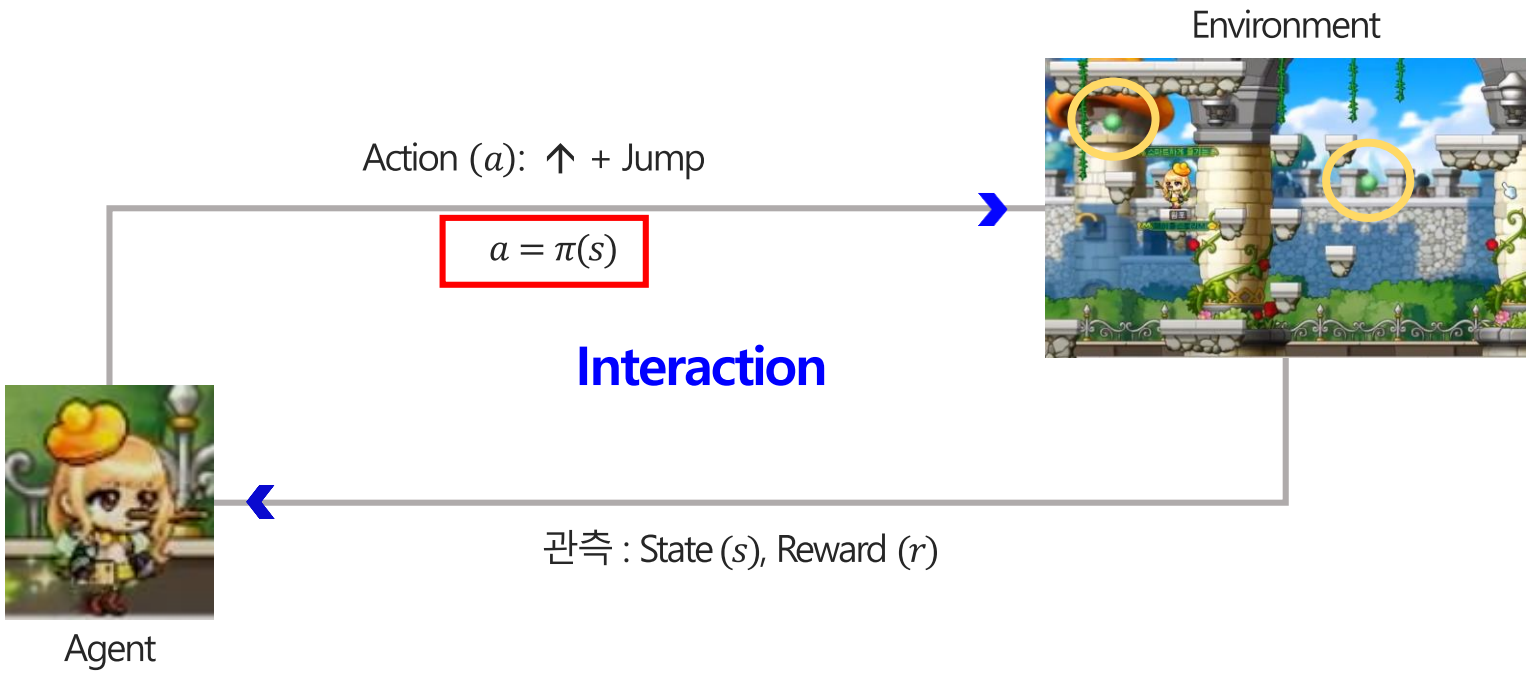
- 환경(Environment)과 상호작용(Interaction)하며 목표를 달성하는 에이전트(Agent)를 다루는 학습 방법



Reinforcement Learning

❖ The Goal of Reinforcement Learning

- 미래에 받을 보상의 합을 최대화하는 정책(Policy) $\boldsymbol{\pi(a|s)}$를 찾는 것
- $\boldsymbol{\pi(a|s)}$는 에이전트의 행동 함수로써 State (s)에서 취할 Action (a)을 알려줌



Details on Reinforcement Learning

❖ Reinforcement Learning vs. Other Learning

- 공통점: Action (a)이나 레이블(y), 사전에 정의한 Task와 같은 것을 예측
- 차이점: **상호작용(Interaction)하는 환경(Environment)**의 존재 여부

	Reinforcement Learning	Supervised Learning	Contrastive Learning
Training Data	$S, A, R, S, A, R, S, A \ldots$	(X, Y)	(X)
Model	$\pi(a\|s)$	$\hat{\mathrm{Y}} = F(\mathrm{X})$	$Self\left(\hat{\mathrm{Y}} = F(\mathrm{X})\right)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$	$\left(\mathrm{Y} - \hat{\mathrm{Y}}\right)^2$	$\mathcal{L}_i = -log \frac{exp\left(\frac{i \cdot i^+}{\tau}\right)}{exp\left(\frac{i \cdot i^+}{\tau}\right) + \sum_{k \notin \{i, i^+\}} exp\left(\frac{i \cdot k}{\tau}\right)}$

Details on Reinforcement Learning

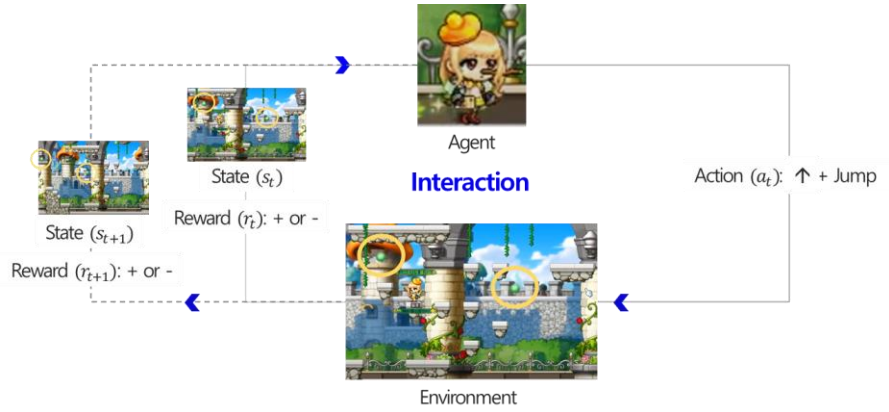
	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Training Data

- Markov Decision Process (MDP)
  - ✓ 순차적인 의사결정 문제를 풀기 위한 프로세스
  - ✓ 강화학습에서 사용하는 모든 환경(Environment)은 MDP로 구성되어 있음
  - ✓ Markov Property: 미래는 과거 상태가 아닌 오직 현재 상태에만 의존

$$(\mathbf{P}[S_{t+1}|S_t] = \mathbf{P}[S_{t+1}|S_1, S_2, \ldots, S_t])$$

- 5가지 원소 $< S, A, P, R, \gamma >$
  - ✓ S –State의 집합
  - ✓ A – Action의 집합
  - ✓ P – 전이 확률 함수
  - ✓ R – Reward 함수
  - ✓ γ – Discount Factor (할인율)



- **Episode: $< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots, s_T >$, total time step: T**

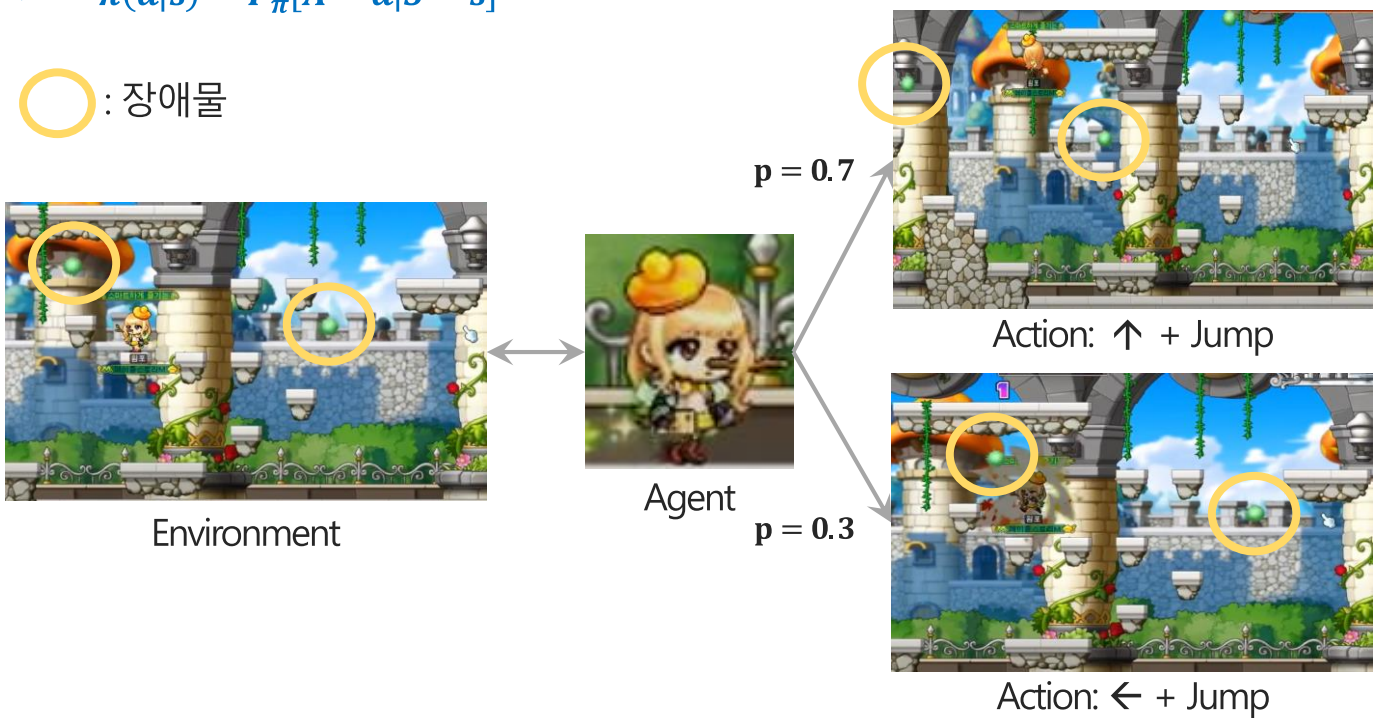
Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Model $\pi(a|s)$

- MDP로 구성된 환경(Environment)의 설명자 역할
- 정책(Policy)이라고 부르며 에이전트(Agent)의 행동 함수
- 주어진 환경의 State (s)에서 취할 Action (a)을 알려주는 역할

✓ $\boldsymbol{\pi(a|s) = P_\pi[A = a|S = s]}$



Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Bellman Equation

- Value Function의 반환값: $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

State-Value Function (상태 가치 함수)

✓ $v_\pi(s) = E_\pi[G_t|S_t = s]$

$= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |S_t = s]$

$= E_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)|S_t = s]$

$= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s]$

$= E_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$

State가 얼마나 좋은지 알려줌

Action-Value Function (행동 가치 함수)

✓ $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma q_\pi(s_{t+1}, a)|S_t = s, A_t = a]$

주어진 State에서 취한 Action이 얼마나 좋은지 알려줌

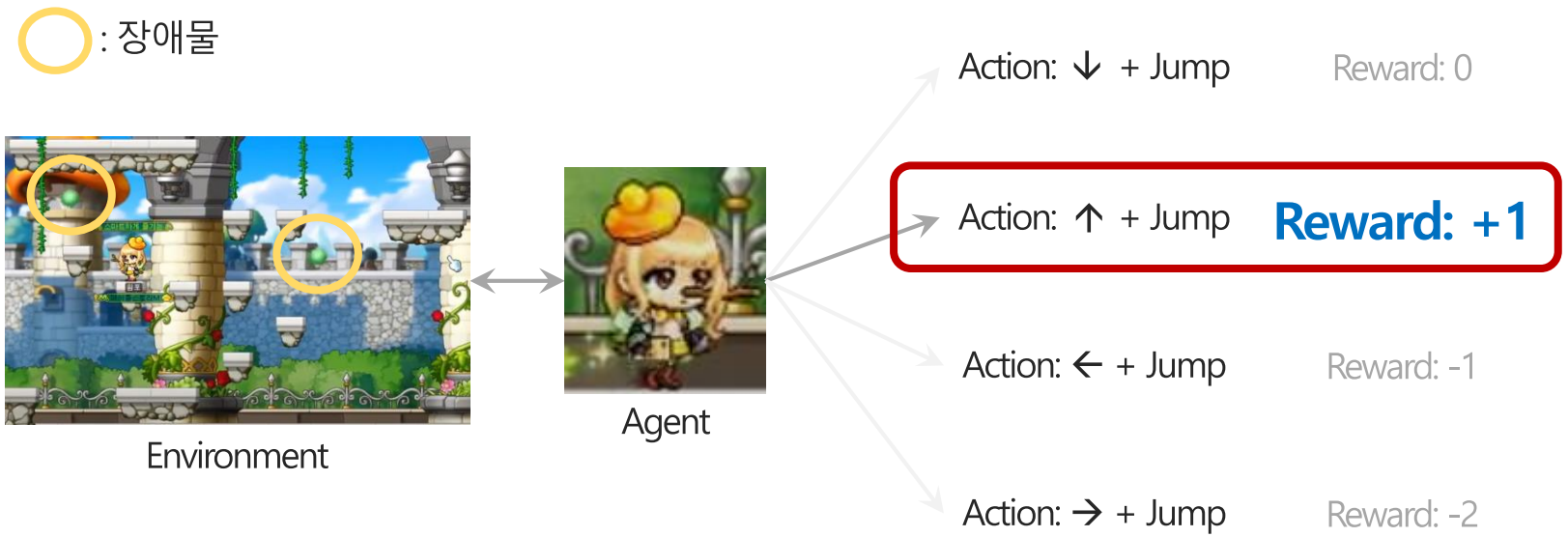
Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Q-Learning

- 가치(Value) 기반의 학습
- 에이전트가 특정 State (s)에서 자신이 취할 수 있는 Action (a)들 중 **이득이 되는 방향으로 학습**하는 방법

✓ $\mathbf{Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)]}$



Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Policy Gradient

- 정책(Policy) 기반의 학습
- 파라미터 θ에 대한 정책 $\pi(a|s;\theta)$를 직접 학습하는 방법

✓ $J(\theta) = E[\sum_{t=0}^{T-1} r_{t+1}|\boldsymbol{\pi_\theta}] = E[r_1 + r_2 + r_3 + \cdots + r_\tau|\boldsymbol{\pi_\theta}]$

✓ $\boldsymbol{\theta' = \theta + \alpha \nabla_\theta J(\theta)}$, $\boldsymbol{\nabla_\theta J(\theta)}$ = Policy Gradient

차이점

Action-Value Function (행동 가치 함수)

$$\nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = E_\tau \nabla_\theta [\sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) r_{t+1}]$$

$$\approx E_\tau[\nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) G_t]$$

$$where\ G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

✓ $q_\pi(s,a) = E_\pi[G_t|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \cdots)|S_t = s, A_t = a]$

$= E_\pi[R_{t+1} + \gamma G_{t+1}|S_t = s, A_t = a]$

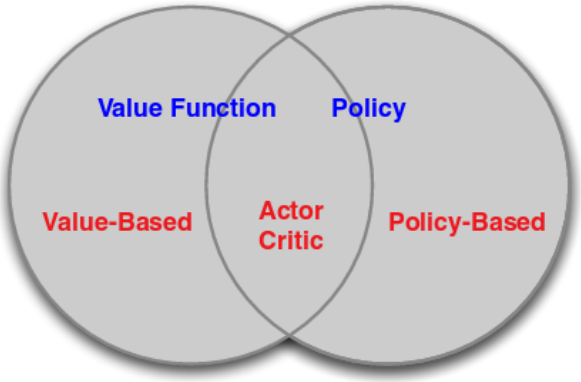
$= E_\pi[R_{t+1} + \gamma q_\pi(s_{t+1}, a)|S_t = s, A_t = a]$

Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ Actor-Critic Methods

- 가치(Value)와 정책(Policy) 모두 고려한 학습



Policy Gradient

$$\nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = E_\tau \nabla_\theta [\sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) r_{t+1}]$$

$$\approx E_\tau[\nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) G_t]$$

$$where\ G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

Action-Value Function (행동 가치 함수)

$$E_\tau[\nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) G_t]$$

$$\approx \sum_{t=0}^{T-1} E_{s_o,a_0,\ldots,s_t,a_t}[\nabla_\theta log\pi_\theta(a_t|s_t)] E_{r_{t+1},s_{t+1},\ldots,s_T,r_T}[G_t]$$

$$= \sum_{t=0}^{T-1} E_{s_o,a_0,\ldots,s_t,a_t}[\nabla_\theta log\pi_\theta(a_t|s_t)] Q_{\pi_\theta}(s_t,a_t)$$

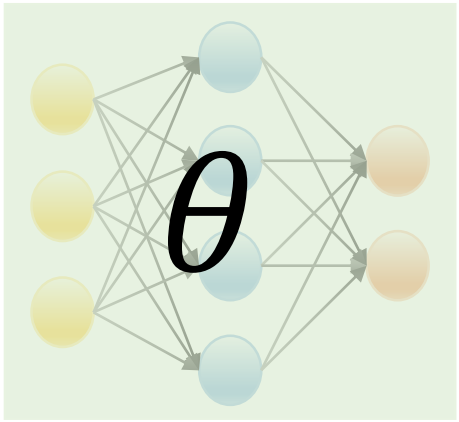
Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

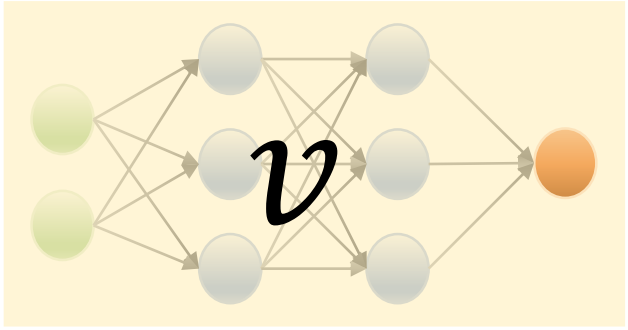
❖ Actor-Critic Methods

- 가치(Value)와 정책(Policy) 모두 고려한 학습

Actor



Critic



$$\theta \leftarrow \theta + \nabla_\theta log\pi_\theta(a_t|s_t)\left(r_{t+1} + \gamma Q_v(s_{t+1}, a_{t+1}) - Q_v(s_t, a_t)\right)$$

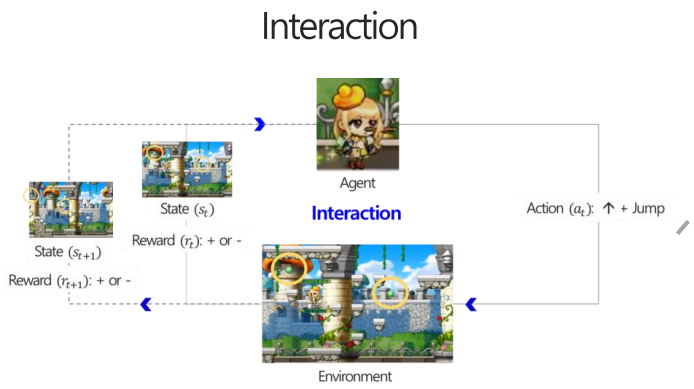
score

- ✓ Actor에서 Action (a)을 선택
- ✓ Critic은 선택된 Action (a)을 평가하는 역할
- ✓ Critic에서 파라미터 v를 업데이트
- ✓ Critic이 제안하는 방향으로 θ를 업데이트

Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

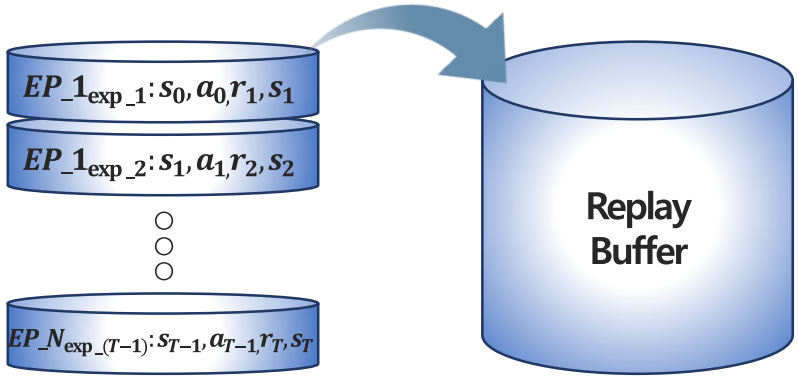
- ❖ Data Efficiency (=Sample Efficiency)
  - Training Data (Remind)
    - ✓ **Episode: $< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots, s_T >$, total time step: T**
  - Experience Replay (Replay Buffer, Replay Memory)
    - ✓ 각 Timestep별로 얻은 Experience $(s_t, a_t, r_{t+1}, s_{t+1})$들을 저장하는 공간



Episode_1: $< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots, s_T >$

Episode_2: $< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots, s_T >$

Episode_N: $< s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, s_3, \ldots, s_T >$

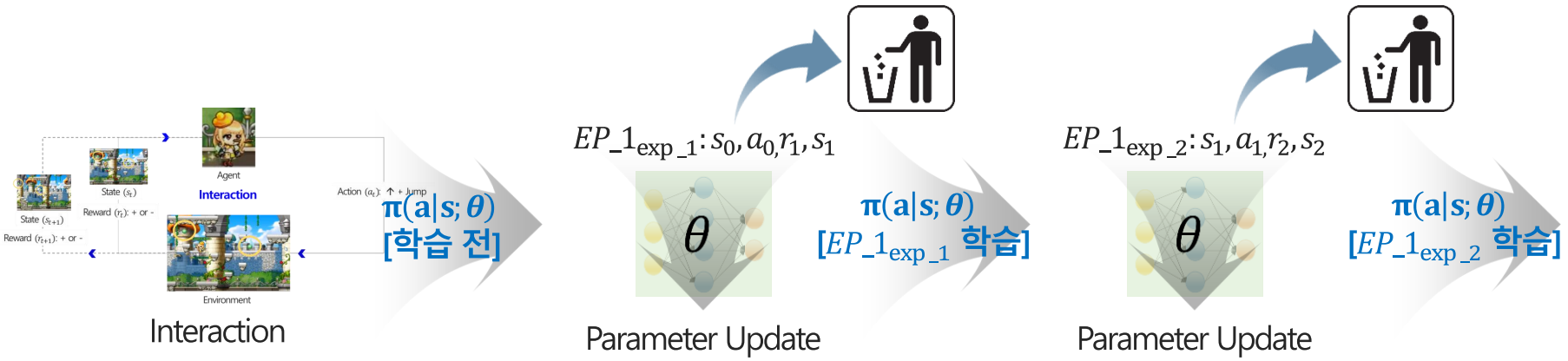


Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ **On-Policy** vs. Off-Policy

- 정책(Policy)이 Experience를 얻음과 동시에 학습을 진행
  - ✓ Experience는 현재 정책(학습 전)으로부터 얻음
  - ✓ 단점 1: Experience를 가지고 정책을 업데이트하기 때문에 Sample 분포 자체가 현재 정책에 의존적(Sample Dependent)
  - ✓ 단점 2: 정책을 업데이트한 후, 이전 Experience는 업데이트 된 정책과 다르기 때문에 사용 불가능
  - ✓ 단점 3: 정책이 발산하거나 Local Optimal에 수렴할 가능성이 커짐

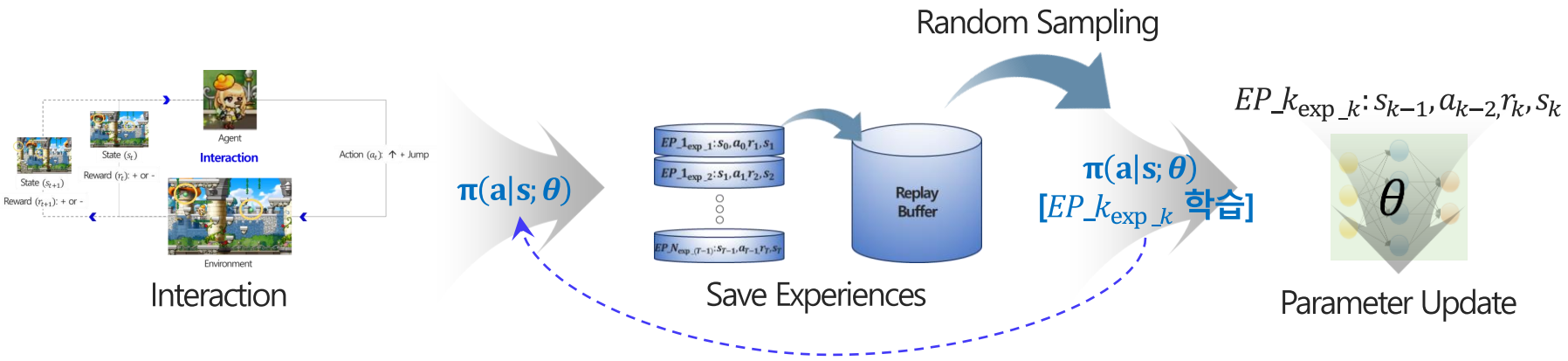


Details on Reinforcement Learning

	Reinforcement Learning
Training Data	$S,A,R,S,A,R,S,A\ldots$
Model	$\pi(a\|s)$
Objective	$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$

❖ On-Policy vs. **Off-Policy**

- Behavior 정책(Policy)과 Target 정책이 분리되어 진행
  - ✓ Behavior 정책은 State (s_t)에서 Action (a_t)를 취한 후 State ($\hat{s}_{t+1}$)를 예측하는 역할(Continue)
  - ✓ Target 정책은 예측한 State ($\hat{s}_{t+1}$)에서 Action ($\hat{a}_{t+1}$)를 예측하는 역할(Fixed Time)
  - ✓ 현재 학습하는 정책이 과거에 경험했던 Experience로도 학습에 사용 가능(**Experience Replay**)
  - ✓ Sample들을 사용하고 버리는 것과 달리 Random Sampling하여 Sample의 효율성을 높임
  - ✓ 정책이 Local Optimal에 수렴하거나 발산하는 경우를 방지



Combination of Reinforcement Learning and Representation Learning

- The Growth of Reinforcement Learning
  - 강화학습은 복잡하고 전략적인 게임 환경과 같은 순차적 의사결정 문제에서 지능형 에이전트를 구축하는데 큰 성공을 거둠
    - ✓ Ex: StarCraft I, StarCraft II, Dota II, …

StarCraft I

StarCraft II

Dota II

Combination of Reinforcement Learning and Representation Learning

- The Growth of Reinforcement Learning
  - Atari 2600, Dota II 게임 환경을 사용한 연구
  - Atari 2600: MuZero, Agent-57은 10~50년의 Experience를 사용
  - Dota II: OpenAI Five는 45,000년의 Experience를 사용

Atari 2600

Dota II

Agent57: Outperforming the Atari Human Benchmark

Adrià Puigdomènech Badia [*1] Bilal Piot [*1] Steven Kapturowski [*1] Pablo Sprechmann [*1] Alex Vitvitskyi [1]
Daniel Guo [1] Charles Blundell [1]

Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model

Julian Schrittwieser,[1*] Ioannis Antonoglou,[1,2*] Thomas Hubert,[1*]
Karen Simonyan,[1] Laurent Sifre,[1] Simon Schmitt,[1] Arthur Guez,[1]
Edward Lockhart,[1] Demis Hassabis,[1] Thore Graepel,[1,2] Timothy Lillicrap,[1]
David Silver[1,2*]

[1]DeepMind, 6 Pancras Square, London N1C 4AG.
[2]University College London, Gower Street, London WC1E 6BT.
*These authors contributed equally to this work.

Abstract

Constructing agents with planning capabilities has long been one of the main challenges in the pursuit of artificial intelligence. Tree-based planning methods have enjoyed huge success in challenging domains, such as chess and Go, where a perfect simulator is available. However, in real-world problems the dynamics governing the environment are often complex and unknown. In this work we present the *MuZero* algorithm which, by combining a tree-based search with a learned model, achieves superhuman performance in a range of challenging and visually complex domains, without any knowledge of their underlying dynamics. *MuZero* learns a model that, when applied iteratively, predicts the quantities most directly relevant to planning: the reward, the action-selection policy, and the value function. When evaluated on 57 different Atari games - the canonical video game environment for testing AI techniques, in which model-based planning approaches have historically struggled - our new algorithm achieved a new state of the art. When evaluated on Go, chess and shogi, without any knowledge of the game rules, *MuZero* matched the superhuman performance of the *AlphaZero* algorithm that was supplied with the game rules.

Dota 2 with Large Scale Deep Reinforcement Learning

OpenAI, *
Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung,
Przemysław "Psyho" Dębiak, Christy Dennison, David Farhi, Quirin Fischer,
Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson,
Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman,
Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang,
Filip Wolski, Susan Zhang

March 10, 2021

Abstract

On April 13th, 2019, OpenAI Five became the first AI system to defeat the world champions at an esports game. The game of Dota 2 presents novel challenges for AI systems such as long time horizons, imperfect information, and complex, continuous state-action spaces, all challenges which will become increasingly central to more capable AI systems. OpenAI Five leveraged existing reinforcement learning techniques, scaled to learn from batches of approximately 2 million frames every 2 seconds. We developed a distributed training system and tools for continual training which allowed us to train OpenAI Five for 10 months. By defeating the Dota 2 world champion (Team OG), OpenAI Five demonstrates that self-play reinforcement learning can achieve superhuman performance on a difficult task.

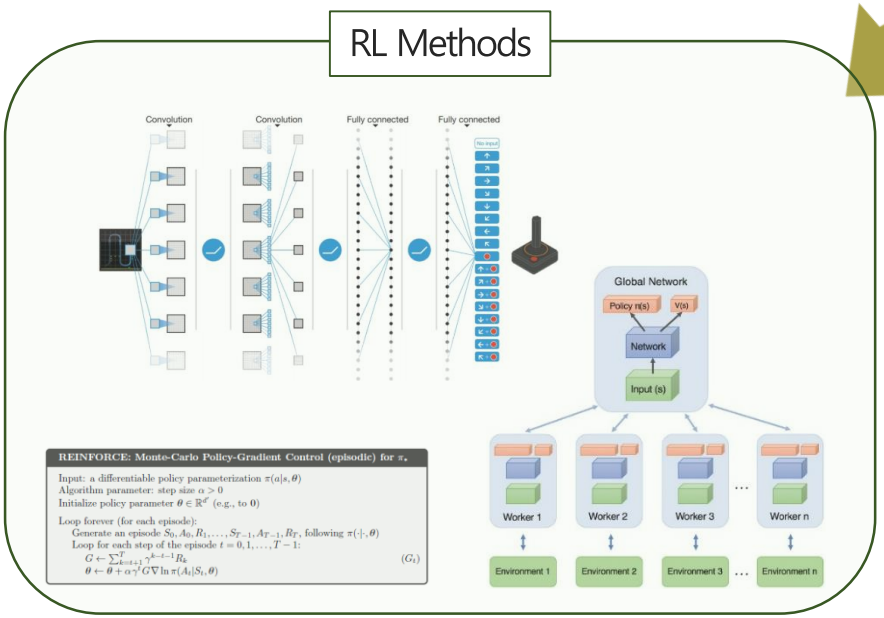
Combination of Reinforcement Learning and Representation Learning

❖ Is it practical in real-world?

- **실생활(Real-world) 관점에서 대부분의 환경은 모든 정보를 알 수 없음(Model-Free)**
- 환경(Environment)과 상호작용(Interaction)하며 10~50년, 45,000년의 Experience를 수집하는데 **수많은 비용 발생**
- 실생활 환경에서 수십, 수만 년의 Experience를 얻는 것은 거의 불가능
- 에이전트 성장에는 환경과 상호작용하며 수많은 시행 착오(Trial and Error)와 수많은 Sample들이 필요
- 학습 소요 시간이 길고 Sample의 효율성이 떨어짐(**Sample-inefficiency**)

Combination of Reinforcement Learning and Representation Learning

- How to solve the problem?
  - 훗날 실생활(Real-world)에 적용하기 위해서는 **학습 소요 시간을 줄이고 Sample의 효율성을 높이는 것**이 중요
  - Solution: Unlabeled Data를 사용하여 Pretext Task, Contrastive Learning으로 데이터 자체에 대한 좋은 Representation을 학습하는 방법인 Self-Supervised Learning을 결합



Combination of Reinforcement Learning and Representation Learning

- CURL: Contrastive Unsupervised Representations for Reinforcement Learning
  - Berkeley, BAIR / 2020 37th International Conference on Machine Learning (ICML)
  - 2021.04.15 기준으로 72회 인용

CURL: Contrastive Unsupervised Representations for Reinforcement Learning

Aravind Srinivas* [1] **Michael Laskin*** [1] **Pieter Abbeel** [1]

Abstract

We present CURL: Contrastive Unsupervised Representations for Reinforcement Learning. CURL extracts high-level features from raw pixels using contrastive learning and performs off-policy control on top of the extracted features. CURL outperforms prior pixel-based methods, both model-based and model-free, on complex tasks in the DeepMind Control Suite and Atari Games showing 1.9x and 1.2x performance gains at the 100K environment and interaction steps benchmarks respectively. On the DeepMind Control Suite, CURL is the first image-based algorithm to nearly match the sample-efficiency of methods that use state-based features. Our code is open-sourced and available at https://www.github.com/MishaLaskin/curl.

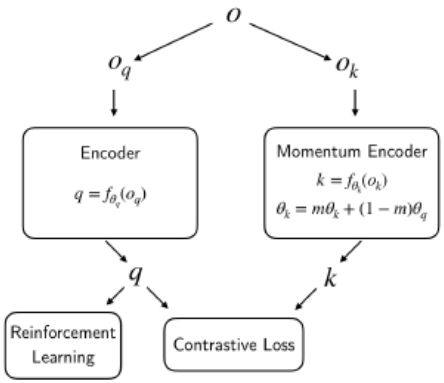
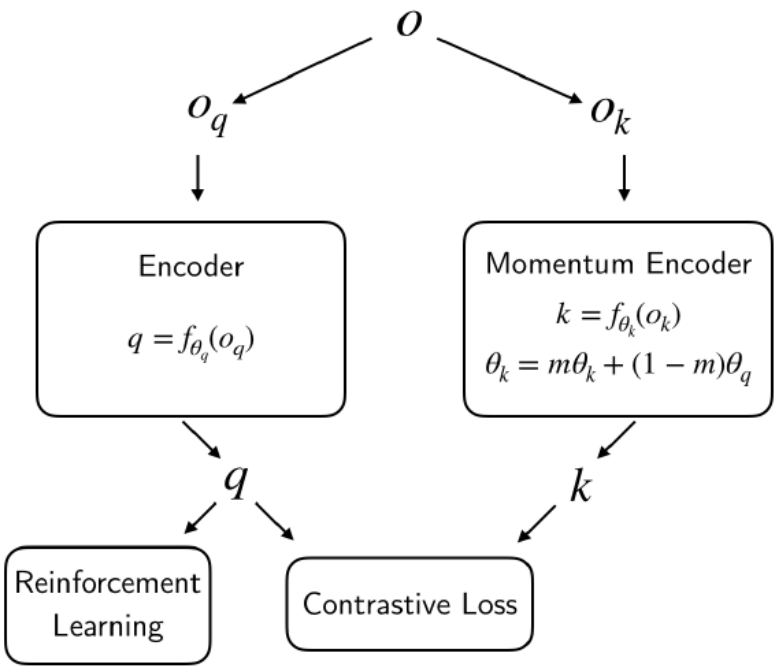


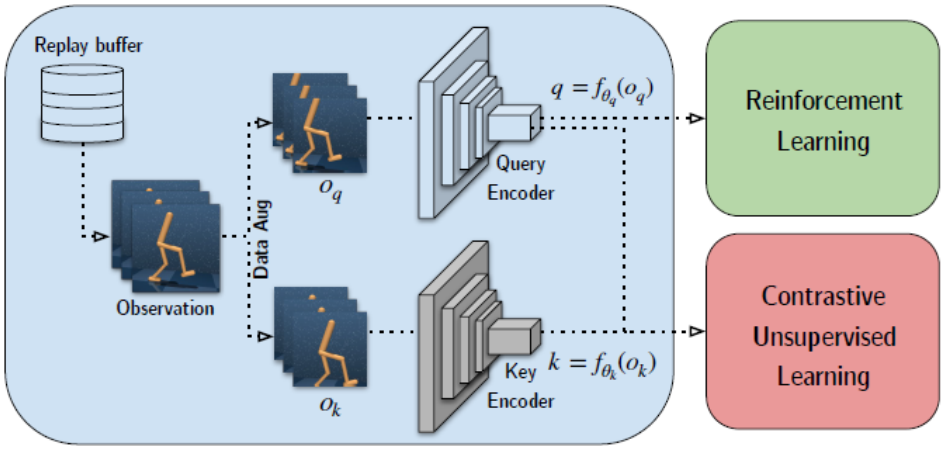
Figure 1. Contrastive Unsupervised Representations for Reinforcement Learning (CURL) combines instance contrastive learning and reinforcement learning. CURL trains a visual

Combination of Reinforcement Learning and Representation Learning

❖ CURL: Contrastive Unsupervised Representations for Reinforcement Learning

- RL Method: 가치(Value) 기반의 학습 방식인 Efficient Rainbow 방법론 사용 (Model-Free, Off-Policy)
- Self-Supervised Learning: MoCo 사용(Contrastive Learning)

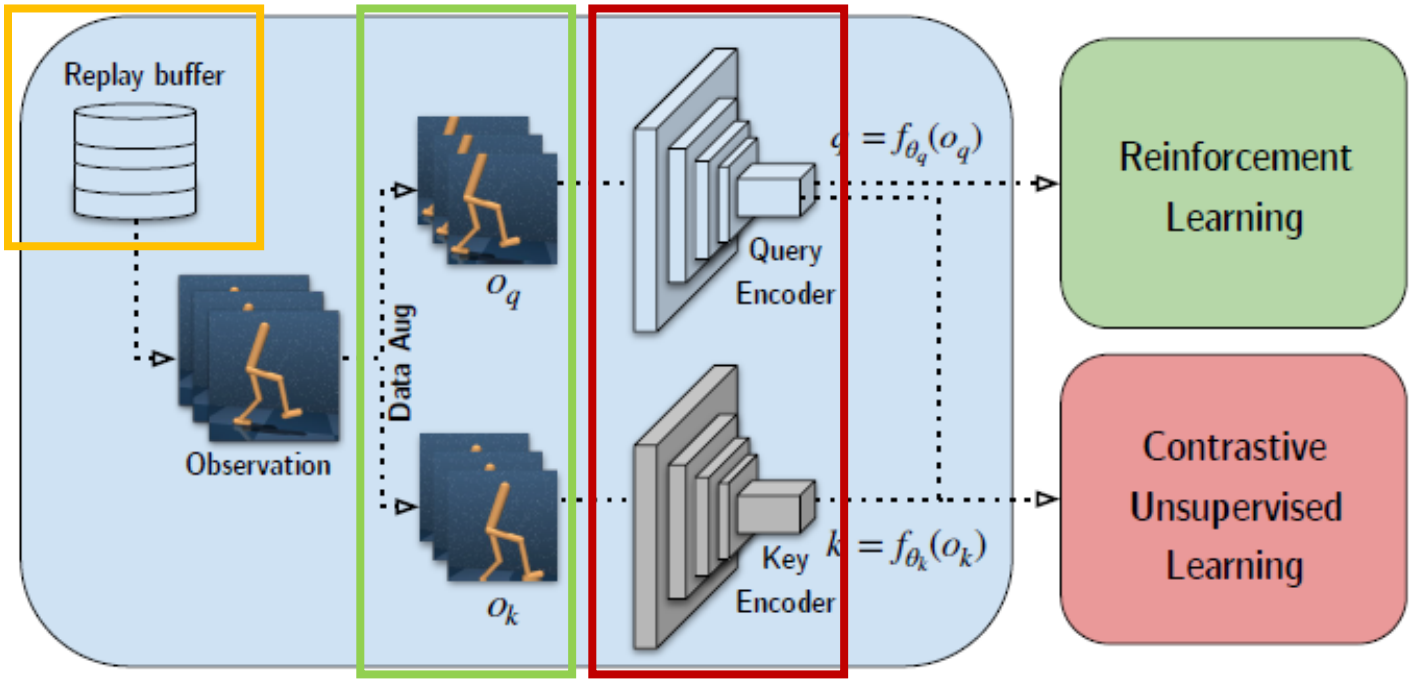




Combination of Reinforcement Learning and Representation Learning

❖ CURL: Contrastive Unsupervised Representations for Reinforcement Learning

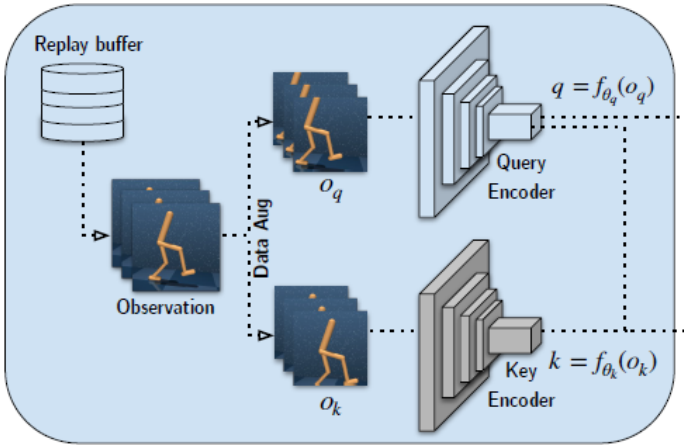
- Query Encoder와 Key Encoder로 네트워크 구성
- 각 Timestep별 Experience를 Replay Buffer에 저장
- Mini Batch 사이즈만큼 샘플링하고 서로 다른 두개의 Data Augmentation을 적용



Combination of Reinforcement Learning and Representation Learning

❖ CURL: Contrastive Unsupervised Representations for Reinforcement Learning

- 동일 이미지로부터 나온 두 이미지는 **Positive Pair**, 이외에는 **Negative Examples**로 정의
- Query Encoder에서 추출한 Feature Vector는 RL과 SSL에 사용
- RL로부터 계산된 Loss와 SSL로부터 계산된 Loss에 λ를 곱한 Loss를 더하여 학습 수행
- RL과 SSL을 동시에 수행하는 **One-Stage 방식**(Not Transfer Learning)



RL Loss Function

$$\lambda * \mathcal{L}_i = -log \frac{exp\left(\frac{i \cdot i^+}{\tau}\right)}{exp\left(\frac{i \cdot i^+}{\tau}\right) + \sum_{k \notin \{i, i^+\}} exp\left(\frac{i \cdot k}{\tau}\right)}$$

Momentum Update (Key Encoder)

$$\theta_{key} \leftarrow m\theta_{key} + (1-m)\theta_{query}$$

Combination of Reinforcement Learning and Representation Learning

- CURL: Contrastive Unsupervised Representations for Reinforcement Learning
  - 26개의 Atari 2600 게임을 사용하여 성능 평가
  - 10만 Timestep으로 한정: 학습 소요 시간이 적음, Sample-Efficiency 향상 대변
  - 비교 방법론보다 26개의 게임 중 7개 勝

GAME	HUMAN	RANDOM	RAINBOW	SIMPLE	OTRAINBOW	EFF. RAINBOW	CURL
ALIEN	7127.7	227.8	318.7	616.9	**824.7**	739.9	558.2
AMIDAR	1719.5	5.8	32.5	88.0	82.8	**188.6**	142.1
ASSAULT	742.0	222.4	231	527.2	351.9	431.2	**600.6**
ASTERIX	8503.3	210.0	243.6	**1128.3**	628.5	470.8	734.5
BANK HEIST	753.1	14.2	15.55	34.2	**182.1**	51.0	131.6
BATTLE ZONE	37187.5	2360.0	2360.0	5184.4	4060.6	10124.6	**14870.0**
BOXING	12.1	0.1	-24.8	**9.1**	2.5	0.2	1.2
BREAKOUT	30.5	1.7	1.2	**16.4**	9.84	1.9	4.9
CHOPPER COMMAND	7387.8	811.0	120.0	**1246.9**	1033.33	861.8	1058.5
CRAZY_CLIMBER	35829.4	10780.5	2254.5	**62583.6**	21327.8	16185.3	12146.5
DEMON_ATTACK	1971.0	152.1	163.6	208.1	711.8	508.0	**817.6**
FREEWAY	29.6	0.0	0.0	20.3	25.0	**27.9**	26.7
FROSTBITE	4334.7	65.2	60.2	254.7	231.6	866.8	**1181.3**
GOPHER	2412.5	257.6	431.2	771.0	**778.0**	349.5	669.3
HERO	30826.4	1027.0	487	2656.6	6458.8	**6857.0**	6279.3
JAMESBOND	302.8	29.0	47.4	125.3	112.3	301.6	**471.0**
KANGAROO	3035.0	52.0	0.0	323.1	605.4	779.3	**872.5**
KRULL	2665.5	1598.0	1468	**4539.9**	3277.9	2851.5	4229.6
KUNG_FU_MASTER	22736.3	258.5	0.	**17257.2**	5722.2	14346.1	14307.8
MS_PACMAN	6951.6	307.3	67	**1480.0**	941.9	1204.1	1465.5
PONG	14.6	-20.7	-20.6	**12.8**	1.3	-19.3	-16.5
PRIVATE EYE	69571.3	24.9	0	58.3	100.0	97.8	**218.4**
QBERT	13455.0	163.9	123.46	**1288.8**	509.3	1152.9	1042.4
ROAD_RUNNER	7845.0	11.5	1588.46	5640.6	2696.7	**9600.0**	5661.0
SEAQUEST	42054.7	68.4	131.69	**683.3**	286.92	354.1	384.5
UP_N_DOWN	11693.2	533.4	504.6	**3350.3**	2847.6	2877.4	2955.2

Combination of Reinforcement Learning and Representation Learning

- Data-Efficient Reinforcement Learning with Self-Predictive Representations
  - Mila, Université de Montréal / 2020 Research gate
  - 2021.04.15 기준으로 2회 인용

Data-Efficient Reinforcement Learning with Self-Predictive Representations

Max Schwarzer* Mila, Université de Montréal

Ankesh Anand* Mila, Université de Montréal, Microsoft Research

Rishab Goel Mila

R Devon Hjelm Microsoft Research, Mila, Université de Montréal

Aaron Courville Mila, Université de Montréal, CIFAR Fellow

Philip Bachman Microsoft Research

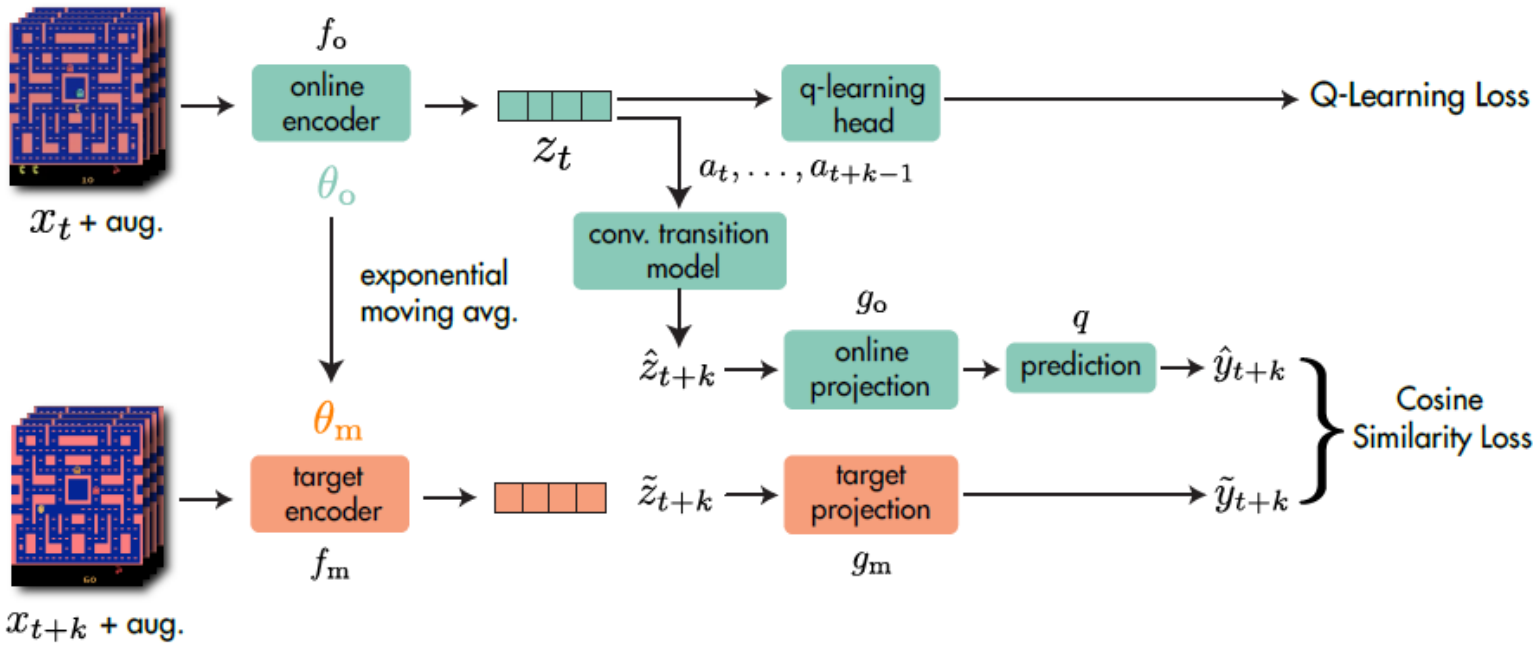
Abstract

While deep reinforcement learning excels at solving tasks where large amounts of data can be collected through virtually unlimited interaction with the environment, learning from limited interaction remains a key challenge. We posit that an agent can learn more efficiently if we augment reward maximization with self-supervised objectives based on structure in its visual input and sequential interaction with the environment. Our method, Self-Predictive Representations (SPR), trains an agent to predict its own latent state representations multiple steps into the future. We compute *target* representations for future states using an encoder which is an exponential moving average of the agent's parameters and we make predictions using a learned transition model. On its own, this future prediction objective outperforms prior methods for sample-efficient deep RL from pixels. We further improve performance by adding data augmentation to the future prediction loss, which forces the agent's representations to be consistent across multiple views of an observation. Our full self-supervised objective, which combines future prediction and data augmentation, achieves a median human-normalized score of 0.415 on Atari in a setting limited to 100k steps of environment interaction, which represents a 55% relative improvement over the previous state-of-the-art. Notably, even in this limited data regime, SPR exceeds expert human scores on 7 out of 26 games. The code associated with this work is available at https://github.com/mila-iqia/spr.

Combination of Reinforcement Learning and Representation Learning

❖ Data-Efficient Reinforcement Learning with Self-Predictive Representations

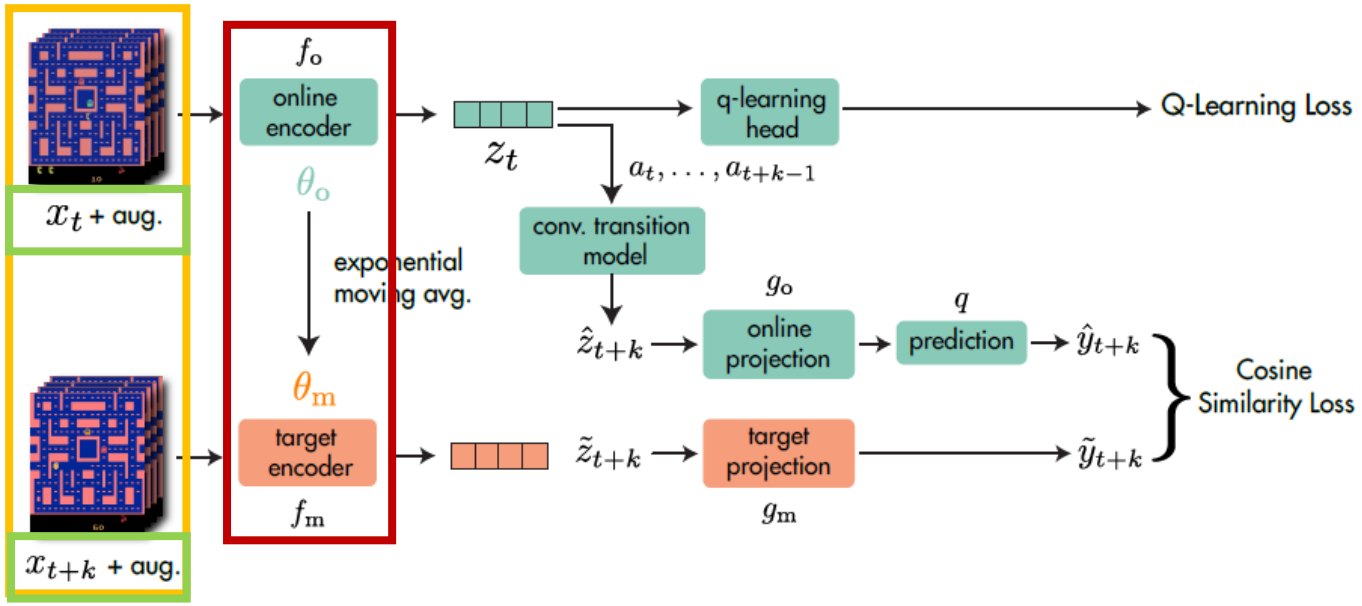
- RL Method: 가치(Value) 기반의 학습 방식인 Efficient Rainbow 방법론 사용 (Model-Free, Off-Policy)
- Self-Supervised Learning: BYOL 사용(Non-Contrastive Learning)



Combination of Reinforcement Learning and Representation Learning

❖ Data-Efficient Reinforcement Learning with Self-Predictive Representations

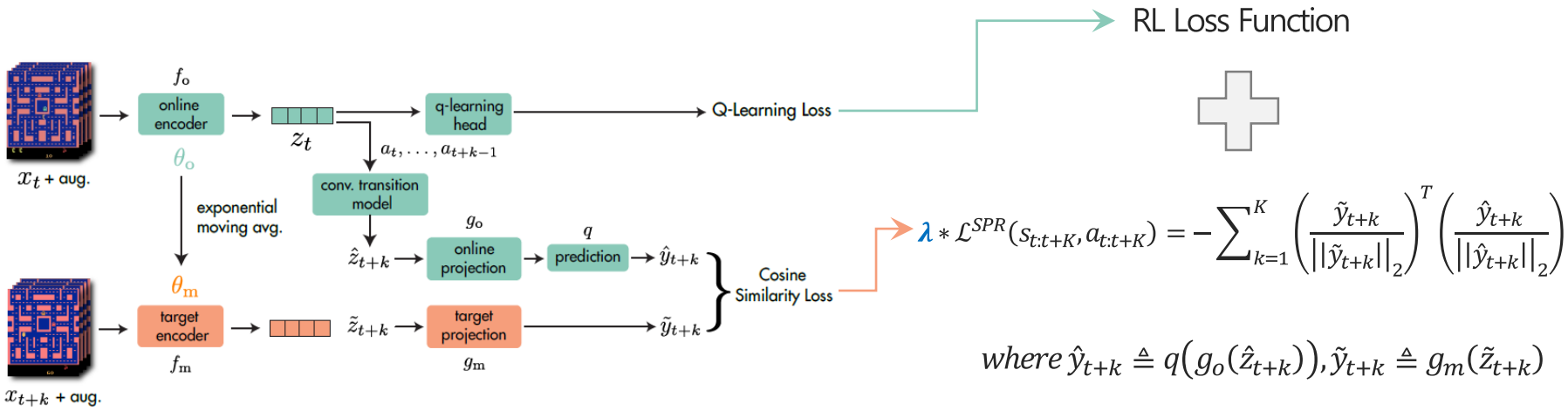
- Online Encoder와 Target Encoder로 네트워크 구성
- 각 Timestep별 Experience를 Replay Buffer에 저장 및 Mini Batch 사이즈만큼 샘플링
- 서로 다른 시점$(t, t+k)$의 이미지에 서로 다른 두개의 Data Augmentation 적용
- Data Augmentation으로부터 나온 두 이미지는 Positive Pair로 정의



Combination of Reinforcement Learning and Representation Learning

❖ Data-Efficient Reinforcement Learning with Self-Predictive Representations

- Online Encoder에서 추출한 Feature Vector (z_t)는 RL과 SSL에 사용
- **Self-Predictive Representations (SPR)** 기법을 제안하여 Sample-Efficiency를 더욱 향상
  - ✓ SPR은 transition Model(h) 추가, 미래의 요약된 State ($\hat{z}_{t+k}$) 예측 및 k 반복하며 SSL Loss 계산
  - ✓ $\hat{z}_{t+k} \triangleq h(\hat{z}_{t+k-1}, a_{t+k-1})$, a_t는 Mini Batch 내 존재하는 미래의 Action 정보
- RL로부터 계산된 Loss와 SSL로부터 계산된 Loss에 λ를 곱한 Loss를 더하여 학습 수행



Momentum Update (Target Encoder)

$$\delta_{target} \leftarrow \tau \delta_{target} + (1 - \tau)\theta_{online}$$

Combination of Reinforcement Learning and Representation Learning

❖ Data-Efficient Reinforcement Learning with Self-Predictive Representations

- 26개의 Atari 2600 게임을 사용하여 성능 평가
- 10만 Timestep으로 한정: 학습 소요 시간이 적음, Sample-Efficiency 향상 대변
- 인간 성능보다 26개의 게임 중 7개 勝(CURL은 2개 勝)

Game	Random	Human	SimPLe	DER	OTRainbow	CURL	DrQ	SPR (no Aug)	SPR
Alien	227.8	7127.7	616.9	739.9	824.7	558.2	771.2	**847.2**	801.5
Amidar	5.8	1719.5	88.0	**188.6**	82.8	142.1	102.8	142.7	176.3
Assault	222.4	742.0	527.2	431.2	351.9	600.6	452.4	**665.0**	571.0
Asterix	210.0	8503.3	**1128.3**	470.8	628.5	734.5	603.5	820.2	977.8
Bank Heist	14.2	753.1	34.2	51.0	182.1	131.6	168.9	**425.6**	380.9
BattleZone	2360.0	37187.5	5184.4	10124.6	4060.6	14870.0	12954.0	10738.0	**16651.0**
Boxing	0.1	12.1	9.1	0.2	2.5	1.2	6.0	12.7	**35.8**
Breakout	1.7	30.5	16.4	1.9	9.8	4.9	16.1	12.9	**17.1**
ChopperCommand	811.0	7387.8	**1246.9**	861.8	1033.3	1058.5	780.3	667.3	974.8
Crazy Climber	10780.5	35829.4	**62583.6**	16185.3	21327.8	12146.5	20516.5	43391.0	42923.6
Demon Attack	152.1	1971.0	208.1	508.0	711.8	817.6	**1113.4**	370.1	545.2
Freeway	0.0	29.6	20.3	**27.9**	25.0	26.7	9.8	16.1	24.4
Frostbite	65.2	4334.7	254.7	866.8	231.6	1181.3	331.1	1657.4	**1821.5**
Gopher	257.6	2412.5	771.0	349.5	**778.0**	669.3	636.3	774.5	715.2
Hero	1027.0	30826.4	2656.6	6857.0	6458.8	6279.3	3736.3	5707.4	**7019.2**
Jamesbond	29.0	302.8	125.3	301.6	112.3	**471.0**	236.0	367.2	365.4
Kangaroo	52.0	3035.0	323.1	779.3	605.4	872.5	940.6	1359.5	**3276.4**
Krull	1598.0	2665.5	**4539.9**	2851.5	3277.9	4229.6	4018.1	3123.1	3688.9
Kung Fu Master	258.5	22736.3	**17257.2**	14346.1	5722.2	14307.8	9111.0	15469.7	13192.7
Ms Pacman	307.3	6951.6	**1480.0**	1204.1	941.9	1465.5	960.5	1247.7	1313.2
Pong	-20.7	14.6	**12.8**	-19.3	1.3	-16.5	-8.5	-16.0	-5.9
Private Eye	24.9	69571.3	58.3	97.8	100.0	**218.4**	-13.6	52.6	124.0
Qbert	163.9	13455.0	**1288.8**	1152.9	509.3	1042.4	854.4	606.6	669.1
Road Runner	11.5	7845.0	5640.6	9600.0	2696.7	5661.0	8895.1	10511.0	**14220.5**
Seaquest	68.4	42054.7	**683.3**	354.1	286.9	384.5	301.2	580.8	583.1
Up N Down	533.4	11693.2	3350.3	2877.4	2847.6	2955.2	3180.8	6604.6	**28138.5**
Mean Human-Norm'd	0.000	1.000	0.443	0.285	0.264	0.381	0.357	0.463	**0.704**
Median Human-Norm'd	0.000	1.000	0.144	0.161	0.204	0.175	0.268	0.307	**0.415**
Mean DQN@50M-Norm'd	0.000	23.382	0.232	0.239	0.197	0.325	0.171	0.336	**0.510**
Median DQN@50M-Norm'd	0.000	0.994	0.118	0.142	0.103	0.142	0.131	0.225	**0.361**
# Superhuman	0	N/A	2	2	1	2	2	5	7

Conclusion

❖ 결론

- 강화학습은 매우 복잡하고 전략적인 게임 환경에서 효과를 입증
- 실생활(Real-world)에 실용적이기 위해서는 학습 소요 시간을 줄이고 Sample-Efficiency를 향상시키는 것이 중요
- 최근 각광받고 있는 Self-Supervised Learning의 효율성을 확인하고 이를 강화학습과 결합하는 연구가 진행되고 있으며 문제를 해결
- 논리적 타당성을 가진다면 한 가지의 학습 뿐만 아니라 다른 학습 방식을 결합하여 해결하는 방안도 고려(Ex: 강화학습 + Meta Learning)

References

DMQA_YouTube_Seminar

문석호(Self-Supervised Learning): http://dmqa.korea.ac.kr/activity/seminar/302

곽민구(Towards Contrastive Learning): http://dmqa.korea.ac.kr/activity/seminar/308

김재훈(Dive into BYOL): http://dmqa.korea.ac.kr/activity/seminar/310

조억(Cooperative Multi-Agent Reinforcement Learning Framework for Scalping Trading): http://dmqa.korea.ac.kr/activity/seminar/277

MoCo: He, K., Fan, H., Wu, Y., Xie, S., & Girshick, R. (2020). Momentum contrast for unsupervised visual representation learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 9729-9738).

SimCLR: Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020, November). A simple framework for contrastive learning of visual representations. In International conference on machine learning (pp. 1597-1607). PMLR.

BYOL: Grill, J. B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., ... & Valko, M. (2020). Bootstrap your own latent: A new approach to self-supervised learning. arXiv preprint arXiv:2006.07733.

Dota II: Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., ... & Zhang, S. (2019). Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680.

MuZero: Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., ... & Silver, D. (2020). Mastering atari, go, chess and shogi by planning with a learned model. Nature, 588(7839), 604-609.

Agent57: 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.

Rainbow: Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., ... & Silver, D. (2018, April). Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

Efficient Rainbow: van Hasselt, H., Hessel, M., & Aslanides, J. (2019). When to use parametric models in reinforcement learning?. arXiv preprint arXiv:1906.05243.

CURL: Laskin, M., Srinivas, A., & Abbeel, P. (2020, November). Curl: Contrastive unsupervised representations for reinforcement learning. In International Conference on Machine Learning (pp. 5639-5650). PMLR..

SPR: Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., & Bachman, P. Data-Efficient Reinforcement Learning with Self-Predictive Representations.

감사합니다

Appendix

- Policy Gradient

❖ 수식 유도

- $J(\theta) = E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = E[r_1 + r_2 + r_3 + \cdots + r_\tau|\pi_\theta]$
- $\theta' = \theta + \alpha\nabla_\theta J(\theta)$, $\nabla_\theta J(\theta)$ = Policy Gradient

$$\nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = \nabla_\theta \sum_{t=0}^{T-1} P(s_t, a_t|\tau) r_{t+1}$$

$$= \sum_{t=0}^{T-1} \nabla_\theta P(s_t, a_t|\tau) r_{t+1}$$

미분 불가능

Appendix

- Policy Gradient

❖ 수식 유도

- $J(\theta) = E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = E[r_1 + r_2 + r_3 + \cdots + r_\tau|\pi_\theta]$
- $\theta' = \theta + \alpha\nabla_\theta J(\theta)$, $\nabla_\theta J(\theta)$ = Policy Gradient

$$\nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = \nabla_\theta \sum_{t=0}^{T-1} P(s_t, a_t|\tau) r_{t+1}$$

$$= \sum_{t=0}^{T-1} \nabla_\theta P(s_t, a_t|\tau) r_{t+1}$$

미분 가능

$$= \sum_{t=0}^{T-1} P(s_t, a_t|\tau) \frac{\nabla_\theta P(s_t, a_t|\tau)}{P(s_t, a_t|\tau)} r_{t+1}$$

$$= \sum_{t=0}^{T-1} P(s_t, a_t|\tau) \nabla_\theta log P(s_t, a_t|\tau) r_{t+1}$$

$$= E_\tau[\sum_{t=0}^{T-1} \nabla_\theta log P(s_t, a_t|\tau) r_{t+1}]$$

Appendix

- Policy Gradient

❖ 수식 유도

- $P(s_t, a_t|\tau) = P(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t|\theta)$
- $P(s_0)\pi_\theta(a_0|s_0)P(s_1|s_0, a_0)\pi_\theta(a_1|s_1)P(s_2|s_1, a_1)\ \ldots$
- $logAB = logA + logB$

$$P(\tau|\theta) = P(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)P(s_{t+1}|s_t, a_t)$$

$$logP(\tau|\theta) = logP(s_0) + \sum_{t=0}^{T-1}[log\pi_\theta(a_t|s_t) + logP(s_{t+1}|s_t, a_t)]$$

θ에 대한 미분 → 상태 전이 확률 미분 시 상수 제거

Appendix

- Policy Gradient

❖ 수식 유도

- $P(s_t, a_t|\tau) = P(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t|\theta)$
- $P(s_0)\pi_\theta(a_0|s_0)P(s_1|s_0, a_0)\pi_\theta(a_1|s_1)P(s_2|s_1, a_1) \ldots$
- $logAB = logA + logB$

$$P(\tau|\theta) = P(s_0)\prod_{t=0}^{T-1}\pi_\theta(a_t|s_t)P(s_{t+1}|s_t, a_t)$$

$$logP(\tau|\theta) = logP(s_0) + \sum_{t=0}^{T-1}[log\pi_\theta(a_t|s_t) + logP(s_{t+1}|s_t, a_t)]$$

$$\nabla_\theta logP(\tau|\theta) = \nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t)$$

Appendix

- Policy Gradient

❖ 수식 유도

- $P(s_t, a_t|\tau) = P(s_0, a_0, r_1, s_1, a_1, r_2, \ldots, s_t, a_t|\theta)$
- $P(s_0)\pi_\theta(a_0|s_0)P(s_1|s_0, a_0)\pi_\theta(a_1|s_1)P(s_2|s_1, a_1)\ldots$
- $logAB = logA + logB$

$$\nabla_\theta logP(\tau|\theta) = \nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) \xrightarrow{\text{대입}} E_\tau[\sum_{t=0}^{T-1} \nabla_\theta logP(s_t, a_t|\tau) r_{t+1}]$$

$$\nabla_\theta E[\sum_{t=0}^{T-1} r_{t+1}|\pi_\theta] = E_\tau \nabla_\theta [\sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) r_{t+1}]$$

$$\approx E_\tau[\nabla_\theta \sum_{t=0}^{T-1} log\pi_\theta(a_t|s_t) G_t],\ where\ G_t = \sum_{t=0}^{T-1} \gamma^t r_{t+1} = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots + \gamma^{T-1} r_T$$

Discounted G_t → 단순 보상의 합 발산 방지