2022 동계 세미나

- Deep Reinforcement Learning 이론 및 동향-

김준규

Vision & Display Systems Lab.

Dept. of Electronic Engineering, Sogang University

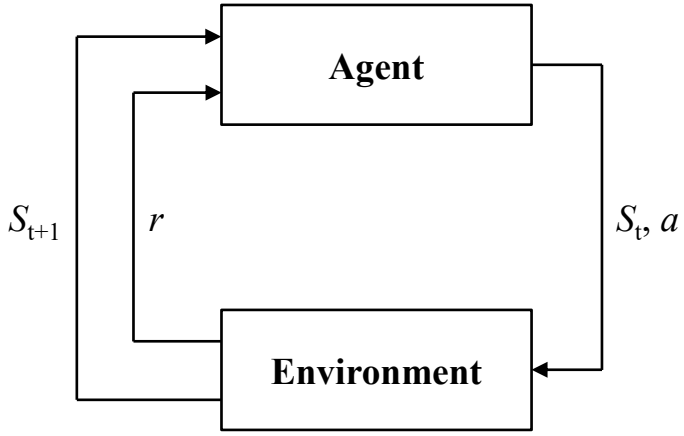
Outline

- Introduction
  - Reinforcement learning
  - Deep Q Network
    - Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." NIPS Deep Learning Workshop, 2013.
    - Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." nature 518.7540 : 529-533, 2015.

- Application
  - HDR + RL 논문 소개
    - Z. Wang et al. “Learing a Reinforced Agent for Flexible Exposure Bracketing Selection, ” CVPR, 2020.

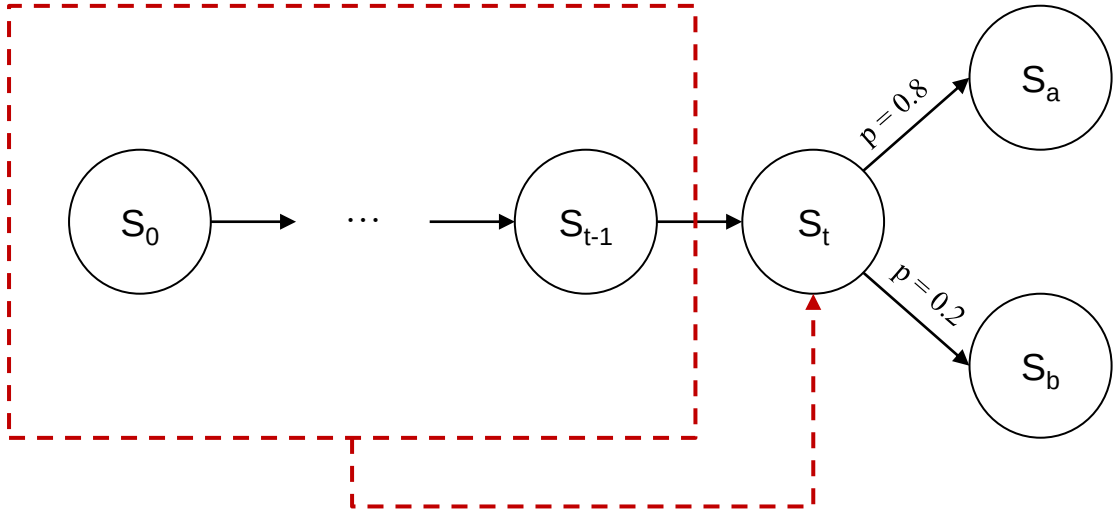
Introduction

- Reinforcement Learning
  - Environment에서 정의된 state에서 취한 action에 따라 state transition
  - 도착한 state에 따라서 reward가 주어짐
    - 조건보다 좋은 결과 : 양의 reward
    - 조건보다 안좋은 결과 : 음의 reward
  - Reward를 최대화하는 action을 취하도록 agent를 학습



Markov Decision Process

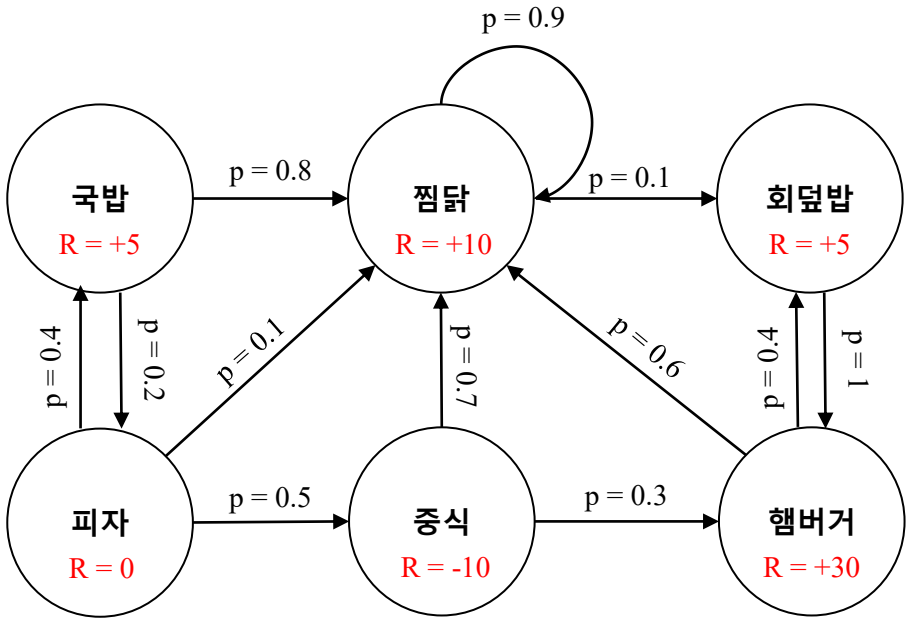
- Markov property
  - State S_t에서 state S_{t+1}로 transition하는 것은 이전에 지나온 state들과는 무관함
    - S_0부터 S_{t-1}까지의 정보는 이미 S_t에 포함되었다는 가정이 전제됨
    - $P(S_{t+1} = S_a \mid S_0, S_1, \ldots, S_{t-1}, S_t) = P(S_{t+1} = S_a \mid S_t)$



< Example of Markov Chain >

Markov Decision Process

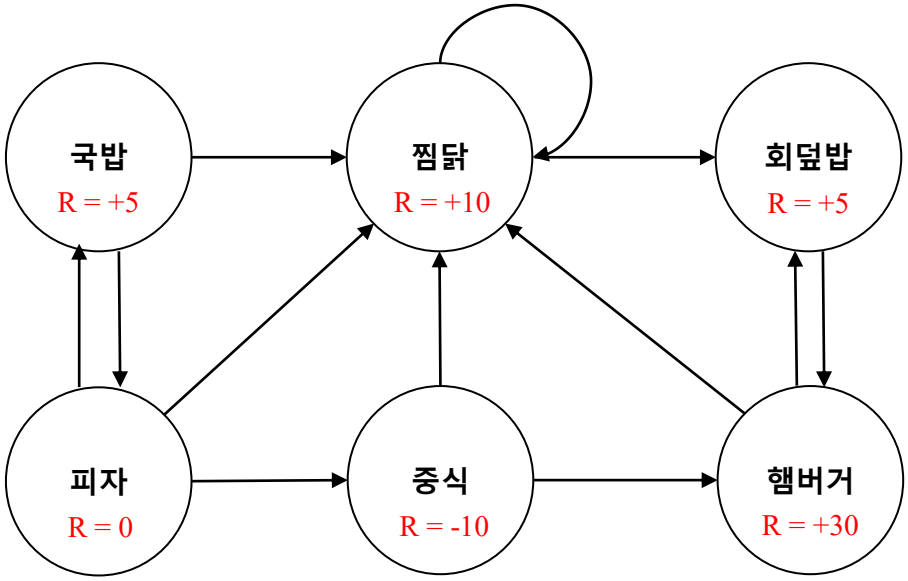
- Markov reward process
  - Markov process에 reward의 개념을 추가
    - State의 가치를 표현하는 reward를 각 state에 부여함
    - Transition은 정해진 확률에 따라서 수행됨



< TE관 배달음식 선정 Markov Reward Process >

Markov Decision Process

- Markov decision process
  - Markov reward process에서 transition을 action을 결정하는 policy에 따라 수행
    - Policy : $\pi(a|s) = P[A_t = a \mid S_t = s]$
      - Policy π에 의해 state s에서 action a를 수행하도록 결정
    - 여러 Iteration을 통해 policy를 개선시켜 더 좋은 return을 얻음



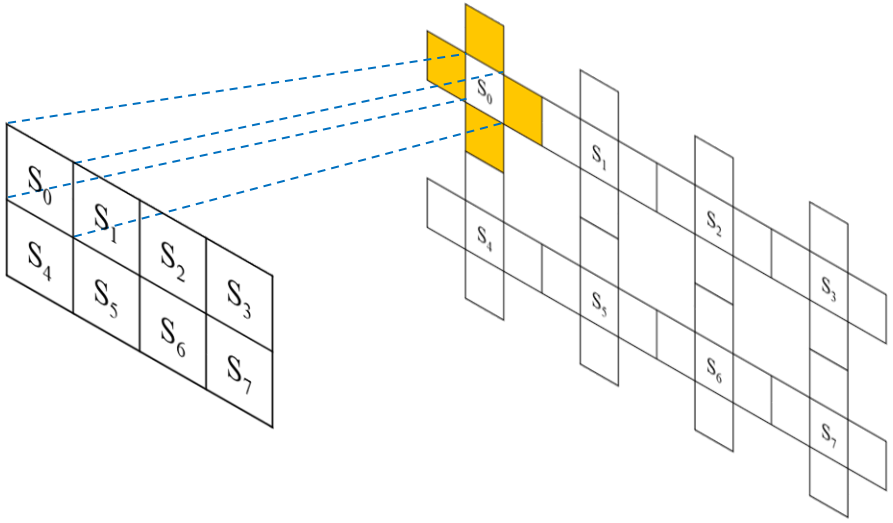
< TE관 배달음식 선정 Markov Decision Process >

Value Function

- 용어 정리
  - Episode
    - Initial state로부터 terminal state까지의 state sequence
      - ex) 체스 한 판
    - Episode는 여러 개의 (time) step으로 구성
      - 각 episode의 step 수를 horizon이라고 함
  - Return
    - $G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$
      - Time step t 로부터 horizon까지의 reward의 discounted sum
        - ✓Discount factor(γ)로 미래의 reward와 immediate reward의 관계성 정의
        - ✓미래의 reward를 현재 가치로 환산하여 합산
        - ✓Cyclic markov process에서의 infinite return 방지

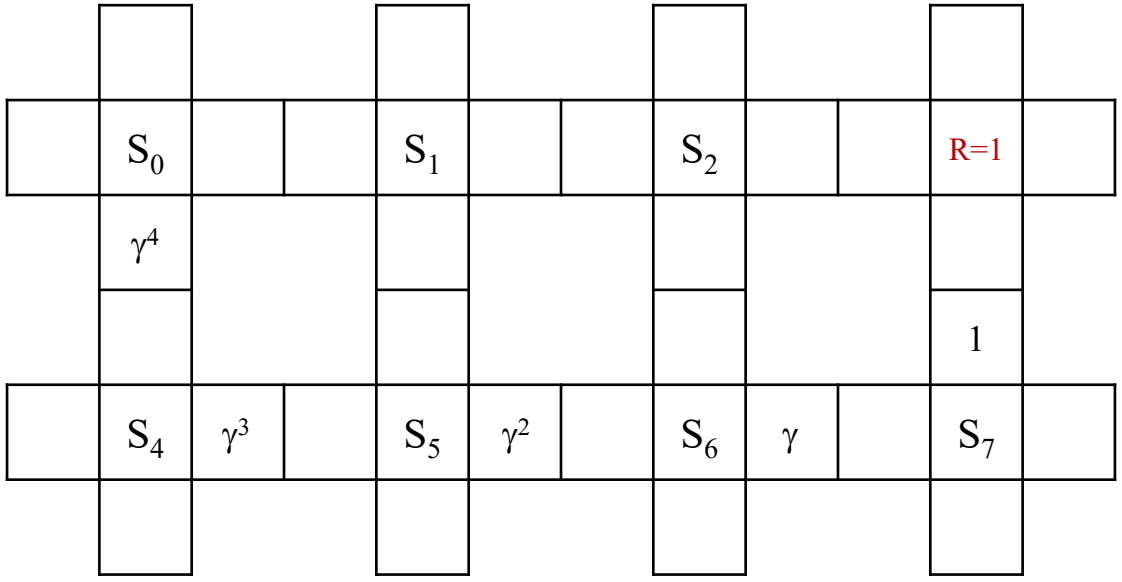
Value Function

- S_0~S_7로 구성된 state space
  - 각 state에서 상, 하, 좌, 우에 위치한 state로 transition 가능
  - 모든 state는 4개의 action value를 갖음
    - 가장 높은 value의 방향으로 transition을 수행
  - S_0에서 출발하여 S_3에 도달하면 1의 reward 부여



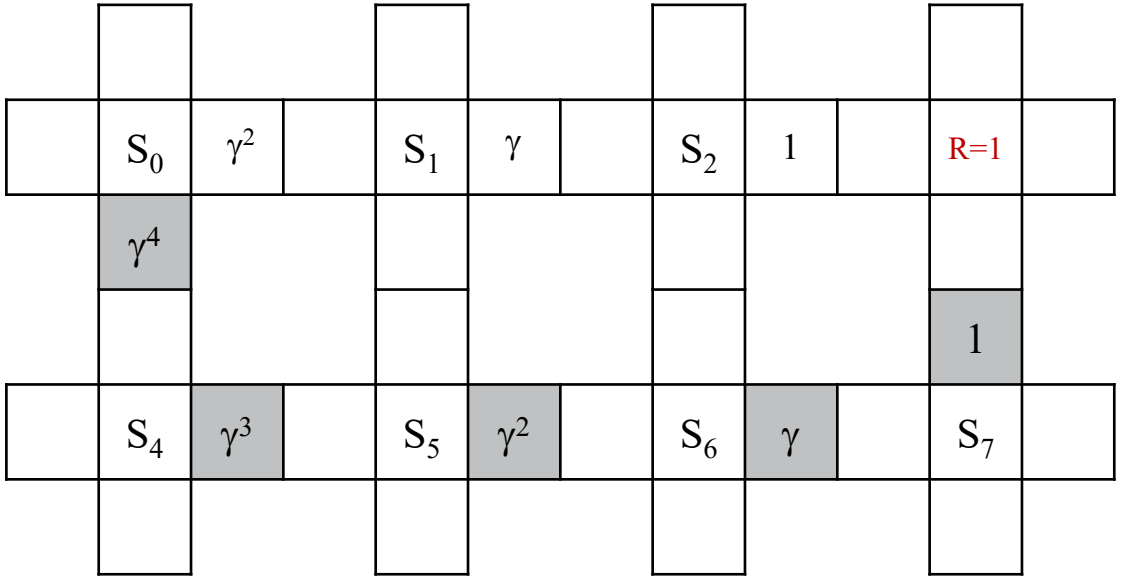
Value Function

- S_0 ▸ S_4 ▸ S_5 ▸ S_6 ▸ S_7 ▸ S_3 의 history로 이동하여 reward를 획득했다고 가정
  - Reward 얻게 되는 state의 action value 자리에 획득한 reward를 가져옴



Value Function

- S_0 ▸ S_1 ▸ S_2 ▸ S_3 의 history로 이동하여 reward를 획득
  - 이전의 경우보다 trajectory가 짧기 때문에 discount factor에 의해 reward 증가



- S_7 : 다음 state에서 reward 1 획득
- S_6 : 두 state 뒤에 reward를 획득하므로 γ를 곱함
- Return value : $G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \dots$

Value Function

- State value function (V)
  - 현재 state로부터 기대되는 return
    - 현재 state에 대한 가치를 판단할 수 있음.
    - 이 값을 최대화하는 것이 optimal policy가 됨
  - $V(S_t) \equiv \int_{S_{t+1:}\, a_\infty} G_t P(a_t, S_{t+1}, a_{t+1}, \cdots \mid S_t) da_t dS_{t+1} \cdots da_\infty$
    - 현재 state로부터 episode 종료까지 action을 선택할 확률, 다음 state로 transition할 확률과 각 state에서의 return값의 곱을 적분해 expected return 계산
- Action value function (Q)
  - State s 에서 action a를 취했을때 기대되는 return
    - Action a에 대한 가치를 판단할 수 있음
  - $Q_\pi(S_t, a_t) \equiv \int_{S_{t+1:}\, a_\infty} G_t P(S_{t+1}, a_{t+1}, \cdots \mid S_t, a_t) dS_{t+1} da_{t+1} \cdots da_\infty$

Bellman Equation

- State value function
  - $V(S_t) \equiv \int_{S_{t+1:}\, a_\infty} G_t P(a_t, S_{t+1}, a_{t+1}, \cdots \mid S_t) da_t dS_{t+1} \cdots da_\infty$

    $= \int_{a_t} Q(S_t, a_t) P(a_t \mid S_t) da_t$

    - 따라서 State value function은 해당 state의 가치를 나타내게 됨
- Action Value function
  - $Q_\pi(S_t, a_t) \equiv \int_{S_{t+1:}\, a_\infty} G_t P(S_{t+1}, a_{t+1}, \cdots \mid S_t, a_t) dS_{t+1} da_{t+1} \cdots da_\infty$

    $= \int_{S_{t+1:}\, a_\infty} G_t P(a_{t+1}, S_{t+2}, \cdots \mid S_t, a_t, S_{t+1}) P(S_{t+1} \mid S_t, a_t) dS_{t+1} da_{t+1} \cdots da_\infty$

    $= \int_{S_{t+1:}\, a_\infty} G_t P(S_{t+2}, a_{t+2}, \cdots \mid S_{t+1}, a_{t+1}) P(a_{t+1} \mid S_{t+1}) P(S_{t+1} \mid S_t, a_t) dS_{t+1} da_{t+1} \cdots da_\infty$

    Bellman equation
  - $P(a_{t+1} \mid S_{t+1})$: Policy
  - $P(S_{t+1} \mid S_t, a_t)$: Transition

Q-learning

- Optimal Policy
  - State value function의 값을 maximize하도록 policy를 update
    - Optimal Q를 의미하는 Q*를 구했다고 가정

$$\text{Max}(V(S_t)) \equiv \max\left(\int_{S_{t+1:}\, a_\infty} G_t P(a_t, S_{t+1}, a_{t+1}, \cdots \mid S_t) da_t dS_{t+1} \cdots da_\infty\right)$$

$$= \underset{P(a_t|S_t)}{argmax} \int_{a_t} Q^*(S_t, a_t) P(a_t|S_t) da_t$$

$$= \underset{P(a_t|S_t)}{argmax} \int_{a_t} \int_{S_{t+1:}\, a_\infty} G_t P(S_{t+2}, a_{t+2}, \cdots | S_{t+1}, a_{t+1}) P^*(a_{t+1}|S_{t+1}) P(S_{t+1}|S_t, a_t) dS_{t+1} da_{t+1} \cdots da_\infty P(a_t|S_t) da_t$$

  - Q*가 주어졌다는 것은 미래 state의 optimal policy는 구해진 것
    - 따라서 현재 state에서의 policy는 Q*중 가장 큰 값을 고르도록 설정하면 됨

$\therefore P^*(a_t|S_t) = \delta(a_t - a_t^*)$, a_t^* : 가장 큰 Q*를 선택하도록 하는 action

Q-learning

- Optimal action value function Q*을 어떻게 얻을 것인가
  - Monte carlo method

$$-Q(S_t, at) = \frac{1}{N}\sum_{i=1}^{N} G_t^{(i)}$$

∴ Reward에 도달할 수 있는 모든 episode를 수행하여 기대값을 구하는 방법

✓현실적으로 수행할 수 없음

  - Temporal difference[1)]

$$-Q(S_t, at) \simeq \frac{1}{N}\sum_{i=1}^{N}(R_t^{(i)} + \gamma Q(S_{t+1}^{(i)}, a_{t+1}^{(i)})) = QN$$

$$= Q_{N-1} + \frac{1}{N}(R_t^{(N)} + \gamma Q(S_{t+1}^{(N)}, a_{t+1}^{(N)}) - Q_{N-1})$$

$$-Q_N = (1-\alpha)Q_{N-1} + \alpha(R_t^{(N)} + \gamma Q(S_{t+1}^{(N)}, a_{t+1}^{(N)}))$$

- 하나의 step만을 보고 Q value를 update

∴ 1)에 의해 TD method로 수행하여도 optimal Q*에 수렴함이 증명됨

Policy

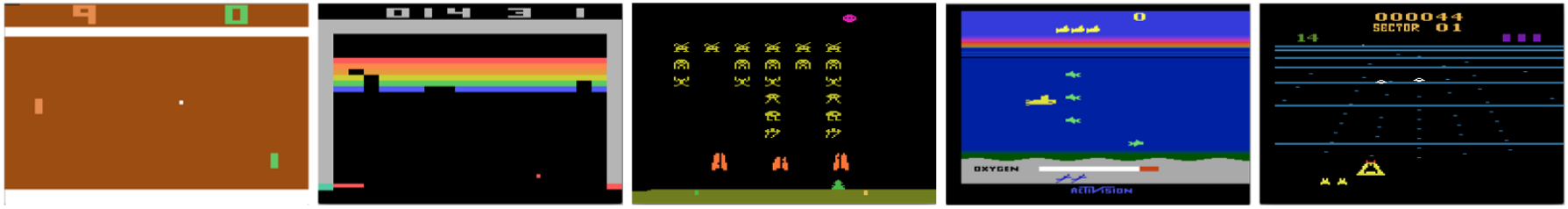
- On-policy
  - 학습시 사용하는 policy와 실제 환경에서 state transition에 사용하는 policy가 같은 경우
    - SARSA 등의 알고리즘이 해당됨
- Off-policy
  - 학습과 state transition에 사용되는 policy를 다르게 적용할 수 있음
    - 과거의 experience를 학습에 사용하는 것이 가능하게 함
      - 과거의 data에 현재의 policy를 적용했을 때 더 나은 action을 취하게 됨으로써 Q값을 재평가할 수 있게됨
        - ✓On-policy method는 학습할 때의 action과 실제 action이 동일해 이 과정이 불가능
    - Exploration을 자유롭게 하여 더 좋은 Q값을 기대할 수 있음
    - Q-learning이 해당됨

1) Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *NIPS Deep Learning Workshop*(2013).
2) Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

DQN[1), 2)]

- Environment의 고도화
  - 기존의 Q-learning 기법으로는 늘어난 state에 대해 연산, 학습이 불가
    - Value function을 weight를 사용한 deep neural network로 approximate
      - State와 action을 input으로 사용하여 각 action에 해당하는 Q value를 output

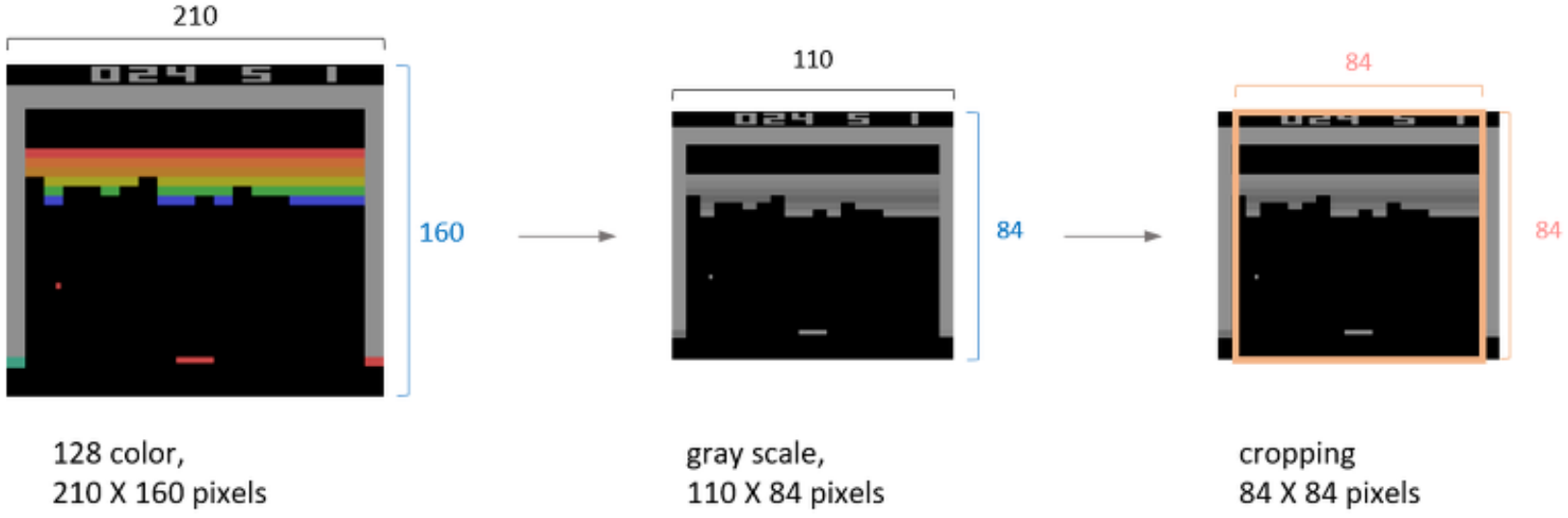
- Atari 2600 game을 DQN agent로 학습



1) Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *NIPS Deep Learning Workshop* (2013).
2) Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

DQN[1), 2)]

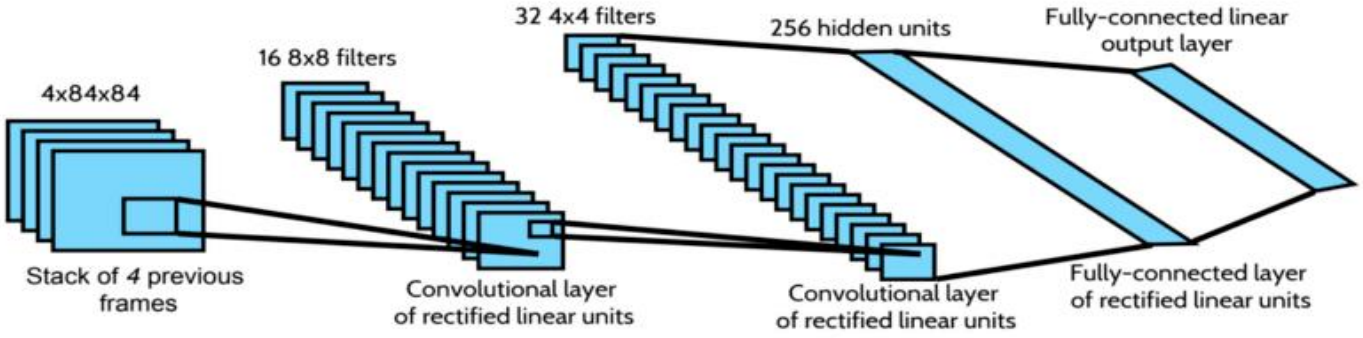
- Environment의 고도화
  - 게임 화면을 downsample, grayscale로 변환, 필요한 부분만 정사각형 크기로 crop하여 input으로 사용



1) Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *NIPS Deep Learning Workshop* (2013).
2) Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

DQN[1), 2)]

- Environment의 고도화
  - K번째 frame마다 frame을뽑아 stack하여 input으로 사용
    - 4장의 연속된 frame은 비슷한 정보를 담고 있어 학습이 잘 이루어지지 않음
    - 게임을 K배 만큼 더 빠르게 구동 가능
  - Output의 dimension은 action의 수와 같음
    - 각 action에 대한 Q value를 regression함
    - 모든 action에 대한 Q value를 한번에 얻어서 Max Q 연산에 용이



1) Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *NIPS Deep Learning Workshop* (2013).
2) Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

DQN[1), 2)]

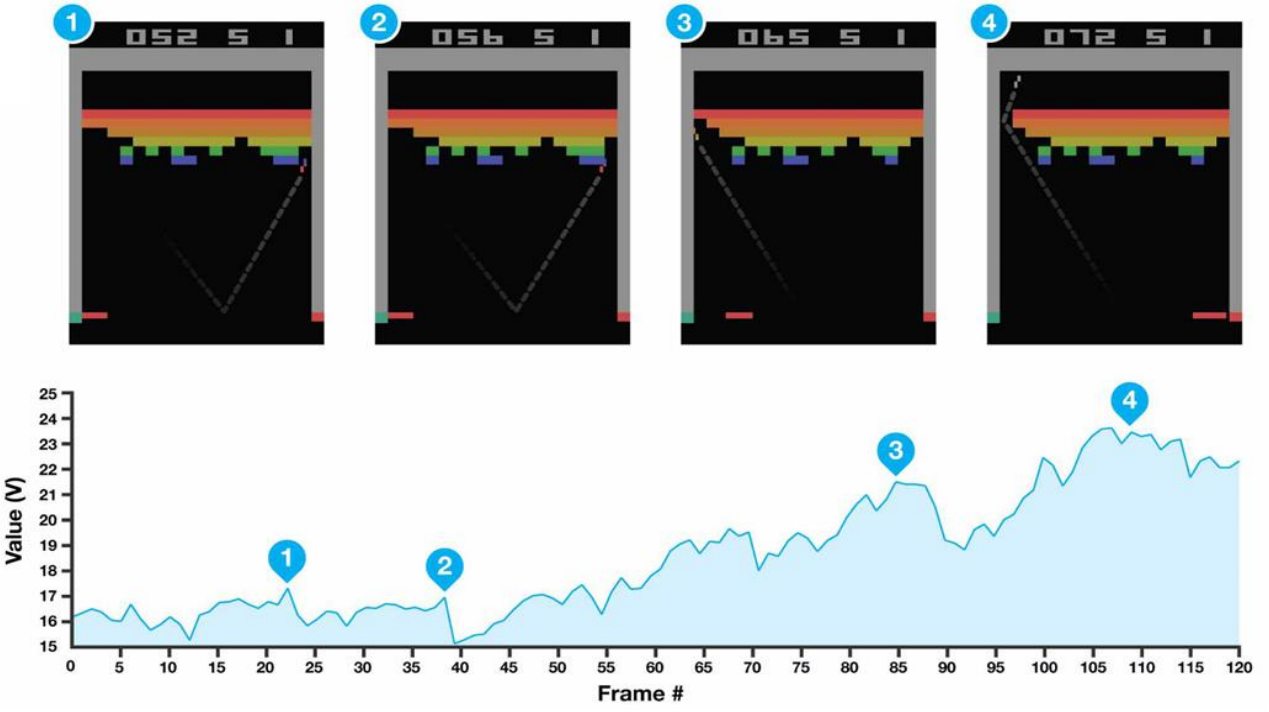
- Environment의 고도화
  - Experience replay
    - 게임을 진행하면서 생기는 최근 N개의 state transition 정보를 buffer에 저장
      - Experience replay로부터 minibatch를 뽑아 학습
        - ✓State correlation 감소
        - ✓이전의 experience로 현재의 policy를 다시 평가할 수 있음
  - Loss function

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a\sim\rho(\cdot);s'\sim\mathcal{E}}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)\right)\nabla_{\theta_i} Q(s, a; \theta_i)\right]$$

  - Exploration
    - Eplison-greedy policy
      - Policy에 의한 action을 취하지 않고 epsilon에 따라 random하게 action
        - ✓Epsilon 값을 episode가 지남에 따라 일정한 비율로 감소시킴

1) Mnih, Volodymyr, et al. "Playing atari with deep reinforcement learning." *NIPS Deep Learning Workshop* (2013).
2) Mnih, Volodymyr, et al. "Human-level control through deep reinforcement learning." *nature* 518.7540 (2015): 529-533.

DQN[1), 2)]



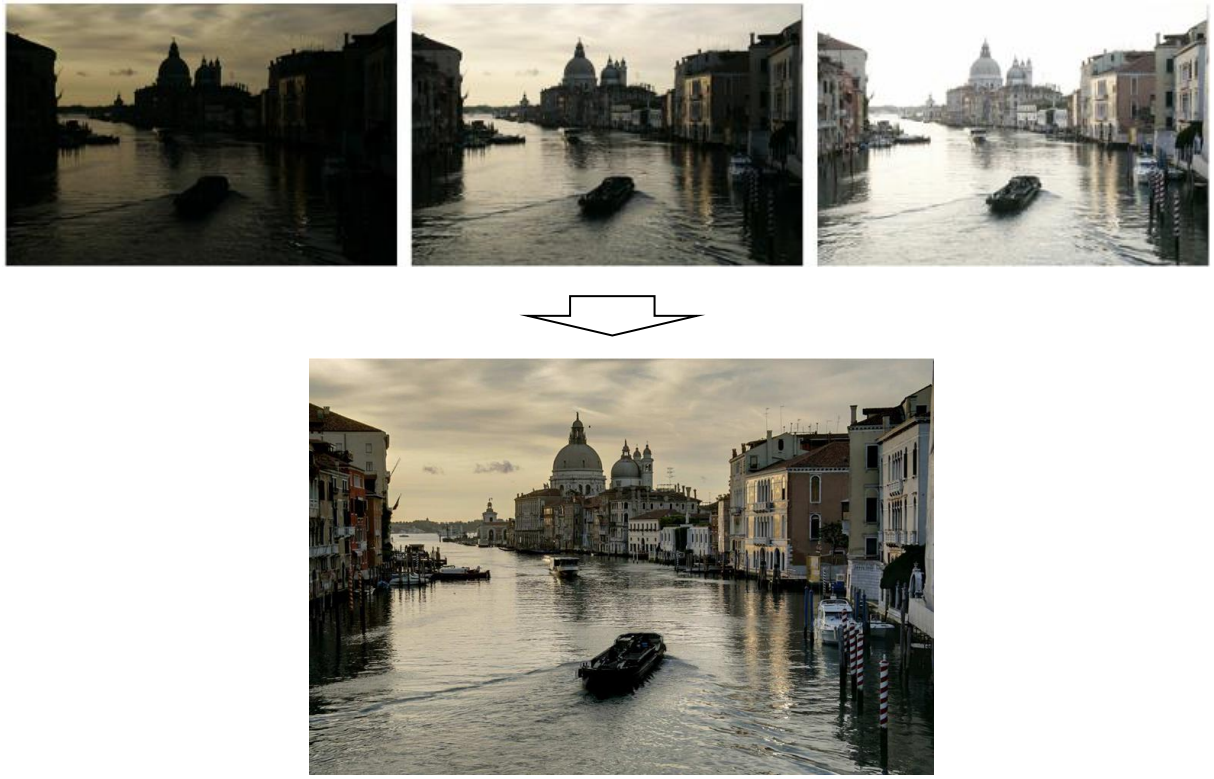
	Pong	Breakout	S. Invaders	Seaquest	B. Rider
Random	−20.4	1.2	179	110	354
Sarsa [3]	−19	5.2	271	665	996
Contingency [4]	−17	6	268	723	1743
DQN	**20**	**168**	**581**	**1705**	**4092**
Human	−3	31	3690	28010	7456

RL Algorithms

- Model-free
  - Simulator가 있다는 가정으로 수행됨
    - Agent가 environment를 이해하는 것은 아니며 단순한 상호작용을 통해서 next state, reward를 획득함
- Model-based
  - State와 action으로부터 다음 state와 reward를 예측하는 model 학습
    - 학습된 model을 사용하여 action을 결정하는 policy 학습에 사용함
      - State와 reward를 예측할 수 있기 때문에 model free 기법에 비해 학습 속도가 빠름
      - 다만 복잡한 state를 예측하는 model의 error에 의해 학습이 매우 어려움
- Value based / Policy based
  - DQN은 획득한 Q network로부터 policy를 구함 (value based)
  - Policy based method는 state로부터 직접적으로 policy를 학습

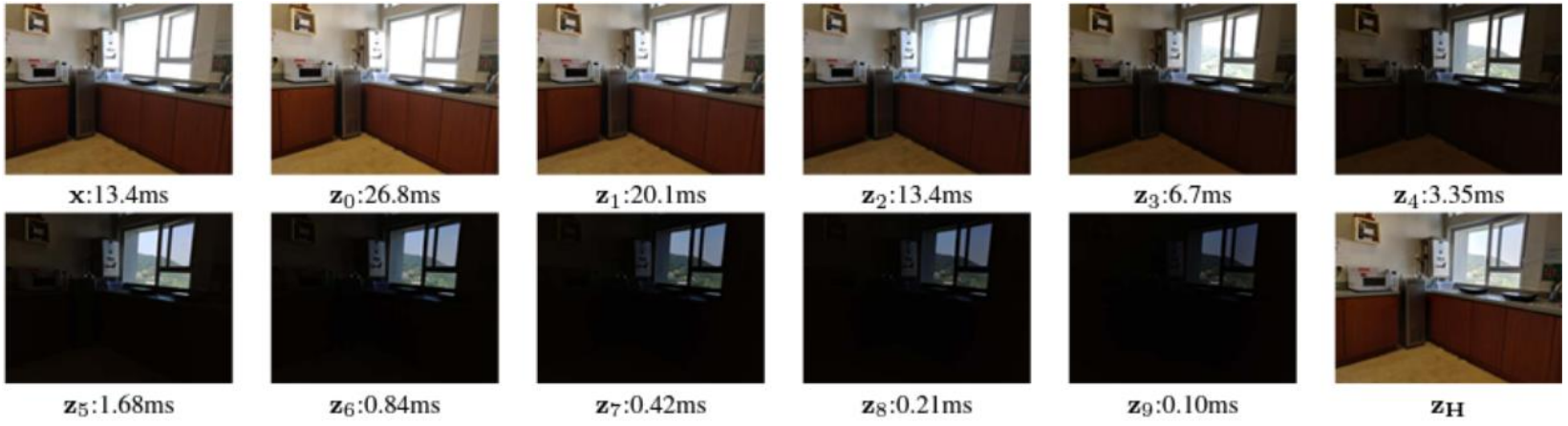
HDR 관련 논문[1)]

- Multi-exposure fusion
  - 여러 low dynamic range image를 합성하여 high dynamic range 이미지 생성



HDR 관련 논문[1)]

- Proposed method
  - 카메라의 Auto-exposure로 얻은 이미지와의 셔터속도 비율 조절을 통해 10장의 LDR 사진 획득
    - 기존의 exposure fusion algorithm을 사용하여 HDR 이미지를 얻어 GT로 사용

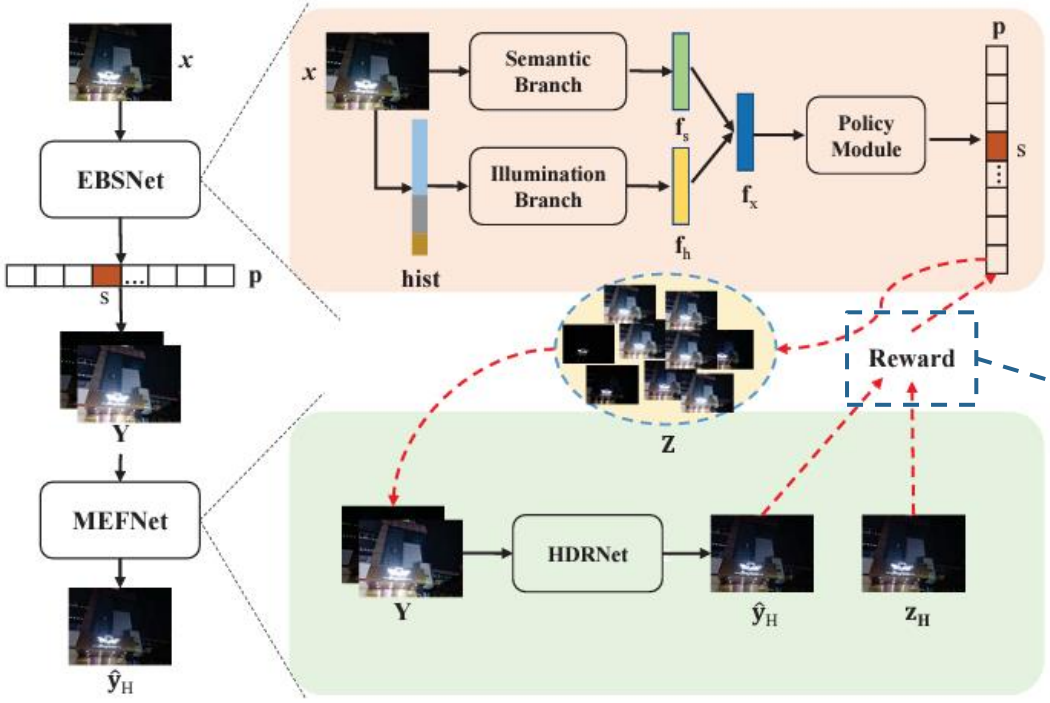


HDR 관련 논문[1)]

- Proposed method
  - 10장의 이미지 중 fusion에 사용할 K개의 이미지를 선택하는 agent 학습
    - Auto-exposure 이미지 자체를 state로 사용하는 EBSNet 제안
      - 내부적으로 두 개의 branch로 구성
        - ✓Alexnet 구조를 가진 sementic branch
        - ✓Histogram을 입력으로 받는 illumination branch
      - 두 branch를 concatenate하여 fully connected layer를 거쳐 10개의 image중 K개의 이미지 선택에 대한 확률 출력
    - Multi-exposure fusion network인 MEFNet 제안
      - 선택된 K개의 LDR 이미지를 합성하여 GT와 비교해 PSNR 획득
        - ✓ GT 생성에는 10개의 image를 모두 사용

HDR 관련 논문[1)]

- Proposed method



얻은 PSNR중 최대값과 비교하여 이번 step의 PSNR이 더 크다면 양의 reward, 더 작다면 음의 reward 부여

HDR 관련 논문[1)]

- Experiment

HDR 관련 논문[1)]

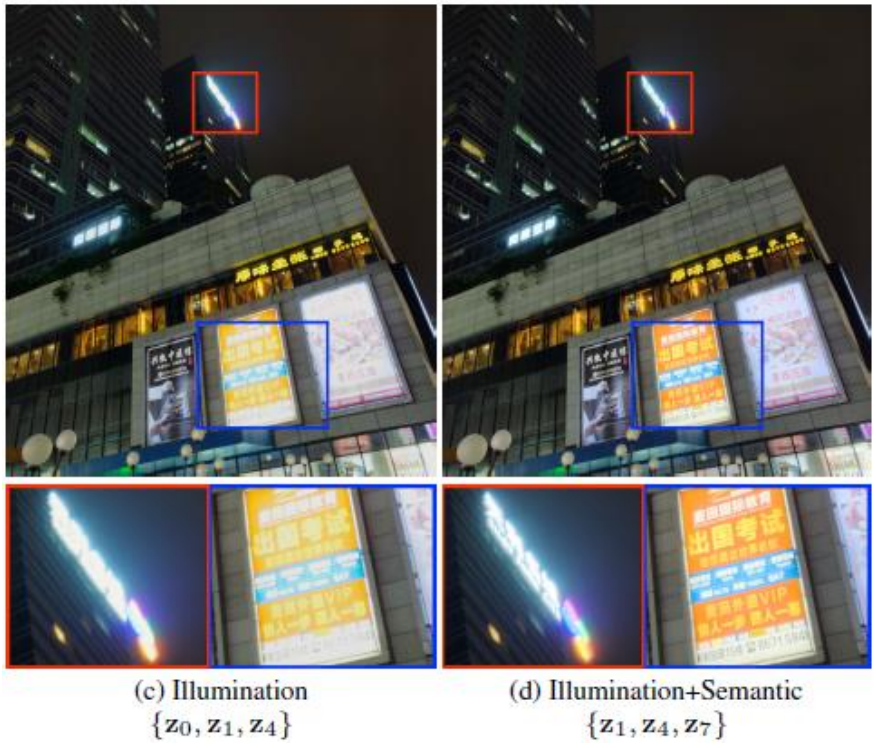
- Experiment
  - Comparison
    - 제안된 MEFNnet을 사용하지 않고 GT를 만들 때 사용한 algorithm으로 fusion한 경우와 비교했을 때 PSNR 상승
    - 선택하는 이미지 개수(K)에 따른 처리 속도, 성능 비교

Method	PSNR
Barakat [2]	27.31
Pourreza-Shahri [23]	26.44
Beek [30]	27.46
EBSNet+EF[20]	28.15
EBSNet+MEFNnet	28.41

K	1	2	3	10
PSNR	26.06	27.53	28.41	30.14
time/ms	218.43	301.21	352.27	811.25

HDR 관련 논문[1)]

- Experiment
  - EBSNet의 sementic, illumination branch가 없는 경우와 성능 비교



(c) Illumination $\{z_0, z_1, z_4\}$

(d) Illumination+Semantic $\{z_1, z_4, z_7\}$

Method	PSNR
Semantic	27.15
Illumination	27.40
EBSNet+MEFNet	28.41