SKInnovation
Adv.CDS 과정
실습2

실습2 : DQN

실습 목표 :

- **Deep Q network** 코드 구현 및 결과 확인
- 구현한 **DQN** 코드를 다양한 환경에 적용 해보기

실습 환경

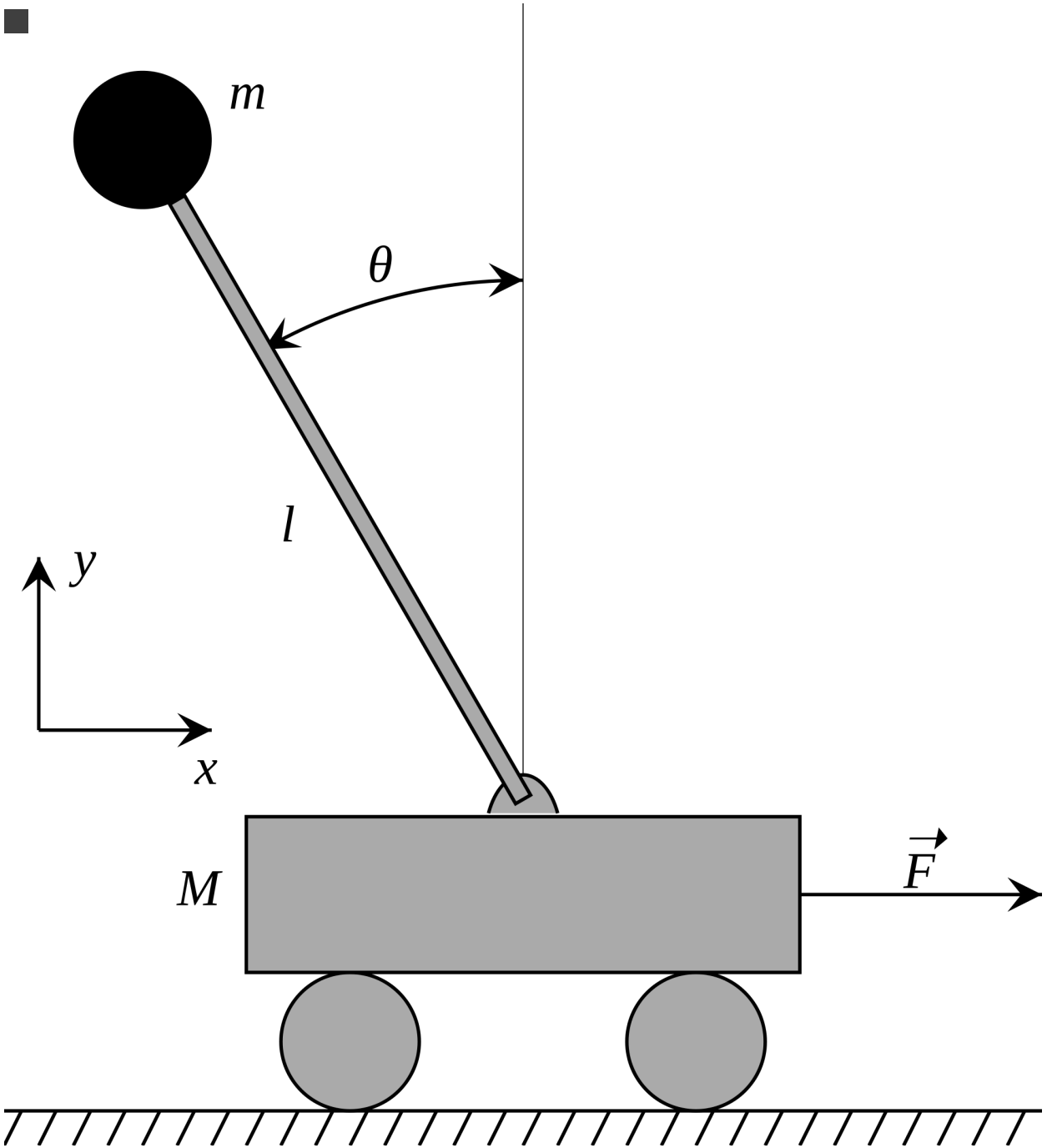
- **CartPole-v0**
- **Highway env**
- **SmartGrid env**

1 CartPole with DQN 예제

2 {} with DQN 실습

3 SmartGrid 환경 설명

4 SmartGrid with DQN 실습

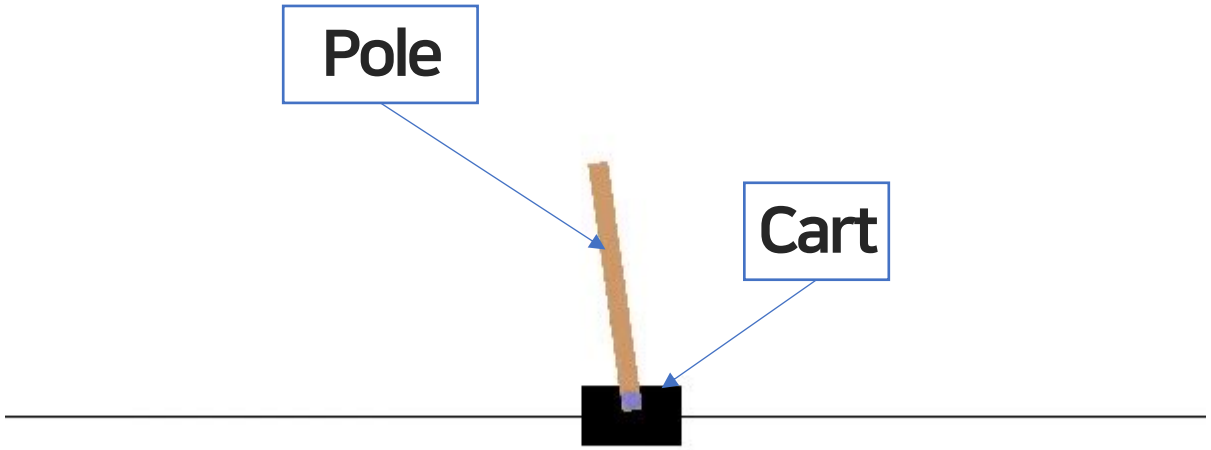


CartPole with DQN

CartPole with DQN

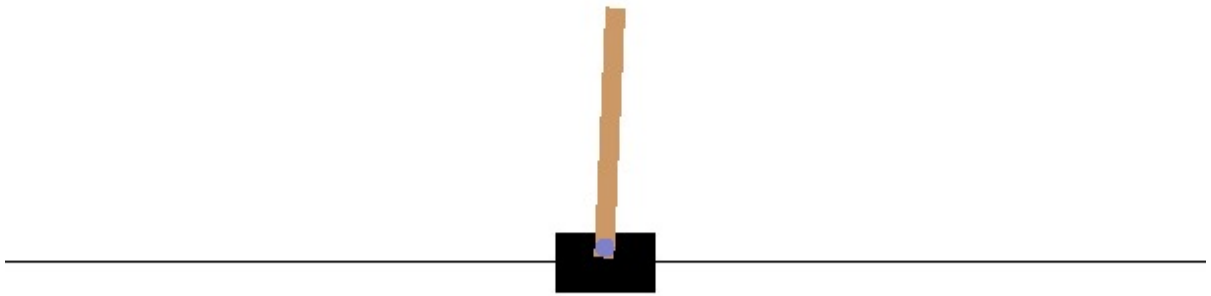
학습 목표 :

- **DQN** 코드 구현
- **Experience Replay** 및 **Target Network** 순차적 구현
- **Open AI gym CartPole-v0** 환경에 적용



CartPole

- **Cart**를 움직여 **Pole**의 균형을 잡는 문제
- **Cart**를 왼쪽 또는 오른쪽으로 밀 수 있음
- 매 **step** 마다 **+1** 의 **reward**를 받음
- 막대가 수직으로부터 **15**도 기울어지거나,
  화면 끝으로 나가면 종료
- **Goal : Episode reward > 195**



State
= Box(4,)

Num	Observation	Min	Max
0	Cart Position	-4.8	4.8
1	Cart Velocity	-Inf	Inf
2	Pole Angle	~ -41.8°	~ 41.8°
3	Pole Velocity At Tip	-Inf	Inf

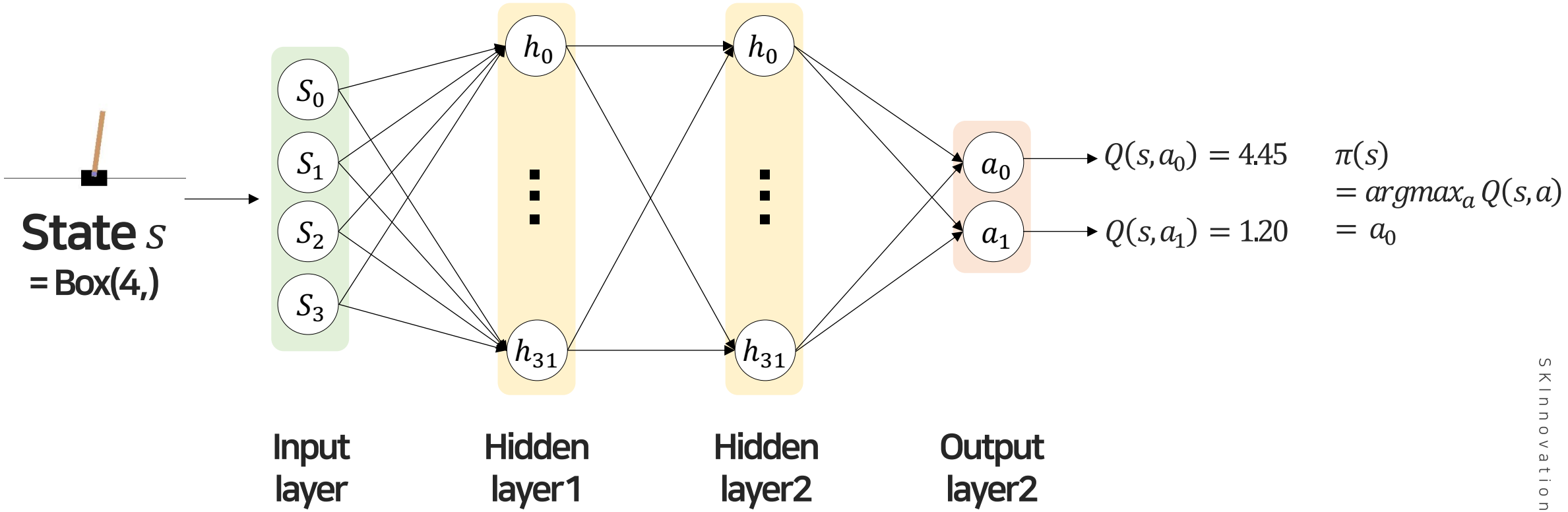
- **4**개의 숫자로 구성된 **4-dimensional box space (vector)**
- 각 숫자는 일정한 범위 **(min ~ max)** 사이의 연속된 값의 하나로 결정됨

Action
= Discrete(2)

Num	Action
0	Push cart to the left
1	Push cart to the right

- **0** 또는 **1** 로 결정되는 이산 **space**

Network Overview



DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$

0. Initialize replay buffer, network, target network

For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

1. Select action using epsilon-greedy

DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

2. Take step and store transition to replay buffer

DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For $t = 1$,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $Q = Q$
  End For
End For

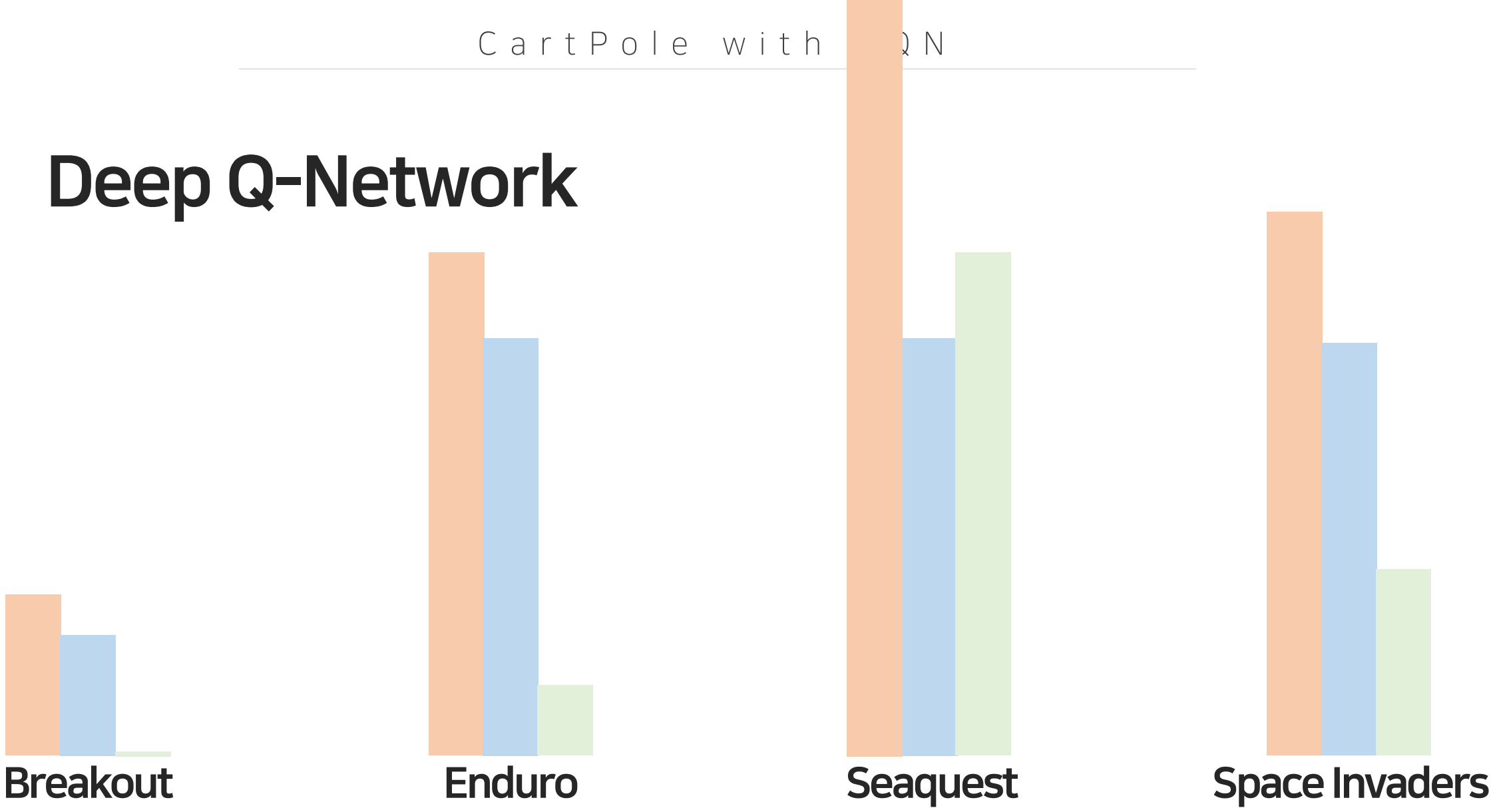
3. Train network

DQN Overview

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
For episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$
  For t = 1,T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in D
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D
    Set $y_j = \begin{cases} r_j & \text{if episode terminates at step j+1} \\ r_j + \gamma \max_{a'} \hat{Q}(\phi_{j+1}, a'; \theta^{-}) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_j; \theta)\right)^2$ with respect to the network parameters θ
    Every C steps reset $\hat{Q} = Q$
  End For
End For

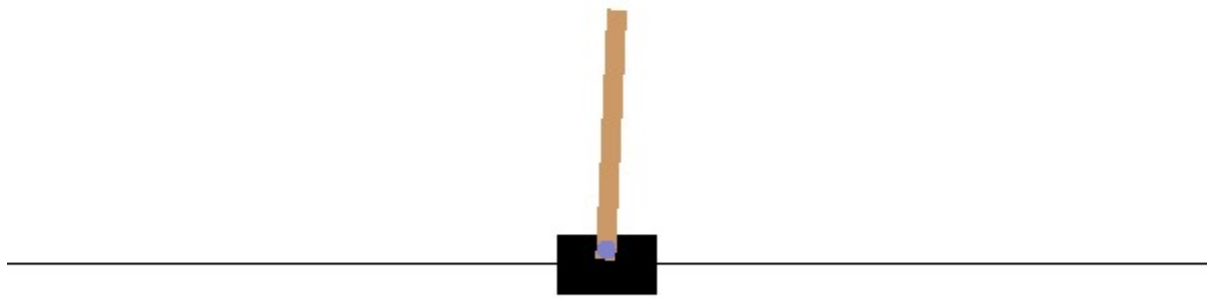
4. Update target network

Deep Q-Network



Mnih, V. et al. Human-level control through deep reinforcement learning, Nature(2015)

실습
DQN 구현



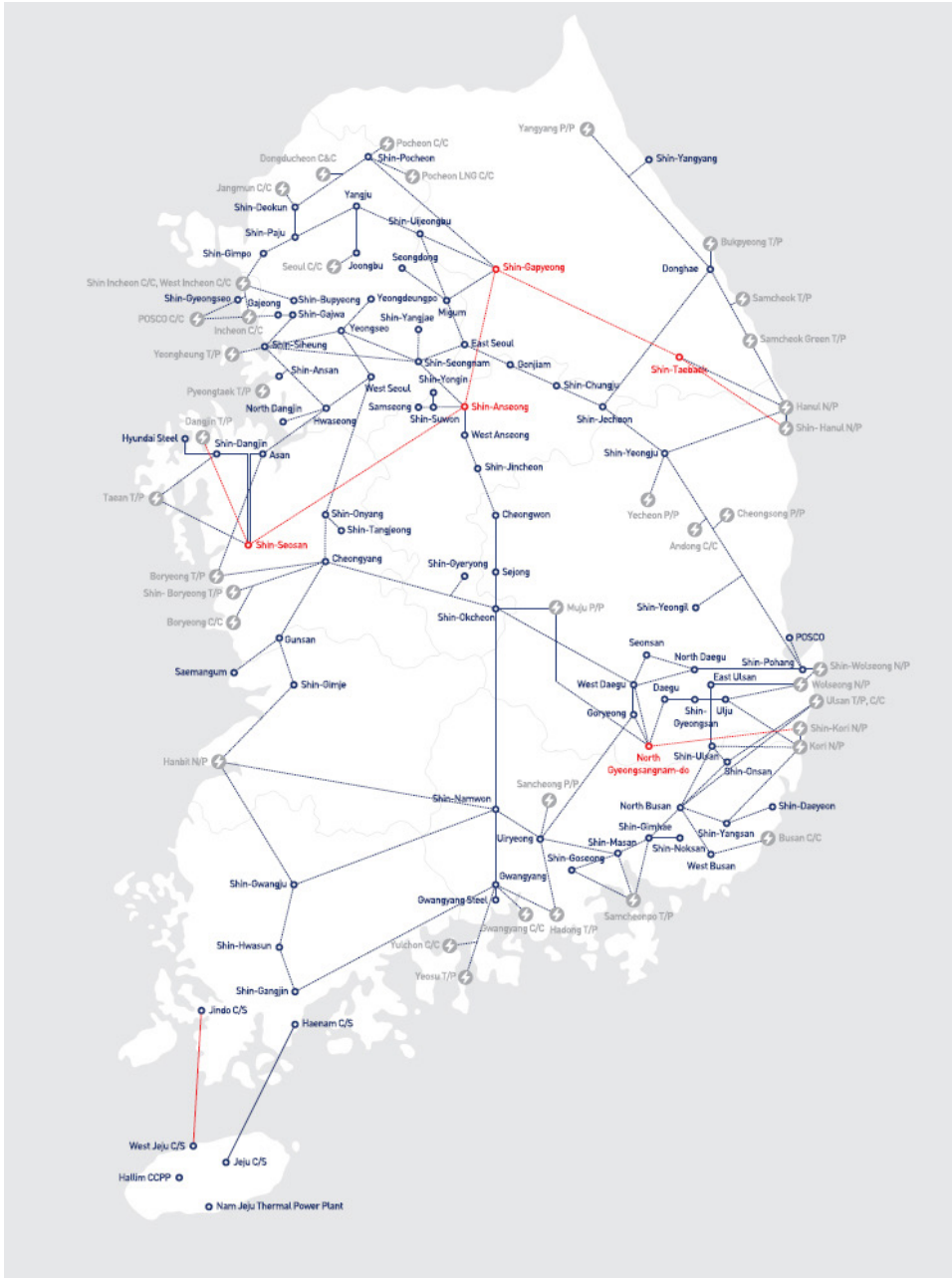
1_dqn.ipynb

SmartGrid Environment

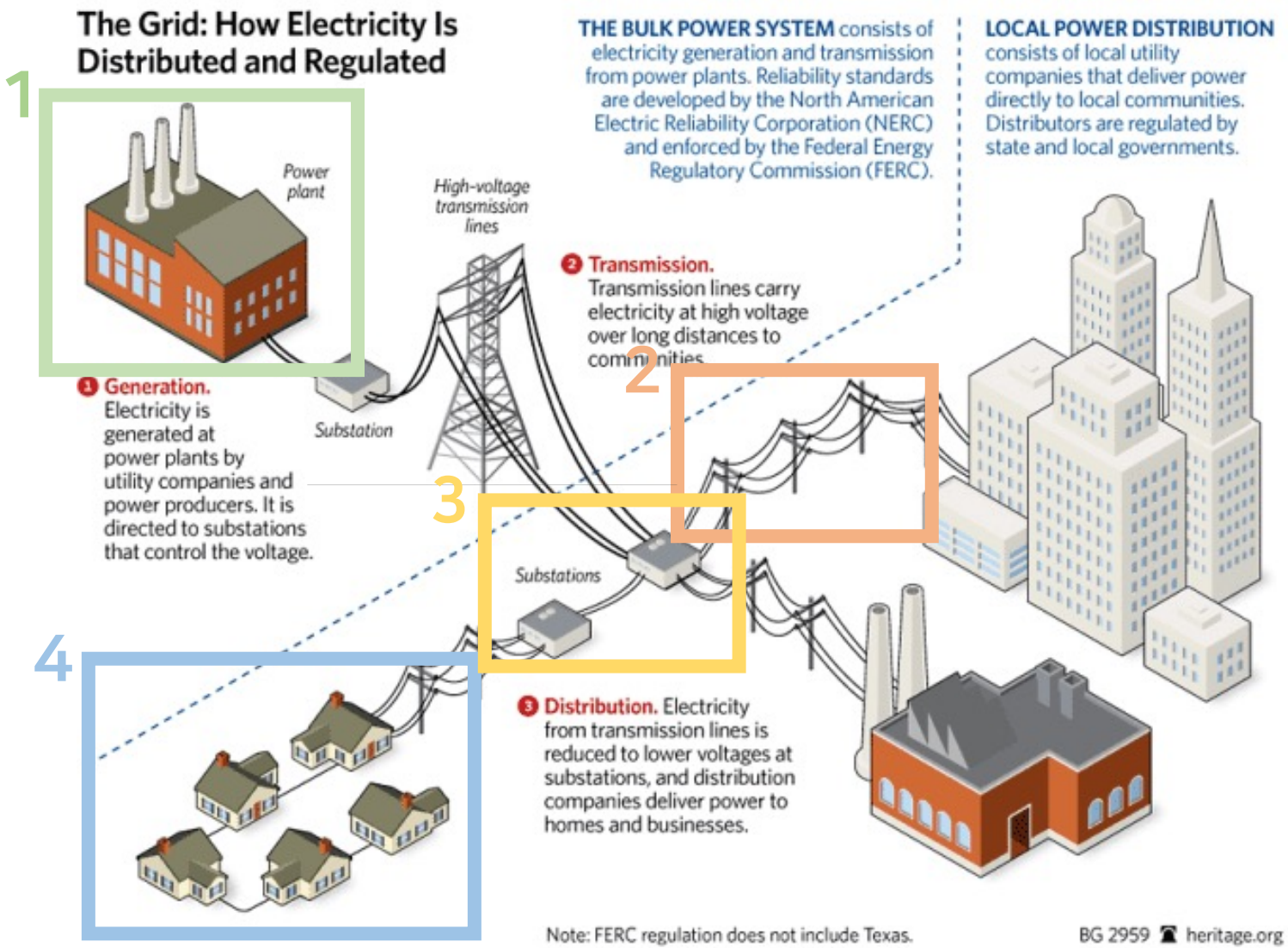
Smart Grid

학습 목표

- 전력망 제어 (**Power Grid Operation**) 환경 설명 및 필요성 인식
- **Smart Grid** 환경 설명

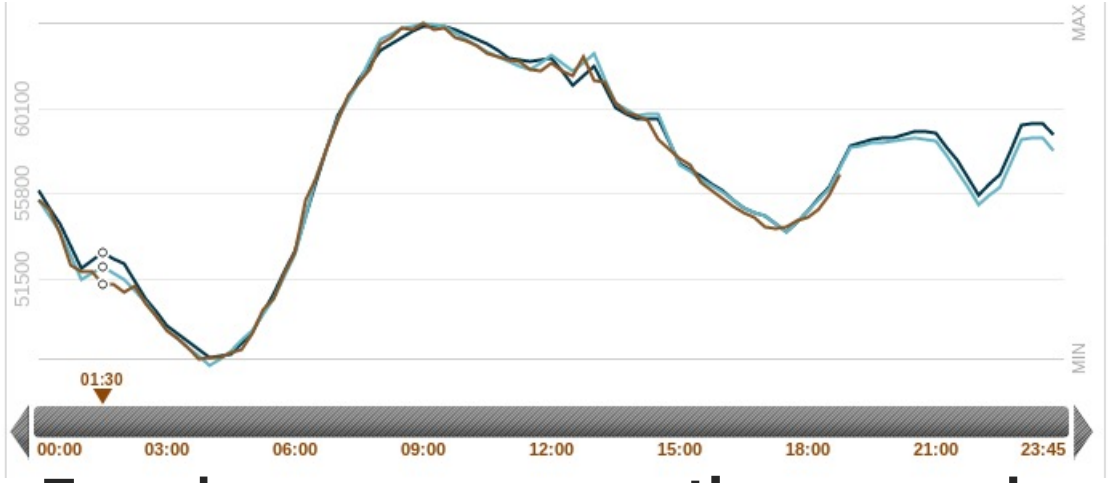


The Reality

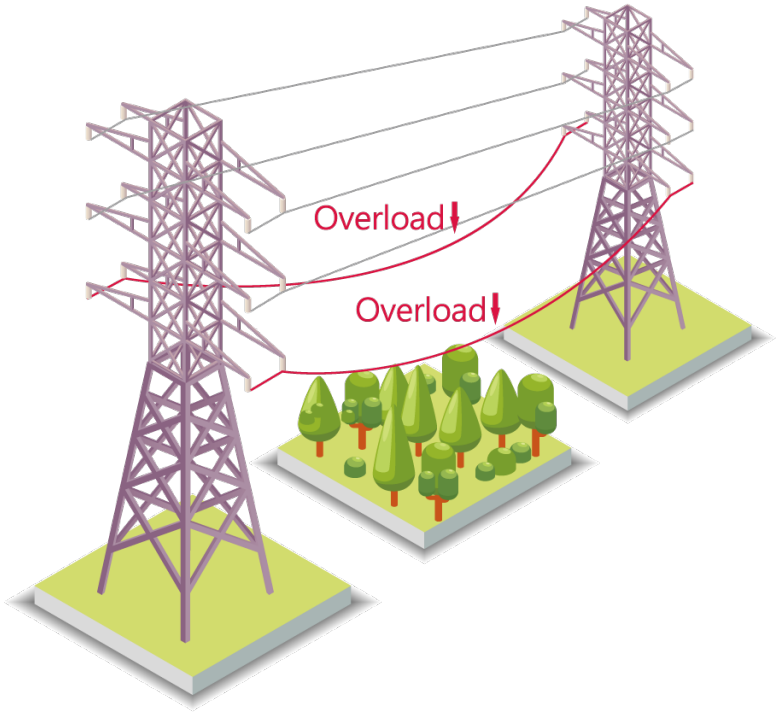


1. Generator
2. Powerline
3. Substation
4. Load

Motivation



French power consumption over a day



Powerline can be overloaded

Maintenance

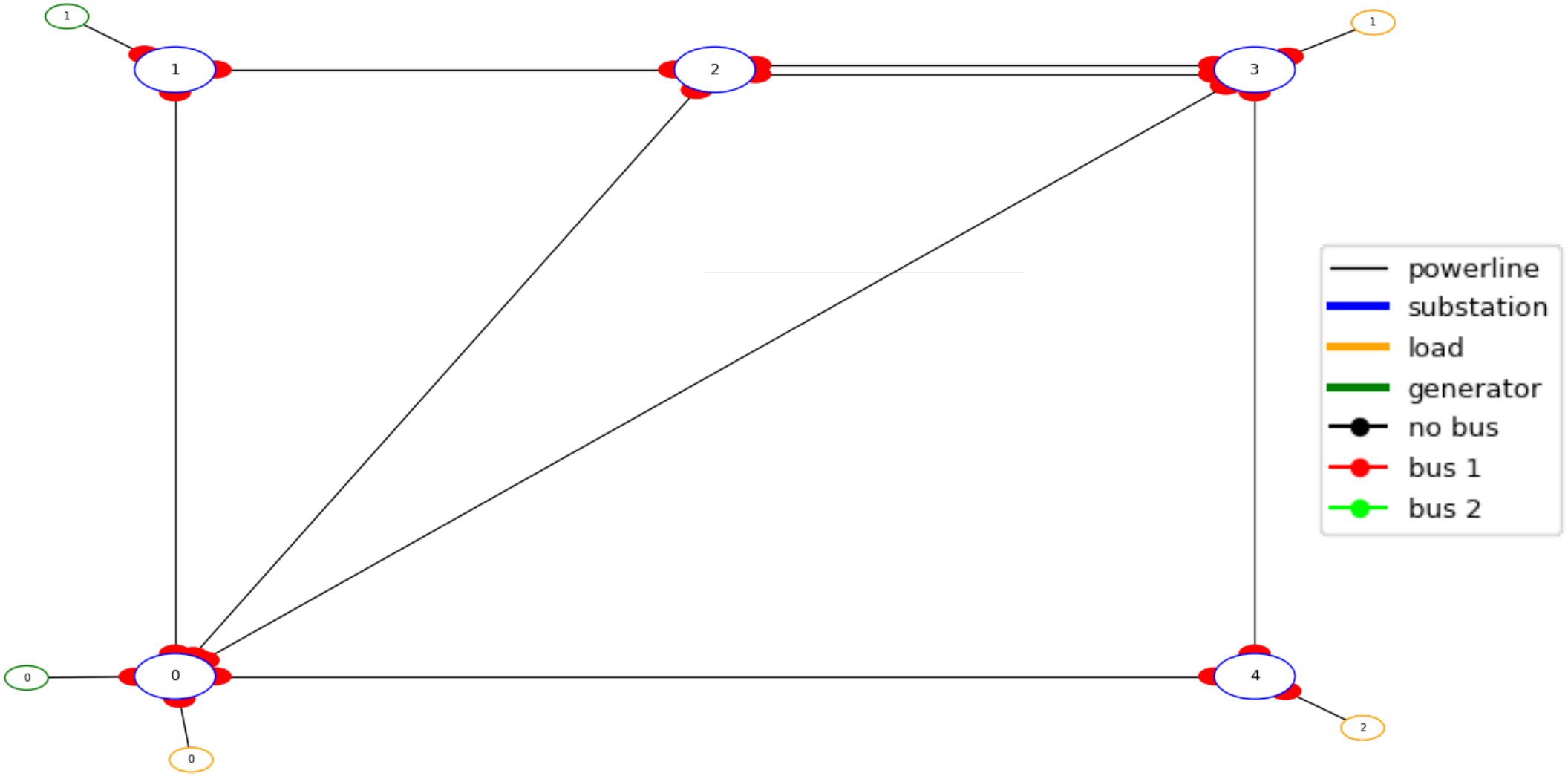
Motivation

How to operate?

- 일반적으로는 실시간으로 전력망을 감시하여 문제 상황을 예측하고 제어
- **SmartGrid** 환경은 전력망 제어 문제를 자동화하고자 하는 목적으로, 전력망을 모사한 환경을 강화 학습을 통해 해결하고자 함

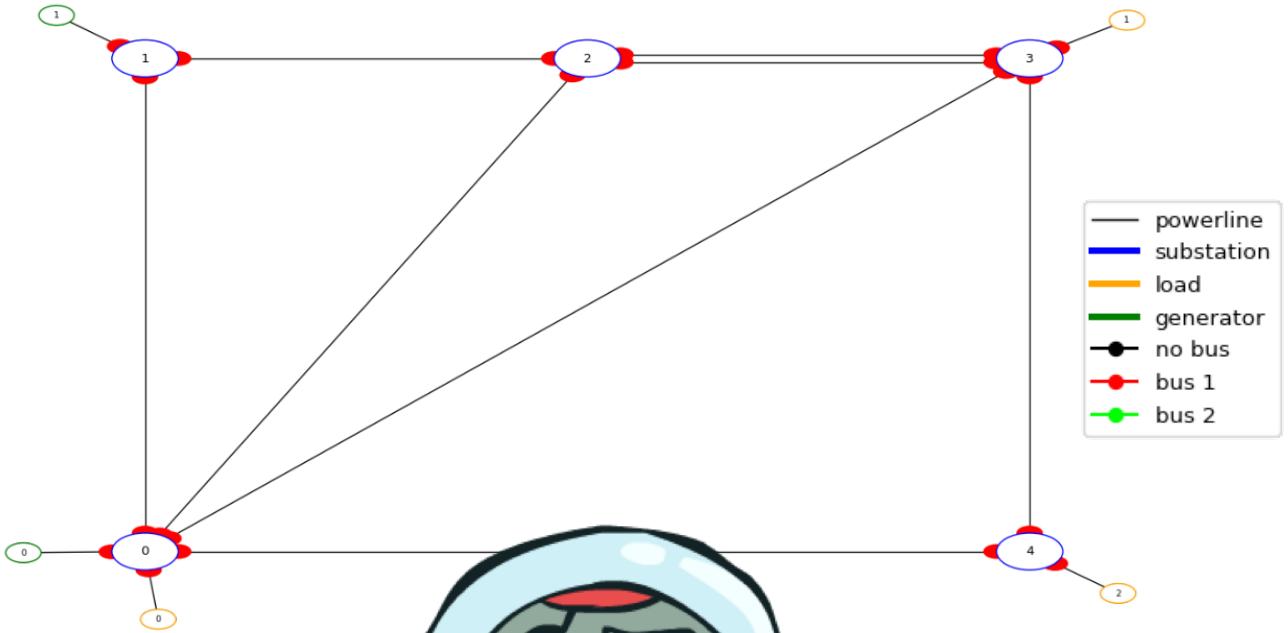
Graphs at Transmissions Grid Operations center in Portland

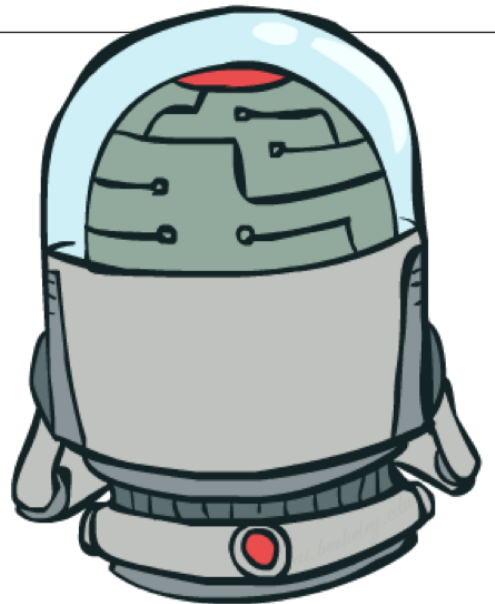
SmartGrid Env



Fully Observable

- **Agent**가 **전력망 전체**의 정보를 관측
- 관측한 정보 **(state)** 를 기반으로 전력망 내의 특정한 요소에 대해 적절한 **action**을 수행함으로써 **전체 전력망**을 안전하게 운영하는 것을 목적으로 함.

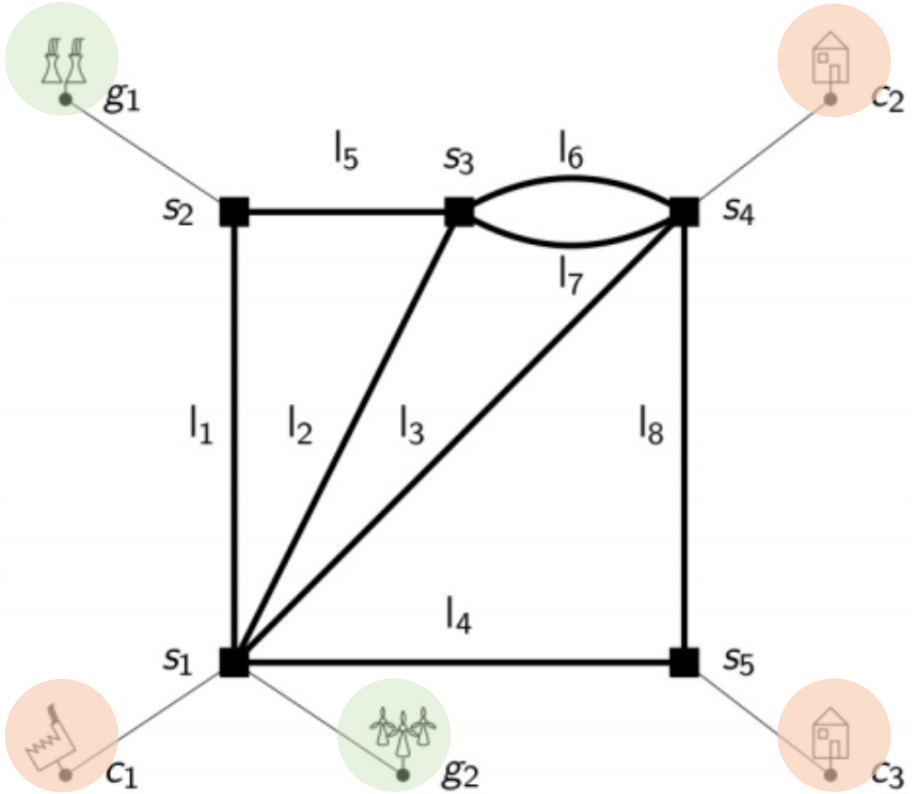




SmartGrid Env

State (Observation)

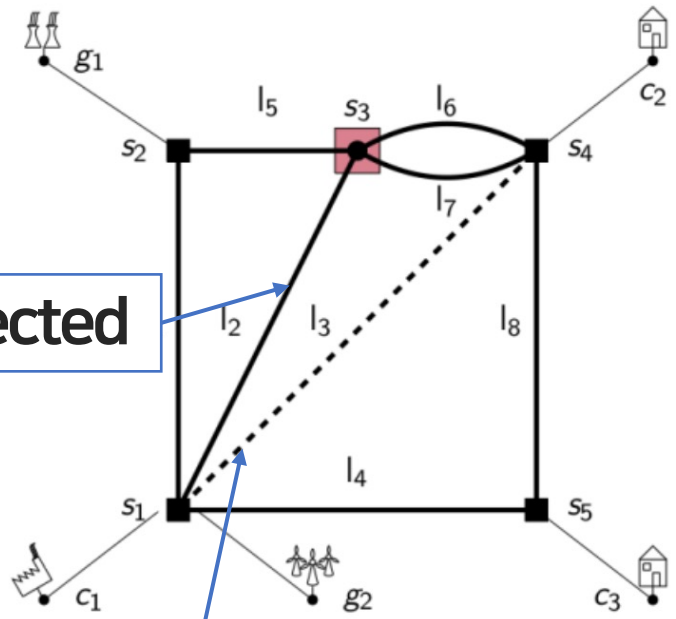
- Vector (39,)
- 6 Generator Info (Production, 생산량)
- 4 Load Info (Consumption, 소비량)
- 8 Powerline Info (flow / thermal limit)
- 21 Bus info (connection information)



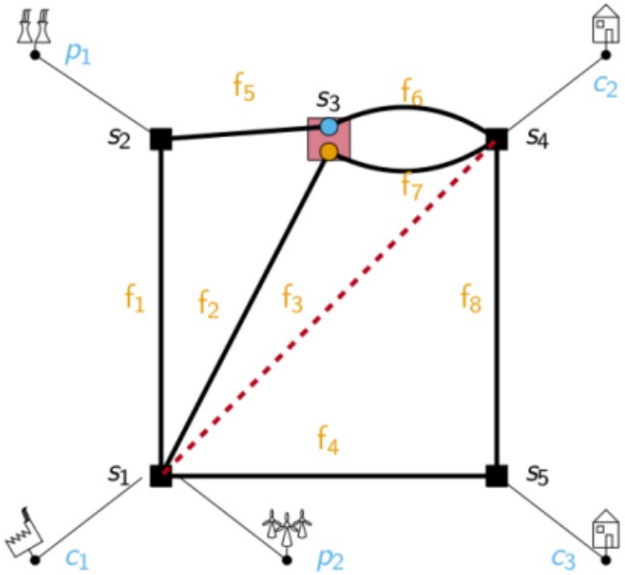
SmartGrid Env

Action

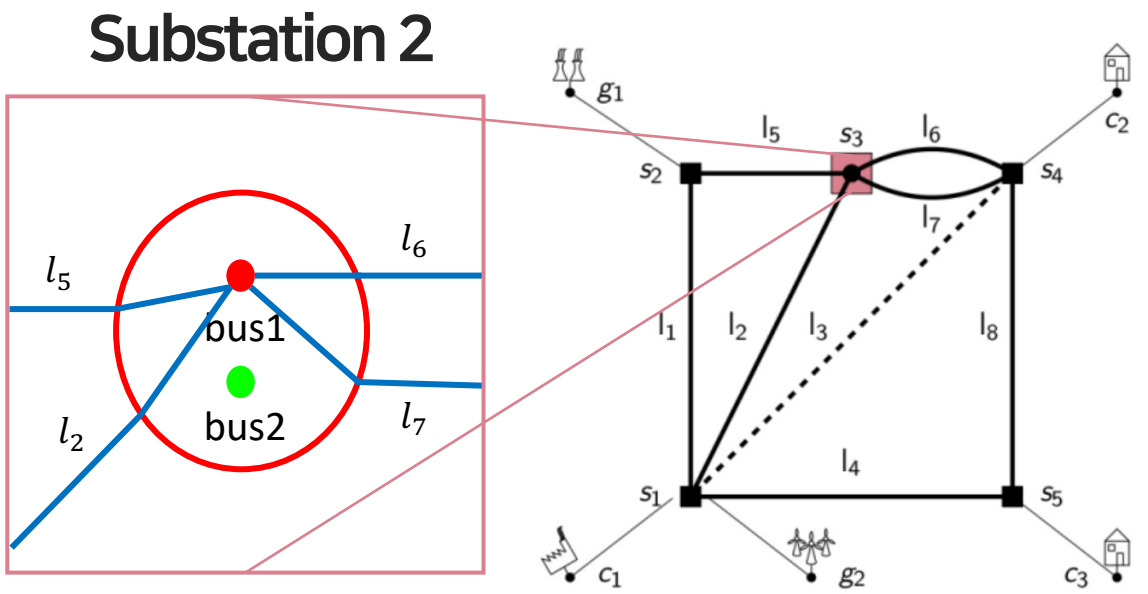
- Discrete action (67)
- 1 do nothing
- 8 power line change
- 58 change bus

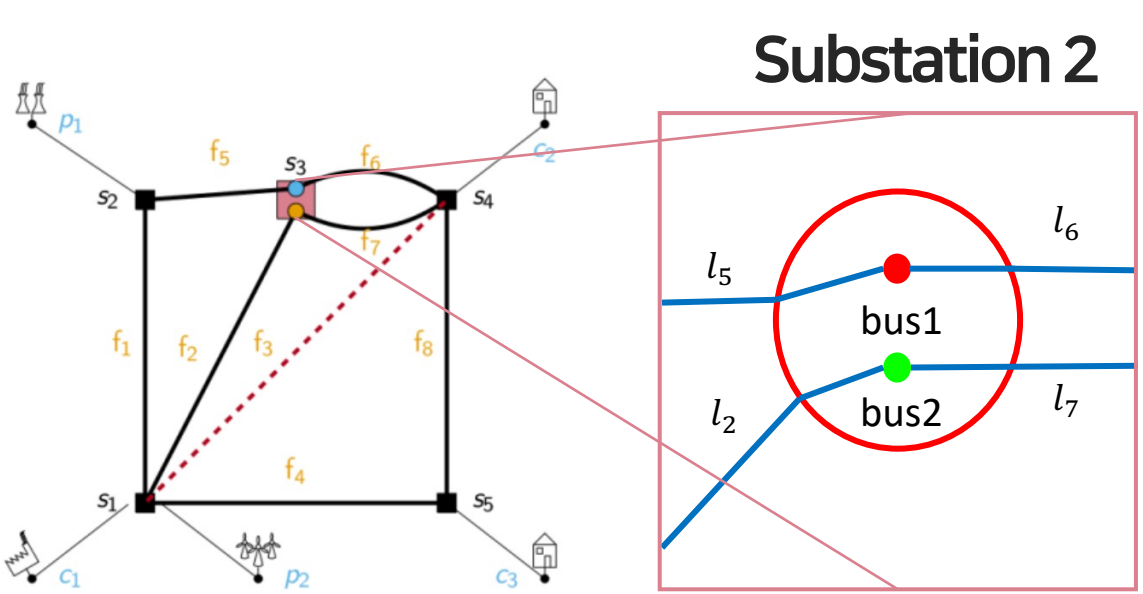


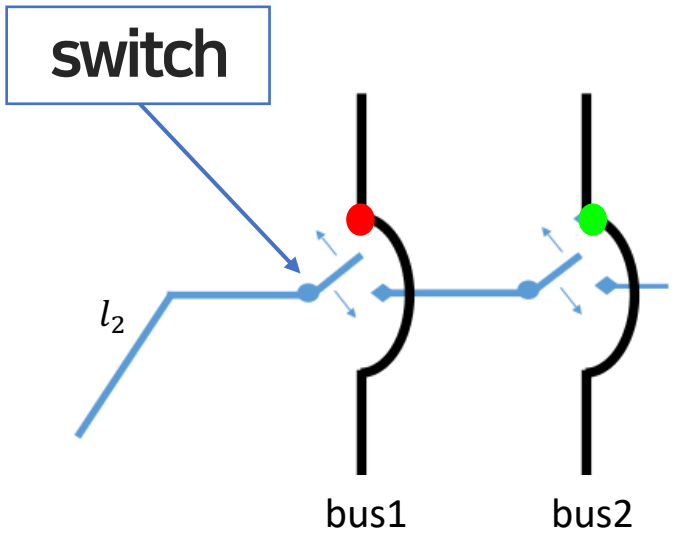
Disconnected



Change Bus

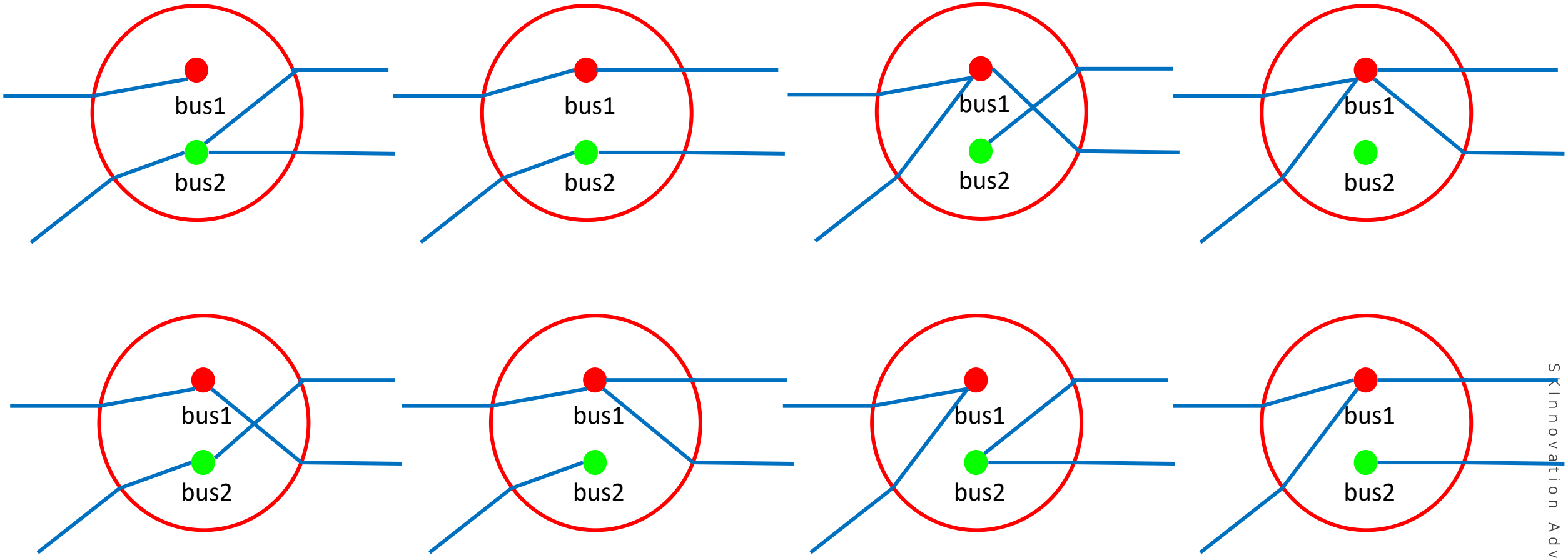






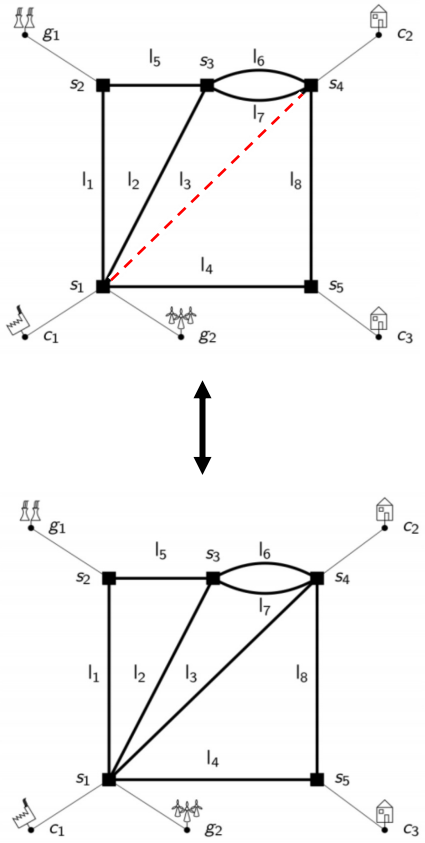
- 한 **substation** 내에서 어느 **bus** 에 연결될지 스위치를 통해 조정 가능
- 다른 **bus** 에 연결된 전선은 물리적으로 **분리** 되어 있음.

SmartGrid Env

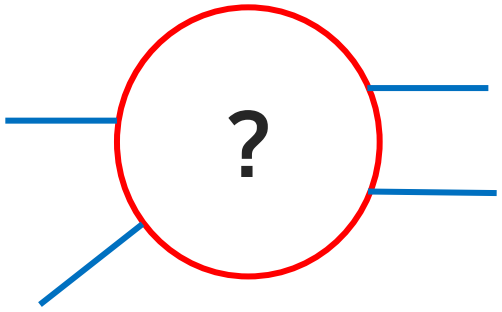


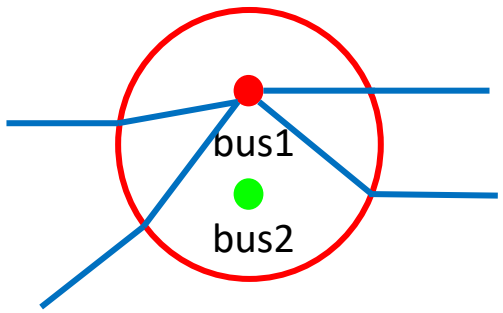
1 action = 1 substation change

Action example
(0 ~ 66)

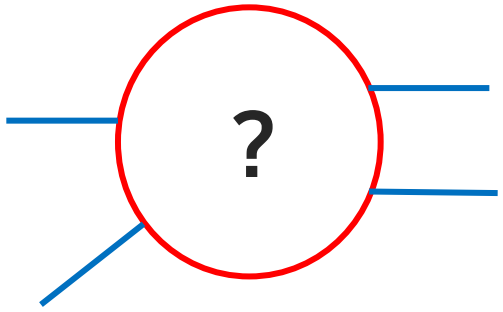


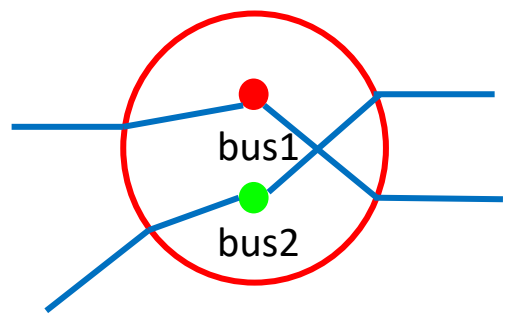
action
=3





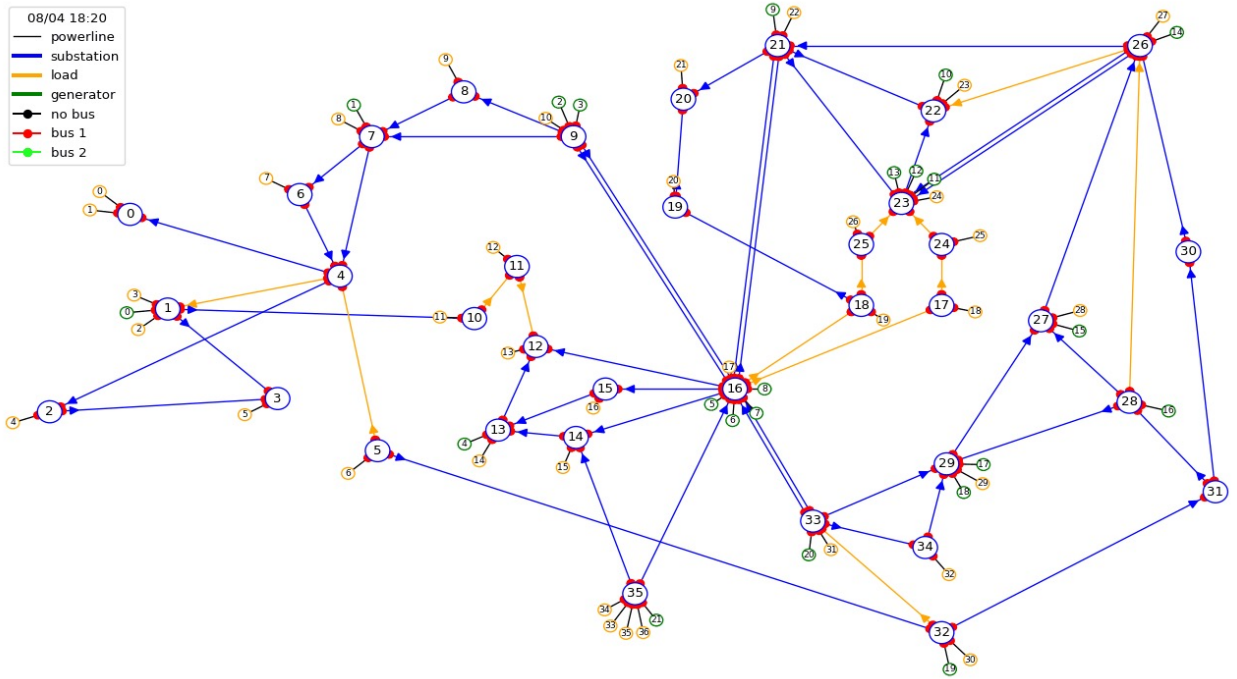
action
= 41





action
= 46

실습
SmartGrid Environment



3_SmartGrid.ipynb

SmartGrid with DQN

SKInnovation Adv.CDS

RL : Only for Gym?

- **Gym** 은 **RL algorithm**을 개발하고 비교하기 위한 **toolkit**
- 편한 사용을 위해 쉬운 **framework**를 제공하지만, **Gym**의 문법을 따르지 않는 환경 또한 존재
- 환경에 맞춰 알고리즘을 수정할 수 있어야 함.

Gym	SmartGrid
State = Box(4,)	SmartGrid state
Action = Discrete(2)	SmartGrid action

Gym

SmartGrid

State

Grid state

Converter!

Action

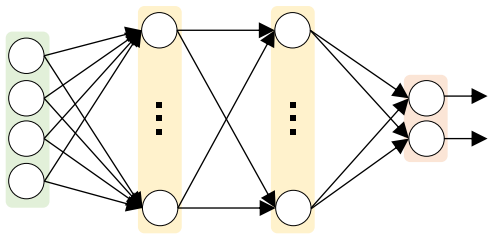
Grid action

Gym

SmartGrid

State

Grid state



Converter!

env.step

Action

Grid action

실습

SmartGrid with DQN

4_SmartGrid_DQN.ipynb

Appendix

CartPole with DQN

0_dqn.ipynb
1_cartpole.ipynb
dqn.py

SmartGrid with DQN

SmartGrid Environment

3_SamrtGrid.ipynb
grid_agent.py

SmartGrid with DQN

4_SamrtGrid_DQN.ipynb
grid_agent.py