# Introduction to Reinforcement Learning 4

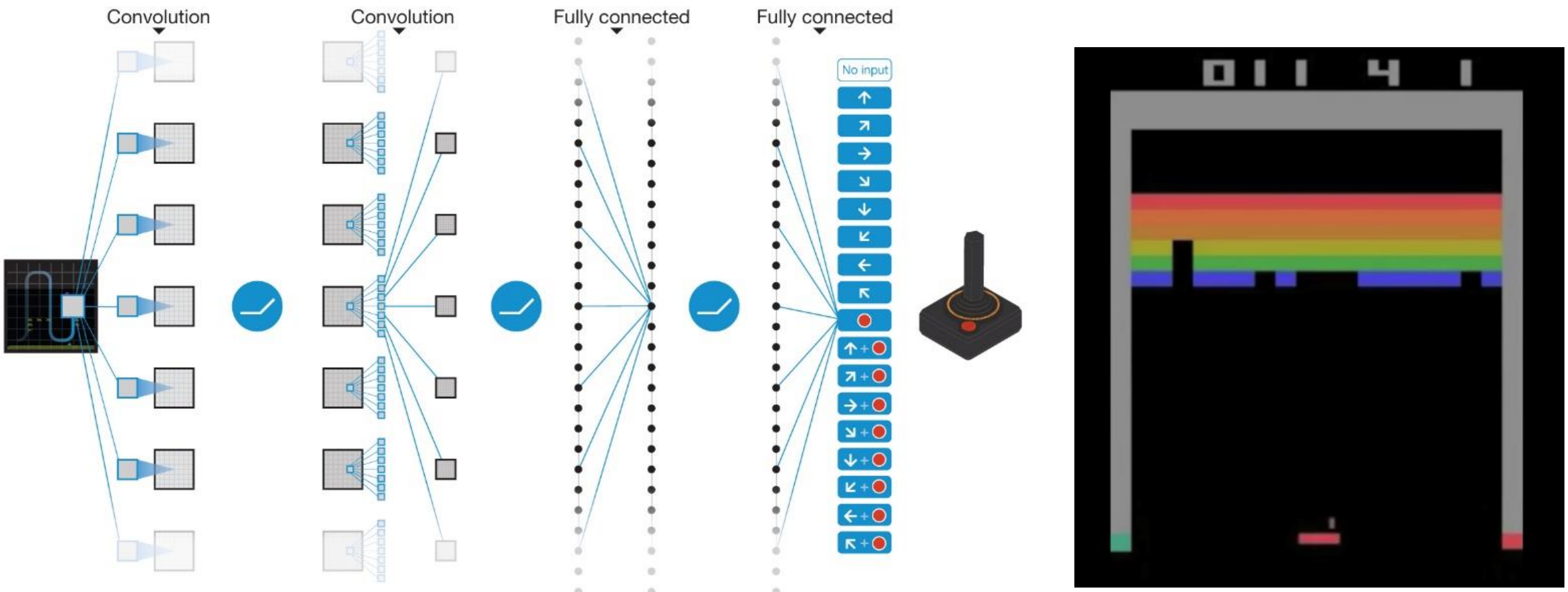
2020-11-12 Jungwook Mun

KAIST
NMAIL
Neuro-Machine Augmented Intelligence Lab

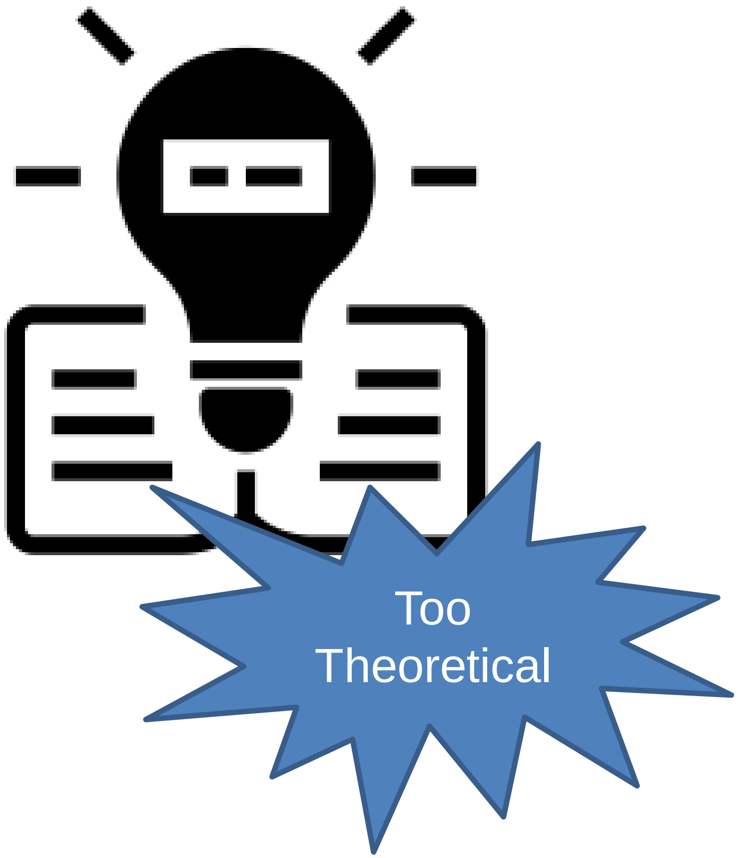
# Review

## In the previous lecture, you learned:

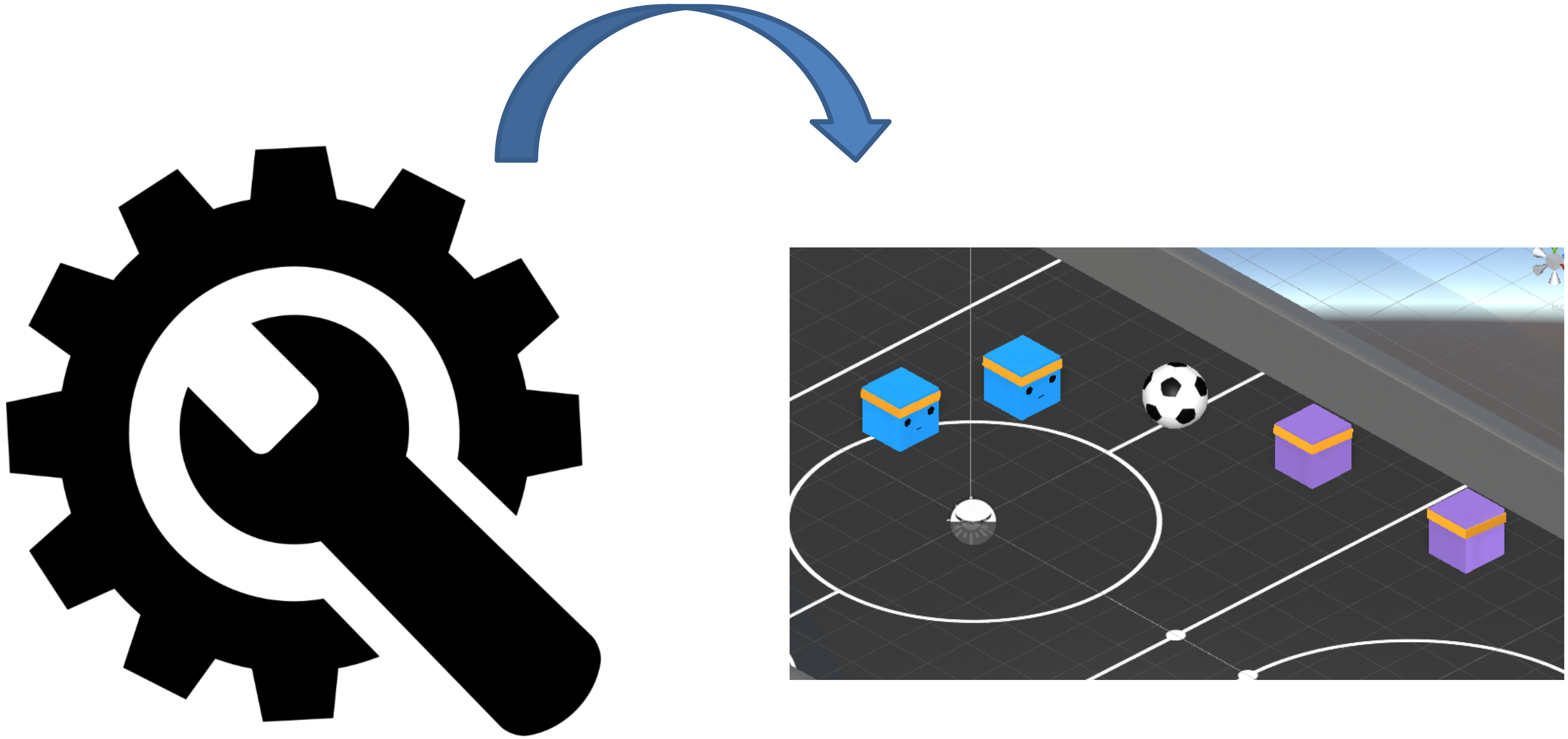
- Deep Q-learning concept



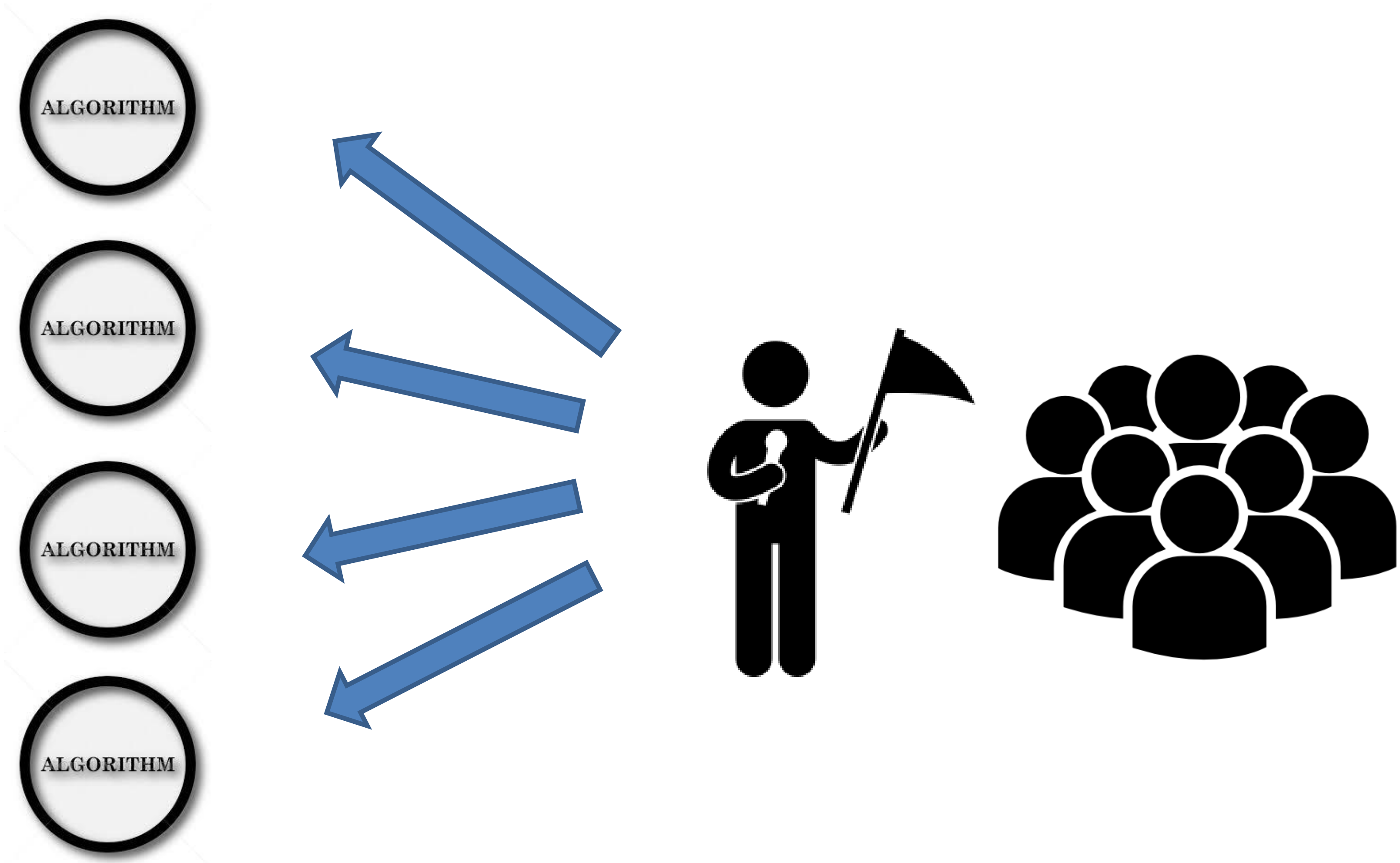
# Before this lecture



# Before this lecture

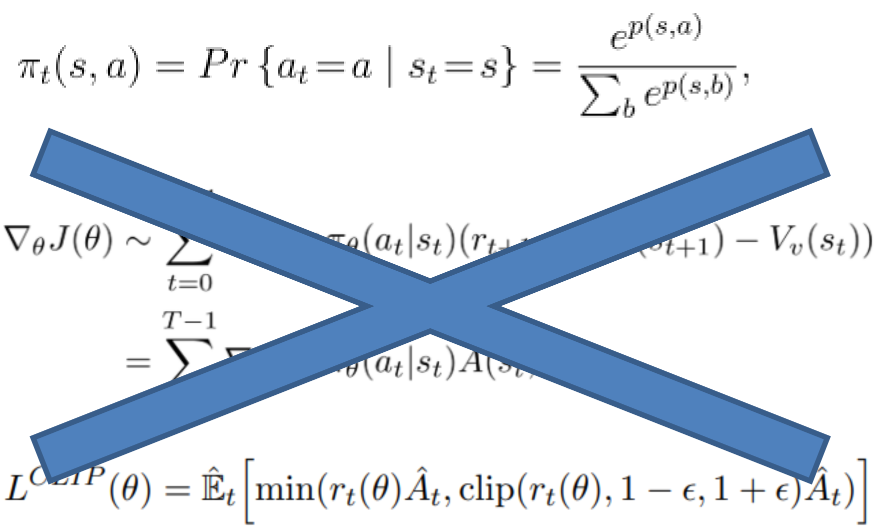


# Before this lecture



# Before this lecture

- There are no formulation but only introduce intuitive algorithm concepts and tutorials how to apply it to our code.

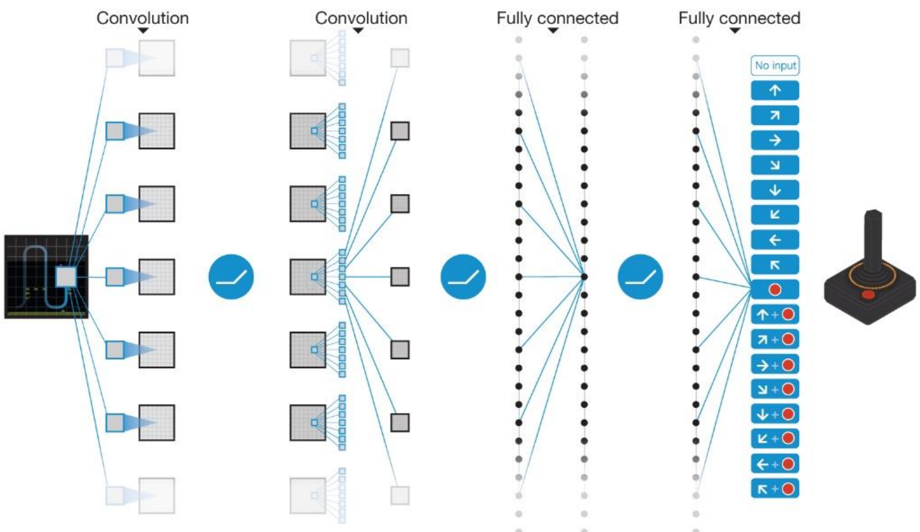


Algorithm formulation

Coding tutorials

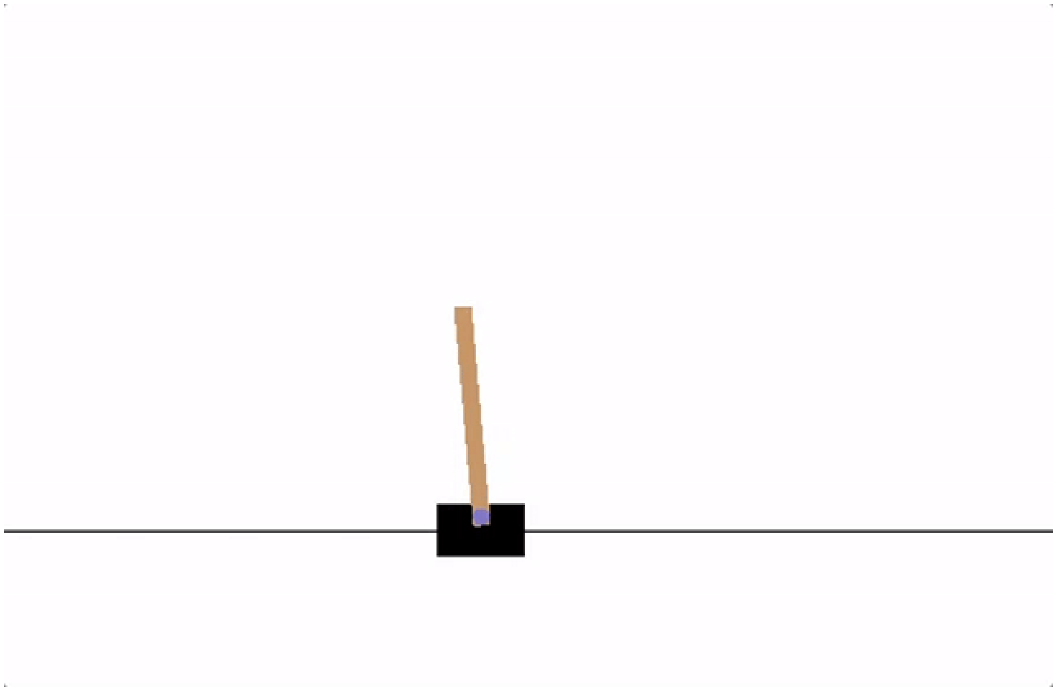
# Deep Q-Network (DQN)

- 2015 In Nature




# Deep Q-Network (DQN)

- Most famous Example of DQN is 'cartpole' game.



**4 states**

- Cart position
- Cart velocity
- Pole angle
- Pole velocity at tip

**2 Actions**

- Right moving
- Left moving

# Deep Q-Network (DQN)

- There are two good tutorials you can study how to apply this DQN algorithm in this cartpole environment.

1. Tutorial provided by Pytorch official website
   https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

   this tutorial use OpenAI Gym environment

```
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.functional as F
import torchvision.transforms as T


env = gym.make('CartPole-v0').unwrapped

# set up matplotlib
is_ipython = 'inline' in matplotlib.get_backend()
if is_ipython:
    from IPython import display
```

# Deep Q-Network (DQN)

- There are two good tutorials you can study how to apply this DQN algorithm in this cartpole environment.

1. Tutorial provided by Pytorch official website
   [https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html](https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html)

   And you can learn how to DQN main concepts like 'Replay memory and Fixed target' we learned previous lecture in the code.

### Replay Memory

We'll be using experience replay memory for training our DQN. It stores the transitions that the agent observes, allowing us to reuse this data later. By sampling from it randomly, the transitions that build up a batch are decorrelated. It has been shown that this greatly stabilizes and improves the DQN training procedure.

For this, we're going to need two classses:

- `Transition` - a named tuple representing a single transition in our environment. It essentially maps (state, action) pairs to their (next_state, reward) result, with the state being the screen difference image as described later on.
- `ReplayMemory` - a cyclic buffer of bounded size that holds the transitions observed recently. It also implements a `.sample()` method for selecting a random batch of transitions for training.

```
Transition = namedtuple('Transition',
                        ('state', 'action', 'next_state', 'reward'))
```

```
class ReplayMemory(object):

    def __init__(self, capacity):
        self.capacity = capacity
        self.memory = []
        self.position = 0

    def push(self, *args):
        """Saves a transition."""
        if len(self.memory) < self.capacity:
            self.memory.append(None)
        self.memory[self.position] = Transition(*args)
        self.position = (self.position + 1) % self.capacity

    def sample(self, batch_size):
        return random.sample(self.memory, batch_size)
```

# Deep Q-Network (DQN)

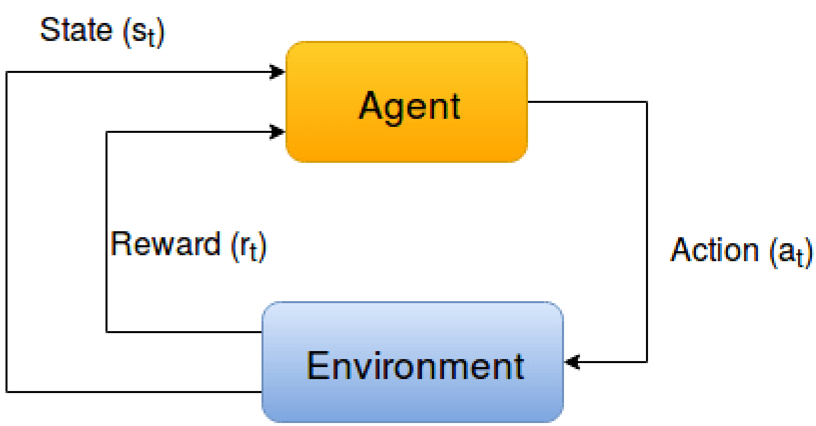
- There are two good tutorials you can study how to apply this DQN algorithm in this cartpole environment.

2. Tutorial provided by python lesson youtube channel
   https://youtu.be/D795oNqa-Vk?t=1

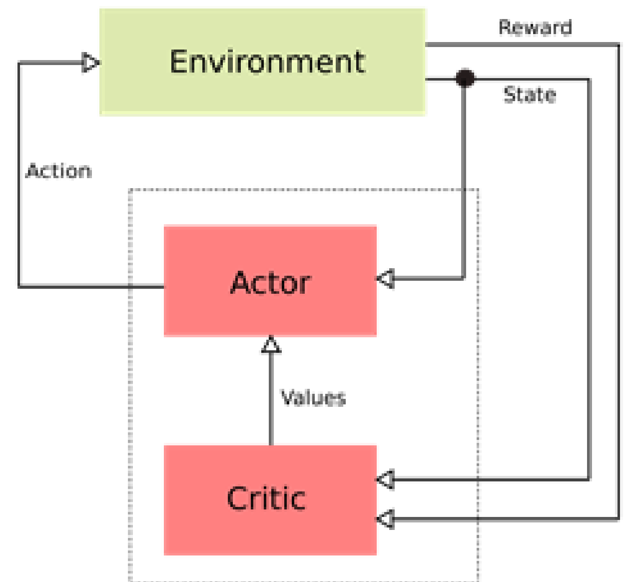
   This channel also well explain about applying DQN algorithm in cartpole game step by step. (about 30 min clip)

# Actor Critic Algorithm (2000)

- The actor critic algorithm takes policy-based and value-based methods together by having separate network approximations for value (critic) and action (actor). These two networks work together to normalize each other and achieve more stable results.
- The key to this algorithm is the idea that there are two different models separated from each other to create a control policy.



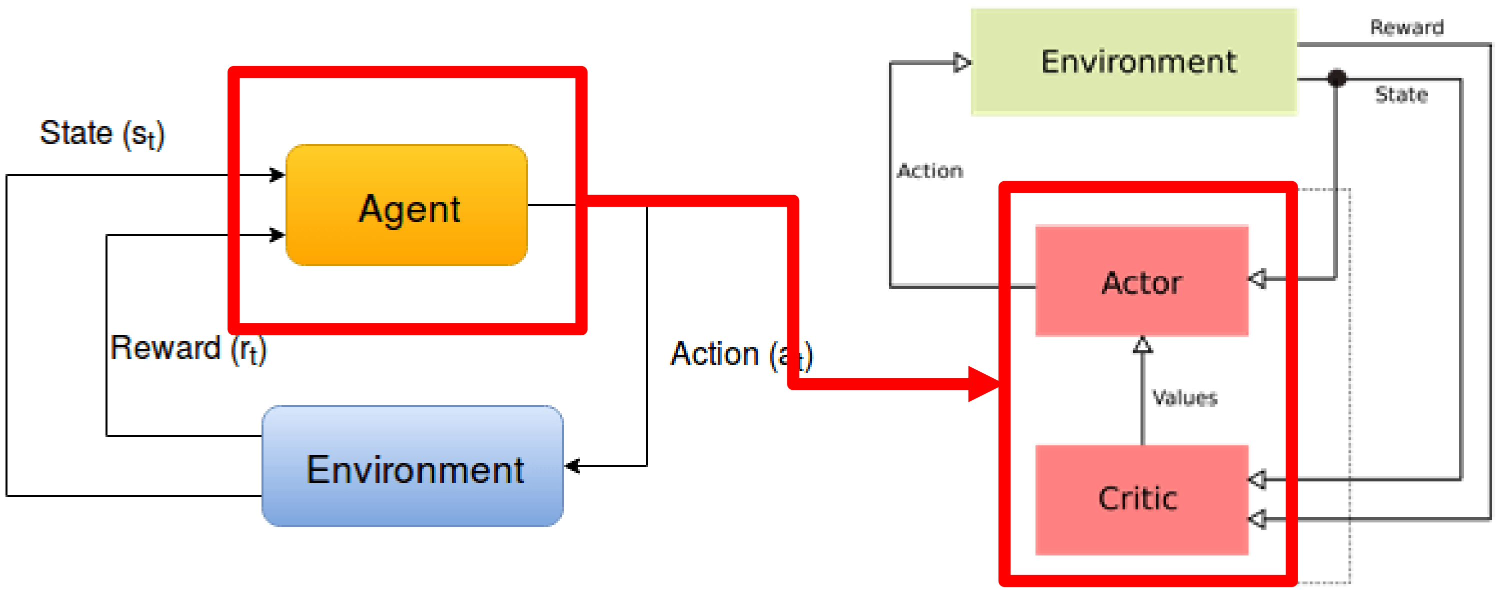

Previous architecture




Actor-Critic architecture

# Actor Critic Algorithm(2000)

- The actor critic algorithm takes policy-based and value-based methods together by having separate network approximations for value(critic) and action(actor). These two networks work together to normalize each other and achieve more stable results.
- The key to this algorithm is the idea that there are two different models separated from each other to create a control policy.




Previous architecture

Actor-Critic architecture

# Actor Critic Algorithm (2000)

- Actor Critic Algorithm
  - Advantage Actor Critic (A2C)
  - Asynchronous Advantage Actor Critic (A3C)
  - Trust region policy optimization (TRPO)
  - Deep Deterministic Policy Gradient (DDPG)
  - Proximal policy optimization (PPO)

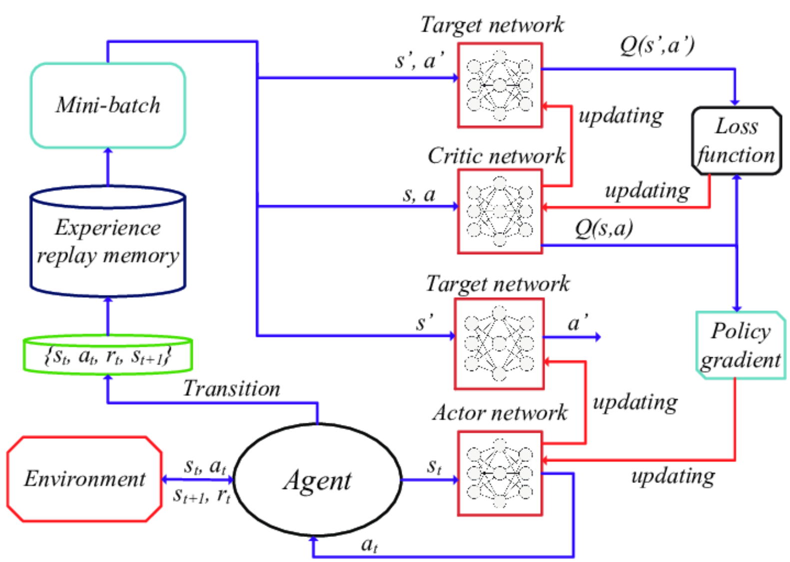
# Actor Critic Algorithm (2000)

- Actor Critic Algorithm
  - Advantage Actor Critic (A2C)
  - Asynchronous Advantage Actor Critic (A3C)
  - Trust region policy optimization (TRPO)
  - Deep Deterministic Policy Gradient (DDPG)
  - Proximal policy optimization (PPO)

# Deep Deterministic Policy Gradient (DDPG) (2016)

- DDPG Algorithm, an algorithm published in 2016, is an algorithm that combines the improvements of Q-learning and the policy gradation update rule to apply q-learning to multiple continuous control environments.

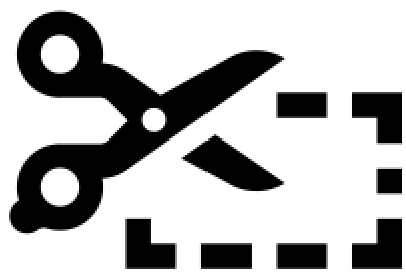
- You can refer to this algorithm in this link
- [https://youtu.be/6Yd5WnYls_Y?t=1](https://youtu.be/6Yd5WnYls_Y?t=1)



# Proximal Policy Optimization(PPO)

- The main idea of PPO algorithm is clipping calculation range to get trust region and avoid calculation burden during model training.  And this algorithm show good performance to build model.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\Big[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\Big]$$



# Proximal Policy Optimization (PPO)

- The main idea of PPO algorithm is clipping calculation range to get trust region and avoid calculation burden during model training. And this algorithm show good performance to build model.

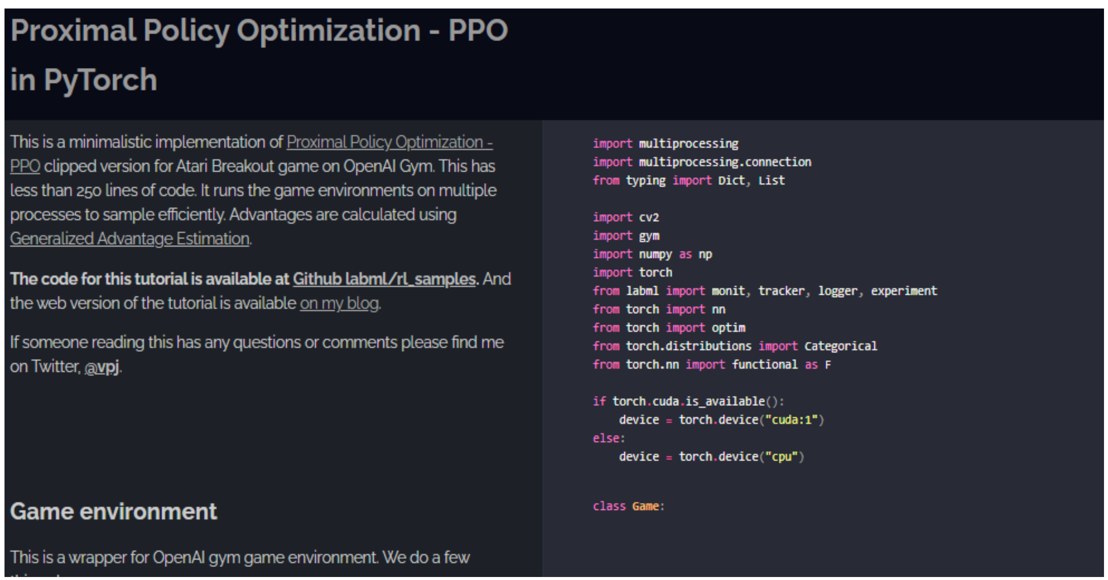
- You can refer to this algorithm a link down below.
  (there are bundle of video clips part1 ~ 5)
- https://youtu.be/SWllbdcrKLI

# Proximal Policy Optimization (PPO)

- The main idea of PPO algorithm is clipping calculation range to get trust region and avoid calculation burden during model training. And this algorithm show good performance to build model.

- You can also refer to this site which show PPO algorithm code process detaily step by step.
- This site will help you understand the steps in the RL code architecture.
- http://blog.varunajayasiri.com/ml/ppo_pytorch.html

Explain



Proximal Policy Optimization - PPO in PyTorch

This is a minimalistic implementation of Proximal Policy Optimization - PPO clipped version for Atari Breakout game on OpenAI Gym. This has less than 250 lines of code. It runs the game environments on multiple processes to sample efficiently. Advantages are calculated using Generalized Advantage Estimation.

**The code for this tutorial is available at Github labml/rl_samples.** And the web version of the tutorial is available on my blog.

If someone reading this has any questions or comments please find me on Twitter, @vpj.

**Game environment**

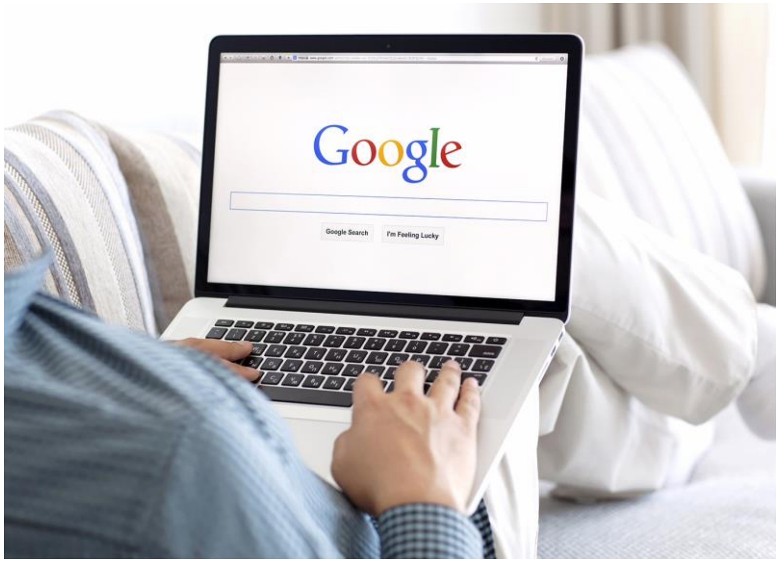
This is a wrapper for OpenAI gym game environment. We do a few

```
import multiprocessing
import multiprocessing.connection
from typing import Dict, List

import cv2
import gym
import numpy as np
import torch
from labml import monit, tracker, logger, experiment
from torch import nn
from torch import optim
from torch.distributions import Categorical
from torch.nn import functional as F

if torch.cuda.is_available():
    device = torch.device("cuda:1")
else:
    device = torch.device("cpu")


class Game:
```

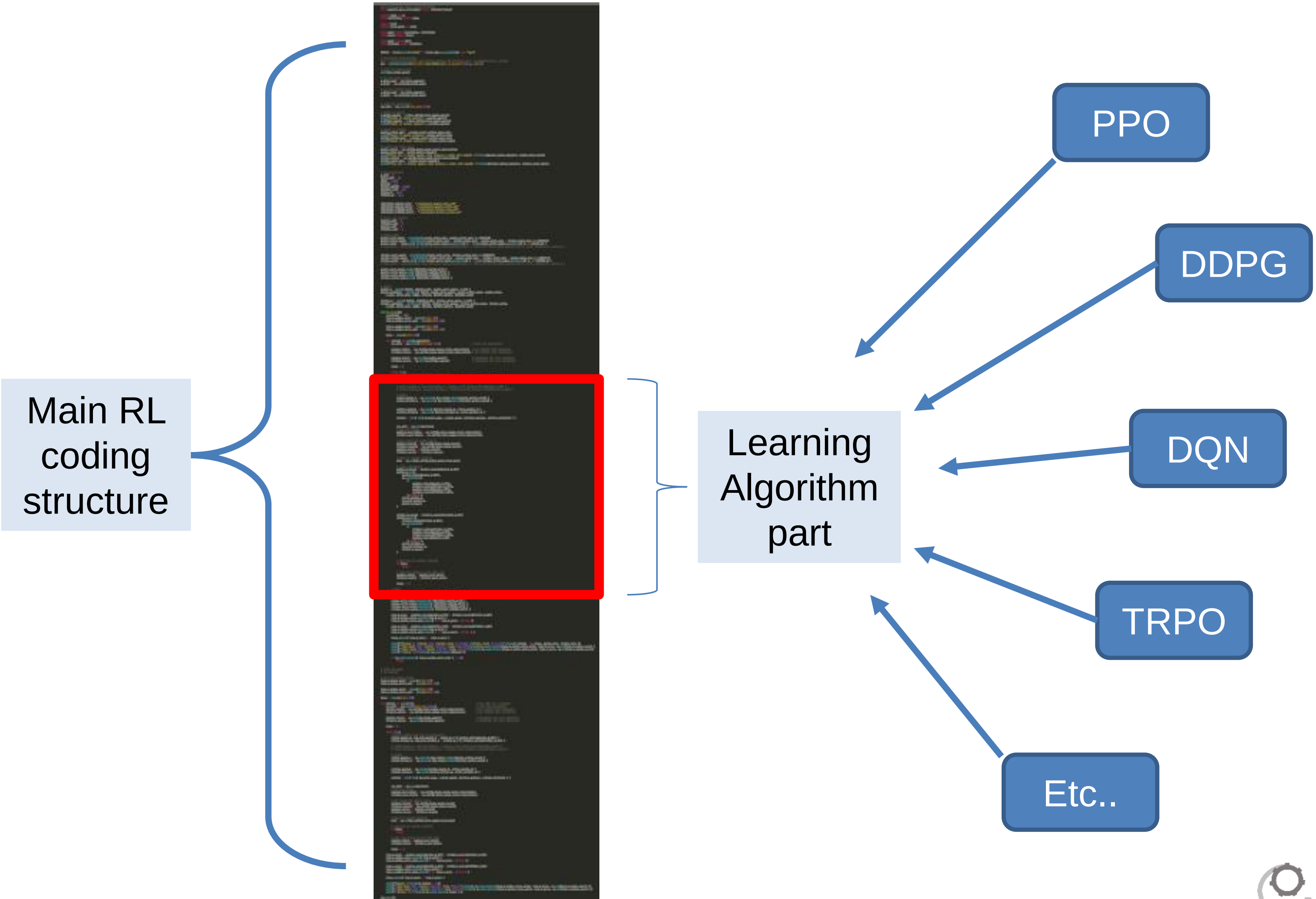
Coding part

# Project Strategy

- There are many other ways or algorithms to improve your soccer agents
- Some algorithms and tutorials introduced from this slide is not mandatory
- This lecture's purpose is to help each team gets the right initial direction for this project
- It is recommended that you frequently refer to Googling and GitHub when searching for a lot of materials as you progress through the assignment.



# Project Strategy



# About Mentoring Session

| | | | | |
|---|---|---|---|---|
| 11/03 - 11/05 | Lab6. RL basic for robotics & Final project | Lab7. RL basic for robotics 2 | | *Midterm Project (Fri)* |
| 11/10 - 11/12 | Lab8. Deep reinforcement learning for robotics 1 | Lab9. Deep reinforcement learning for robotics 2 | *11/13 review 4* | |
| 11/17 - 11/19 | Mentoring Session | Team Discussion & Q&A | | |
| 11/24 - 11/26 | Mentoring Session | Team Discussion & Q&A | *11/27 review 5* | |
| 12/01 - 12/03 | Mentoring Session | Final Competition | | *Final Project Codes (Wed)* |
| 12/08 – 12/10 | Final project presentation and evaluation | | *12/11 review 6* | *Final Presentation Video (Mon)* |
| 12/15 – 12/17 | *No class (Final exam period)* | | | *Final Report (Fri)* |

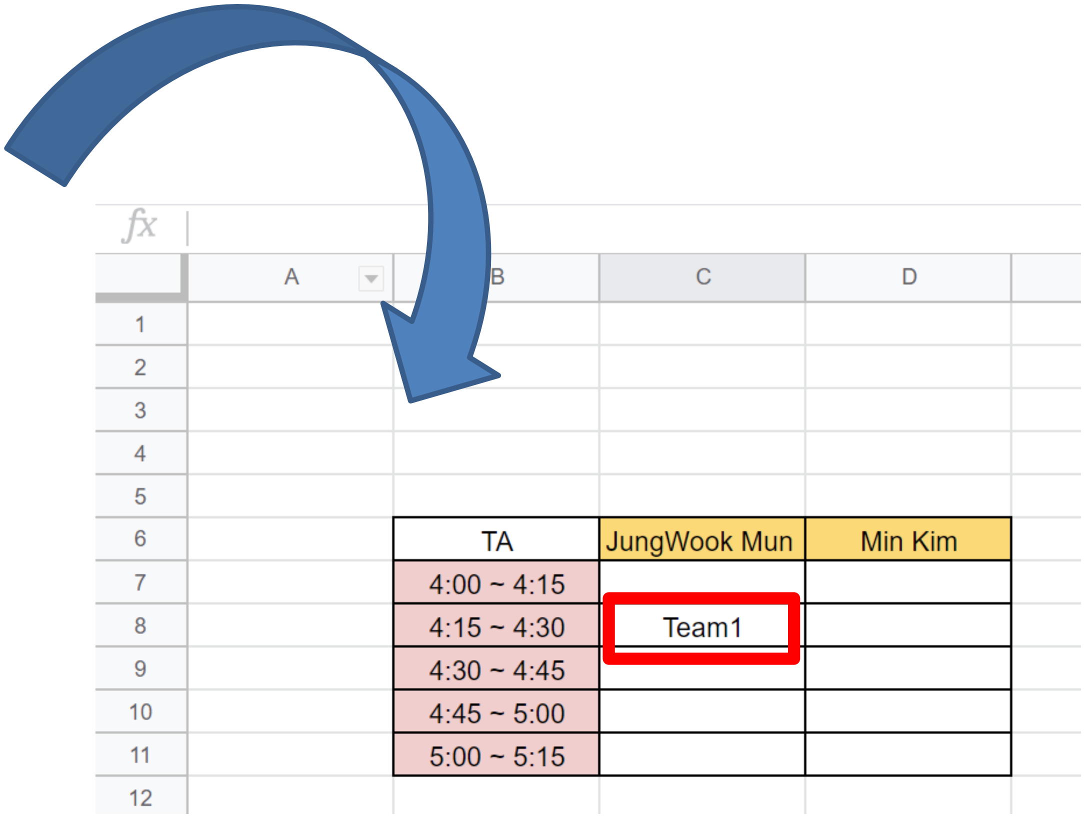
# How to book mentoring schedule.

Google Sheets

| TA | JungWook Mun | Min Kim |
|---|---|---|
| 4:00 ~ 4:15 | | |
| 4:15 ~ 4:30 | | |
| 4:30 ~ 4:45 | | |
| 4:45 ~ 5:00 | | |
| 5:00 ~ 5:15 | | |

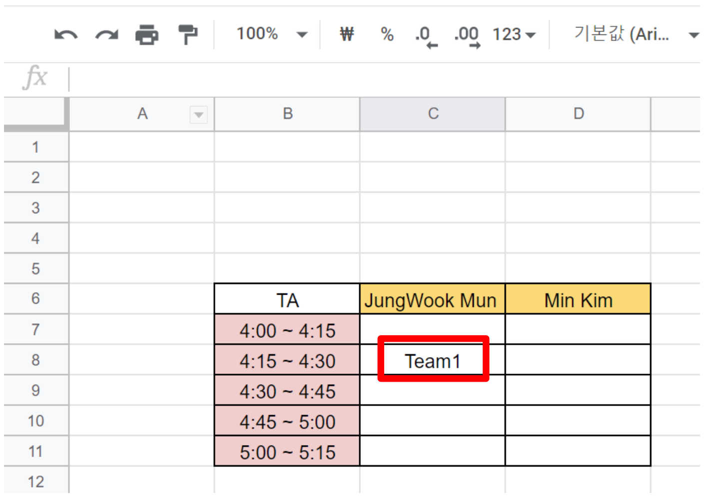
# How to book mentoring schedule.

Google Sheets

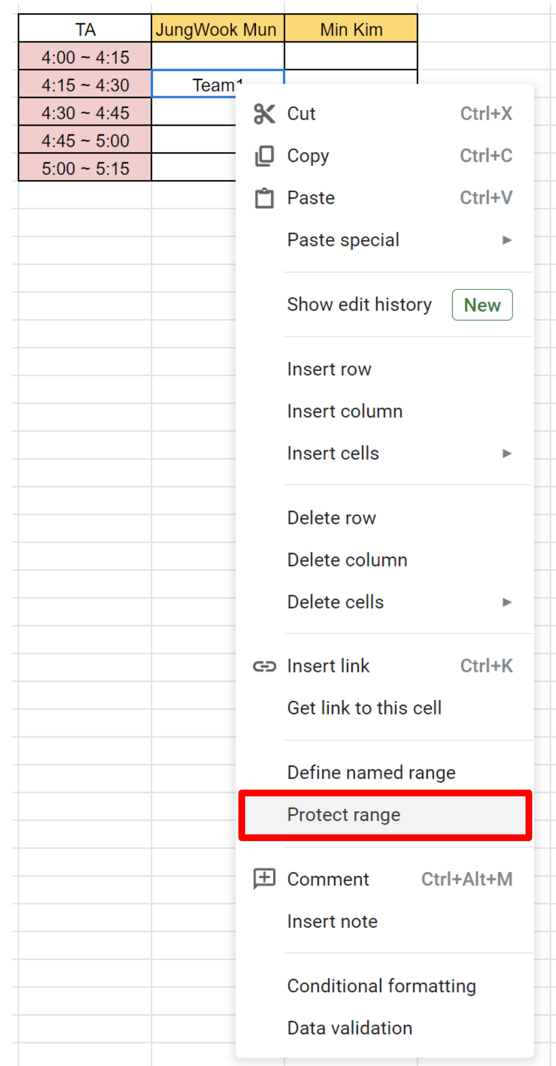


| | A | B | C | D |
|---|---|---|---|---|
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | TA | JungWook Mun | Min Kim |
| 7 | | 4:00 ~ 4:15 | | |
| 8 | | 4:15 ~ 4:30 | Team1 | |
| 9 | | 4:30 ~ 4:45 | | |
| 10 | | 4:45 ~ 5:00 | | |
| 11 | | 5:00 ~ 5:15 | | |
| 12 | | | | |

# How to book mentoring schedule.




**Right Click**



- If you have any questions, contact Email to TA with the following address
  - munjw777@kaist.ac.kr
  - m-kim@kaist.ac.kr

# Thank you