Robot Learning and Control 4

Deep Reinforcement Learning

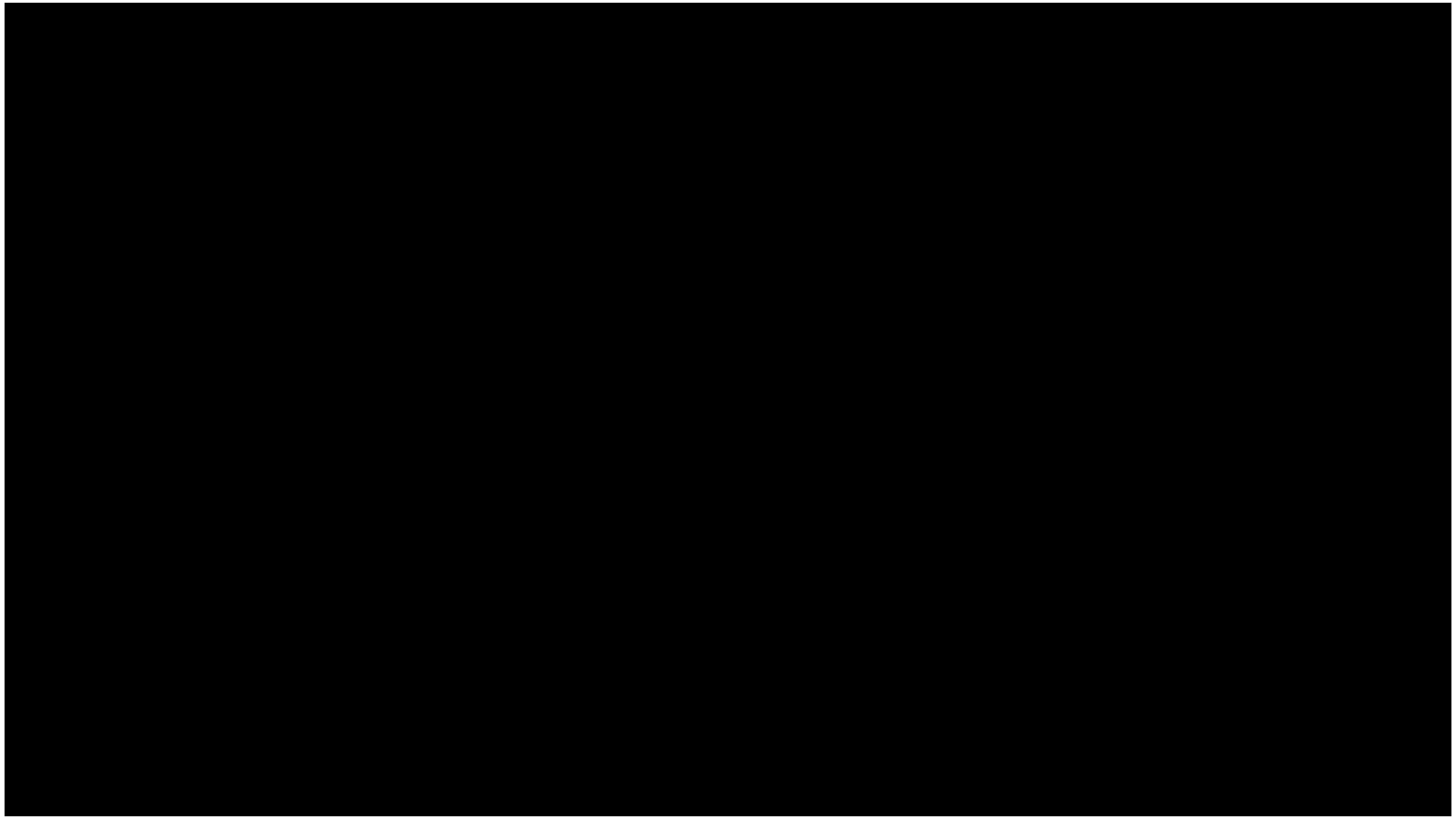
Yoshihisa Tsurumine[1]

[1] Robot Learning Lab

Nov. 29th 2024

Today's lecture

Goal: Understand how deep reinforcement learning learns robot policies



E.g., soccer policy learned by deep reinforcement learning

[Haarnoja et al. , 2023]

Contents

1. Reinforcement Learning Problem Settings and the Concept of the Learning Process

2. Deep Reinforcement Learning: Challenges and Solutions in Neural Network Integration

Contents

1. Reinforcement Learning Problem Settings and the Concept of the Learning Process

2. Deep Reinforcement Learning: Challenges and Solutions in Neural Network Integration

Objective and Contents of First Section

- Objective

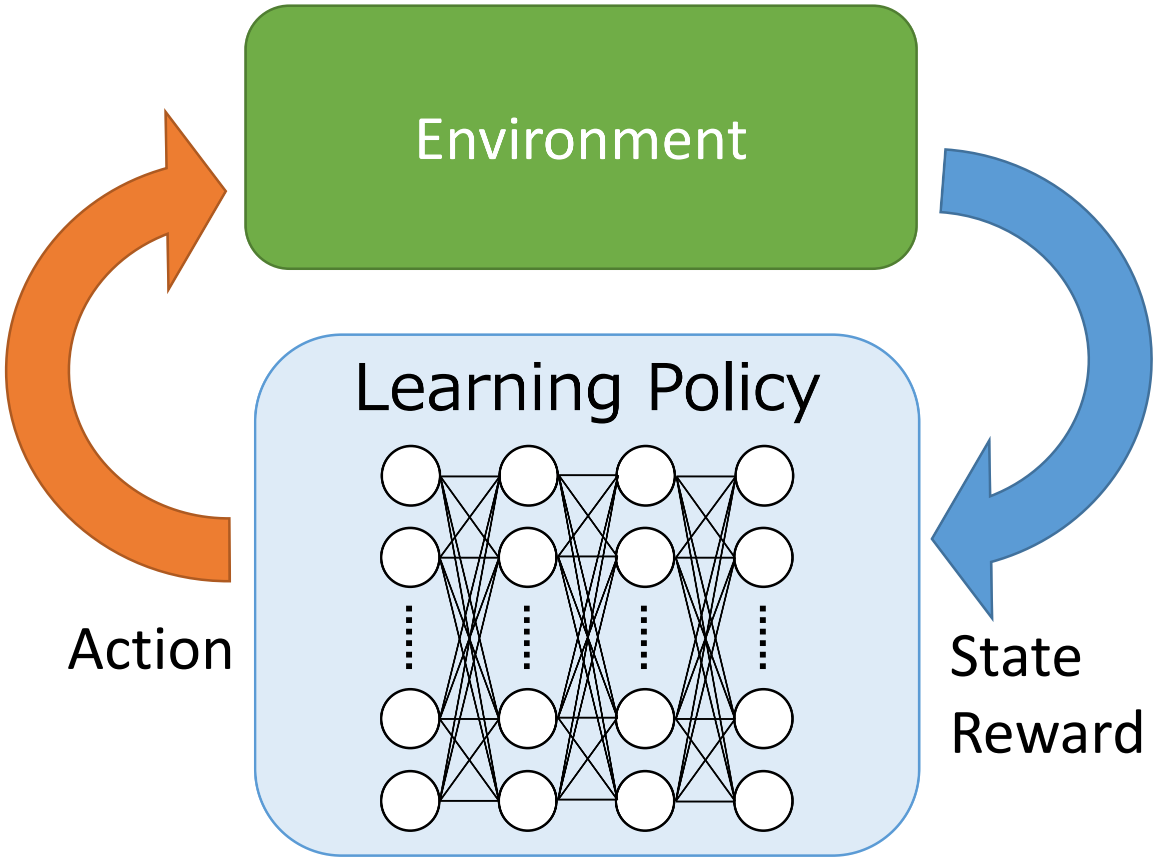
Understanding Reinforcement Learning problem settings and the concept of the learning process

- Contents

1. Overview of Reinforcement Learning
2. Problem settings
3. Concept of the learning process

Overview: Reinforcement Learning

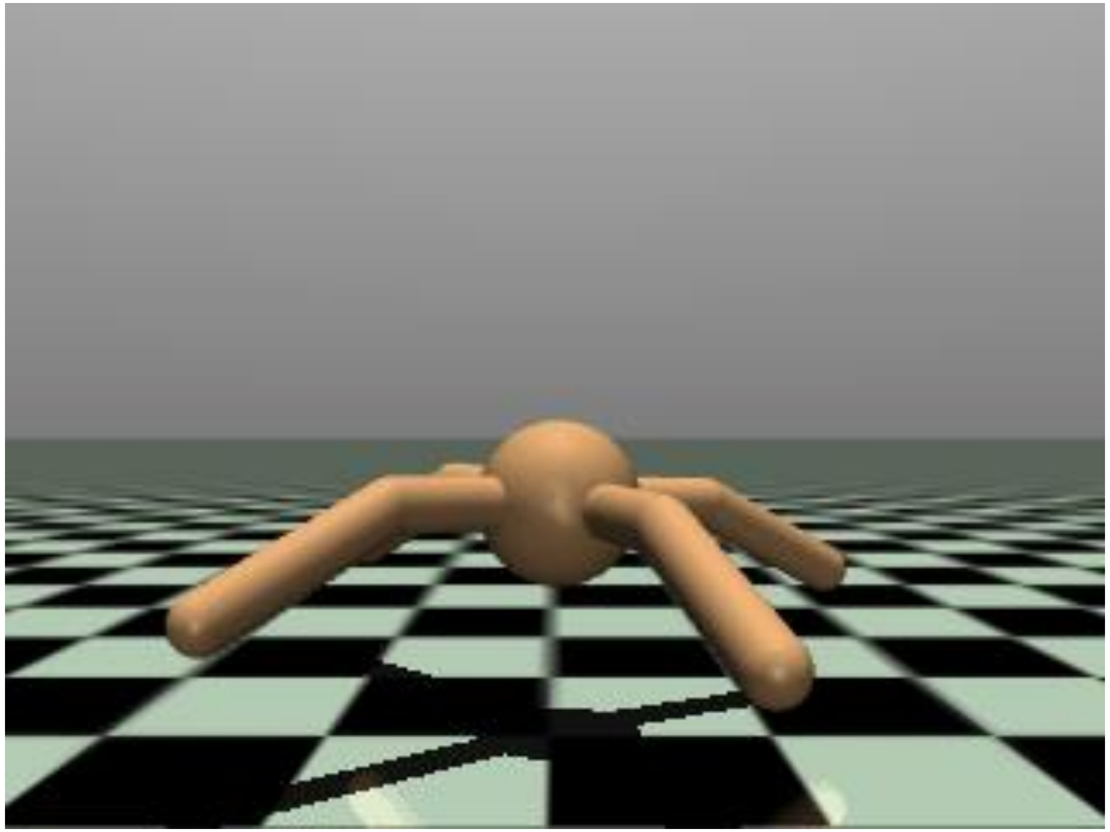
Learning behaviors that maximize total reward from trial and error



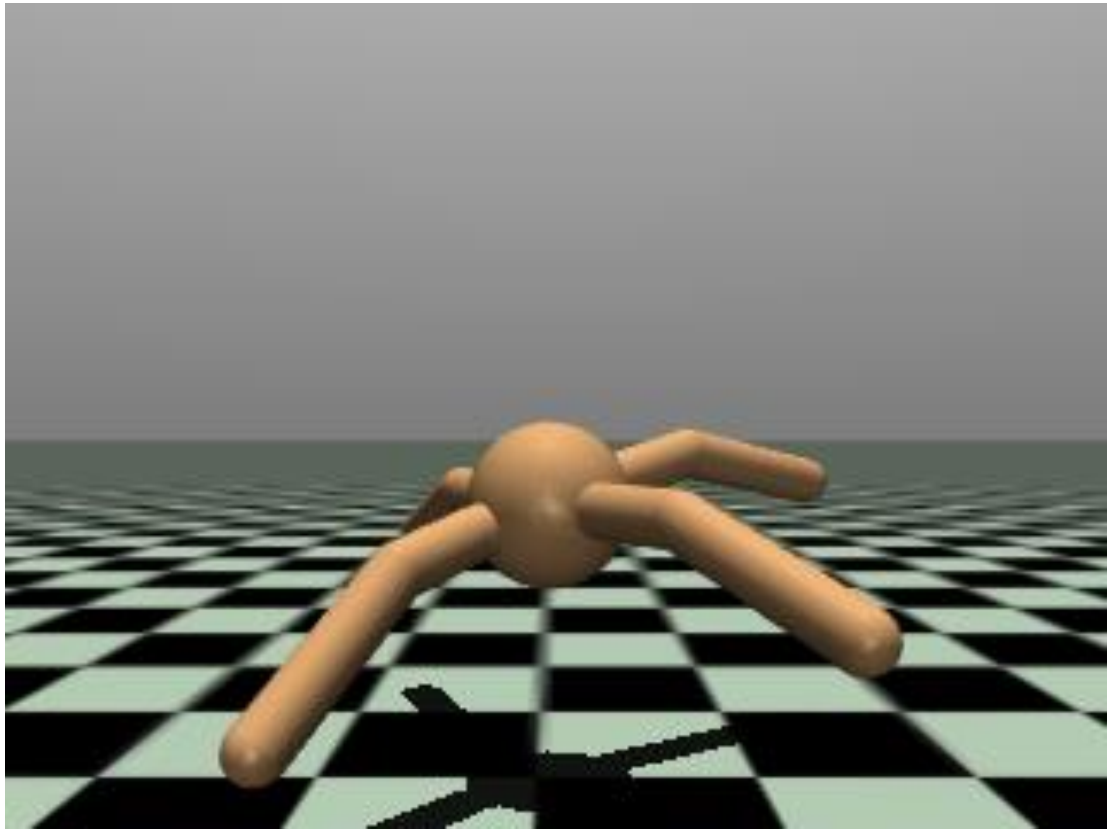
Overview: Reinforcement Learning

Example: Learning to walk on four legs

Trial and error during learning



After Learning Movement



Reinforcement Learning gradually improves policy from random action

Examples of DRL-based robot learning

Grasp learning with multiple robots

[S. Levine et al.,2016]

Learning Dexterous In-Hand Manipulation

[M. Andrychowicz et al.,2018]

Quadrupedal Locomotion over Challenging Terrain

[J. Lee et al.,2020]

Learning complex behavioral rules with high-dimensional input and output

Examples of DRL Applications other than in Robotics

Won over a professional Go player

[Z.Wang et al.,2016]

40% reduction in power used for cooling in data centers

[R. Evans et al.,2016]

DRLs are used to suppress GPT-3 halmination

[TB. Brownet al.,2020]

Task Scale and Performance are Increasing through the Integration with DNN

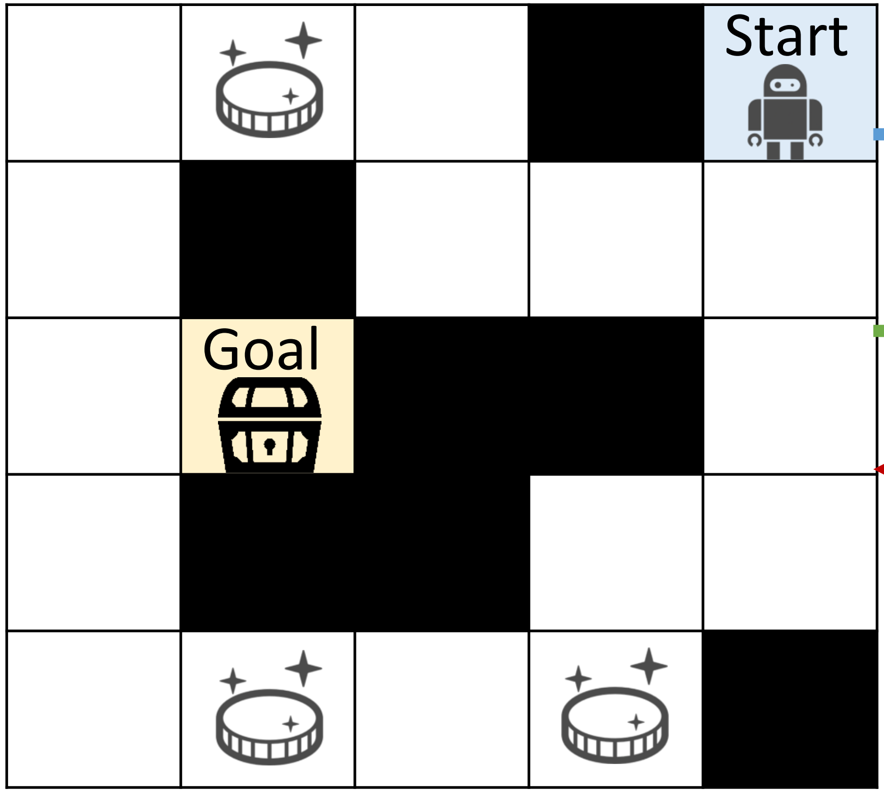
Reinforcement Learning Problem Setting and Objective

- RL objective

Learning a policy (a function that inputs states and outputs actions) to maximize total rewards through trial-and-error

- Examples of problem setting: grid world, Q-learning

Environment



State s:

grid coordinate

Reward $r(s, a, s')$:

goal, coin

Action a :

Up, down, left, right movement

Agent

Approximation of total reward (value function)

$$Q(s, a) = \mathbb{E}_{\pi, \mathcal{T}_{ss'}^{a}} \left[\sum_{t=0}^{\infty} \gamma^{t} r(s, a, s') \right]$$

Policy for Learning Objective

$$\pi(a|s) = \underset{a'}{\operatorname{argmax}}\, Q(s, a')$$

Concept of Learning Process: Outline

- Gradually learning a policy to maximize total rewards by repeating two steps

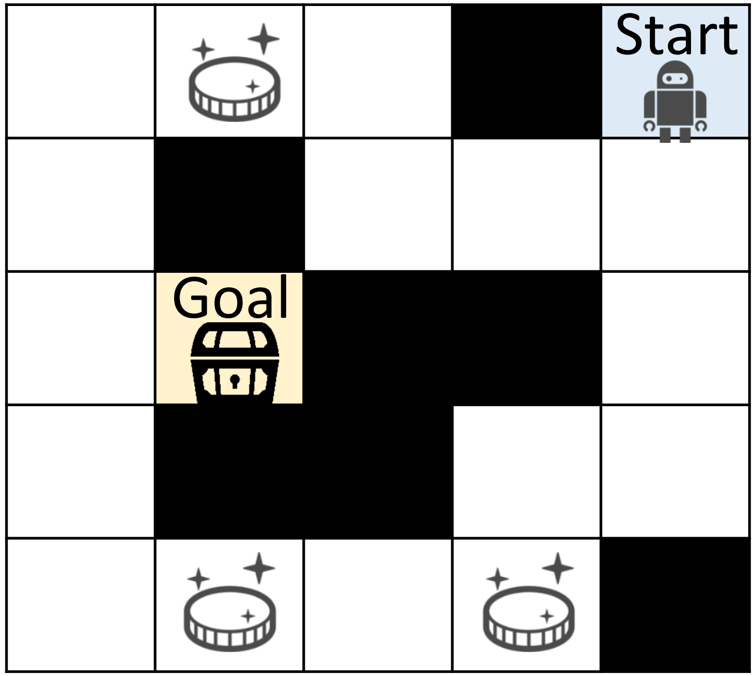
S1. Collecting data (state, action, reward, next state) through trial-and-error using the learning policy

S2. Update policy to maximize total rewards using collected data

After explaining how the trajectory of the learning policy changes,
I will explain the details of each step

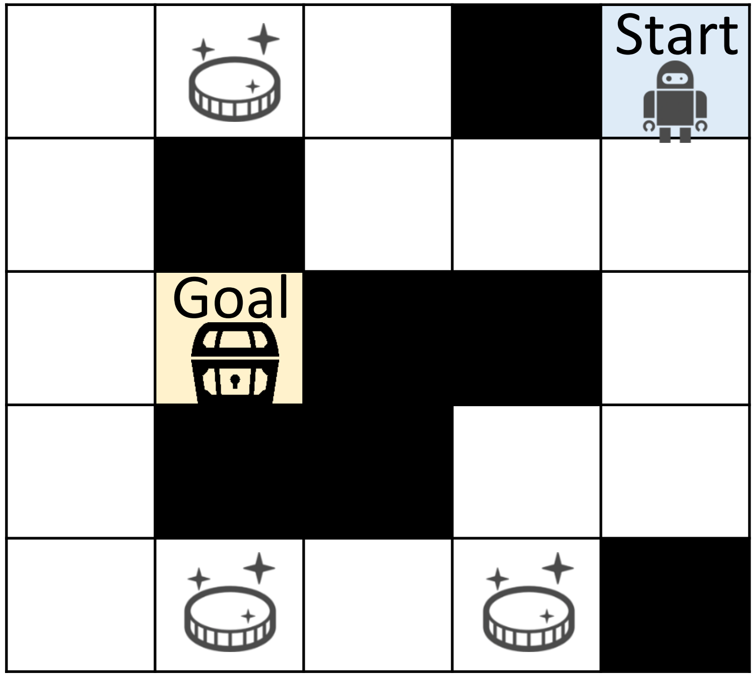
Change in policy trajectory due to repeated updates

1. Early



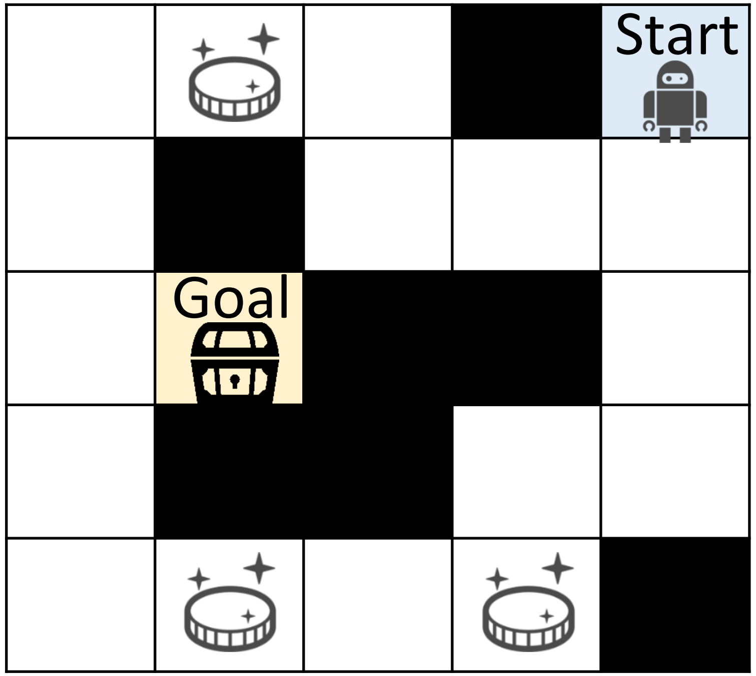
Random exploration

2. Middle



Floundering rewarding conditions
Agents may reach a goal by accident

3. Final

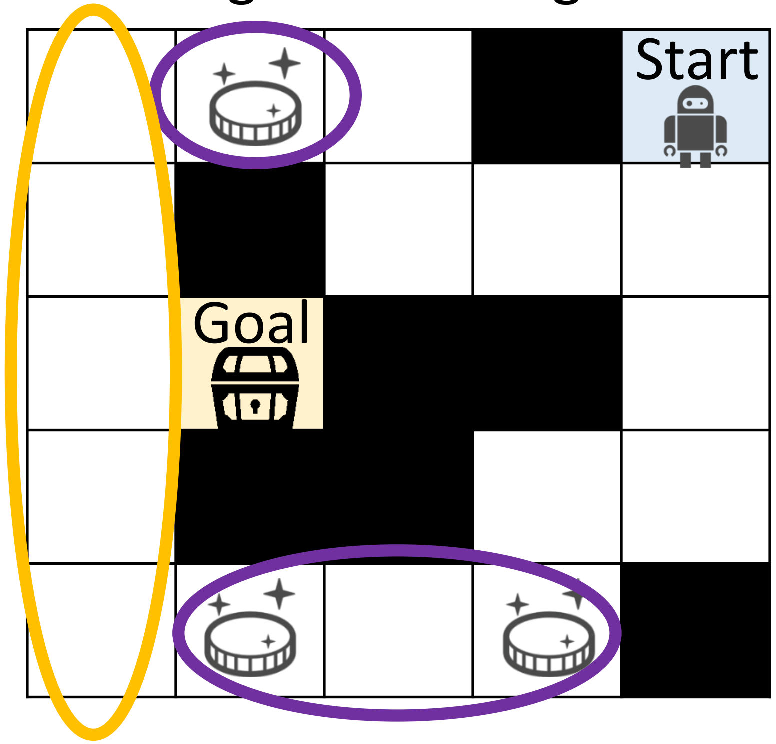


Aim for the goal state in the shortest path

Gradually increasing total rewards through trial-and-error to approach the optimal policy

S1. Collect data using learning policy

Example of middle stage of learning



- Collecting data

The following pairs of data are collected for each policy decision
state s, action a, next state s', reward $r(\mathrm{s}, \mathrm{a}, \mathrm{s}')$

- Trade-off between exploration and exploitation in policy

Exploitation:

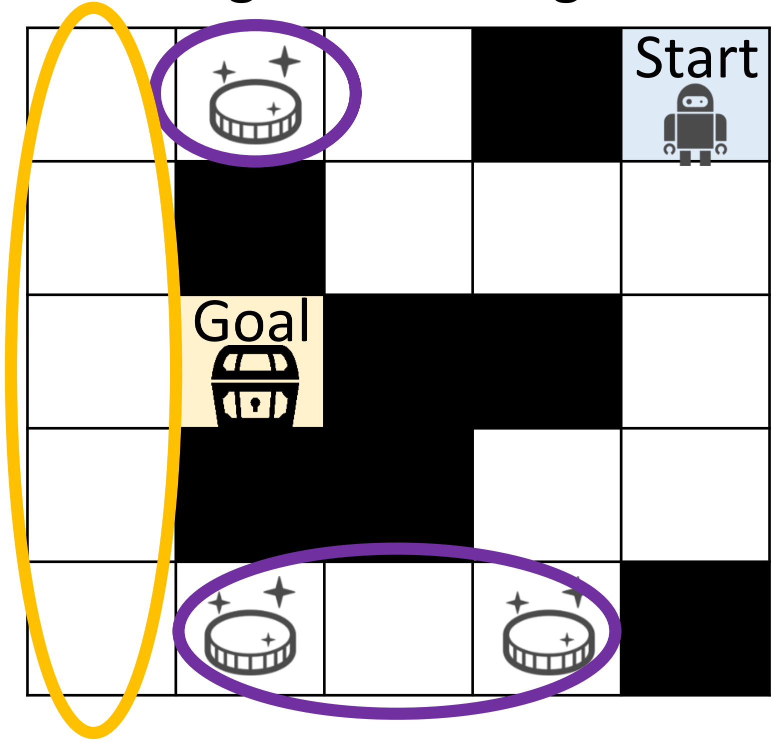
Decide on actions with high total reward based on experience
Actions approaching coin coordinates known from experience

Exploration:

Try less experienced behaviors and explore new rewards
Explore away from coins. Reached goal state by accident

S1. Collect data using learning policy

Example of middle stage of learning



- e.g., ε-greedy policy in Q-learning

Behaviors with high total reward are checked by reference to the value function $Q(s, a)$ regressed on the total reward

Selecting the action with the highest value $Q(s, a)$ with a probability of ε
Random action selected with a probability of 1-ε

Exploitation

$$\pi(a|s) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A|} & \left(a = \underset{a'}{\operatorname{argmax}}\, Q(s, a')\right) \\ \dfrac{\varepsilon}{|A|} & (otherwise) \end{cases}$$

Exploration

S2. Update policy using collected data

- Update policy with Q-learning

Q-learning approximates total reward as a value function $Q(s, a)$, Learning a policy $\pi(a|s)$ to maximize total reward using the value function $Q(s, a)$

Approximate the expected total reward for selecting action a in state s as the value function $Q(s, a)$

Since the policy decides on the action based on the value function, Indirectly updating the policy through updating the value function

Value function

$$\pi(a|s) = \begin{cases} 1 - \varepsilon + \dfrac{\varepsilon}{|A|} & \left(a = \underset{a'}{\operatorname{argmax}} \boxed{Q(s, a')}\right) \\ \dfrac{\varepsilon}{|A|} & (otherwise) \end{cases}$$

S2. Update policy using collected data

- Update value function

Update the approximation of the total reward based on the collected data

Approximate target value gradually to consider stochastic factors such as state transitions, etc.

One data point collected

Reward $r(s, a, s')$, next state s', state s, action a

Update equation

Error $\delta \coloneqq \boxed{r(s, a, s') + \gamma \max_{a'} \hat{Q}(s', a')} - \boxed{\hat{Q}(s, a)}$

Target value (to be explained later) Current value

Updated Value $\hat{Q}(s, a) \coloneqq \hat{Q}(s, a) + \alpha\delta$

Learning rate

Update value function with collected data

Concept of Learning Process: Summary

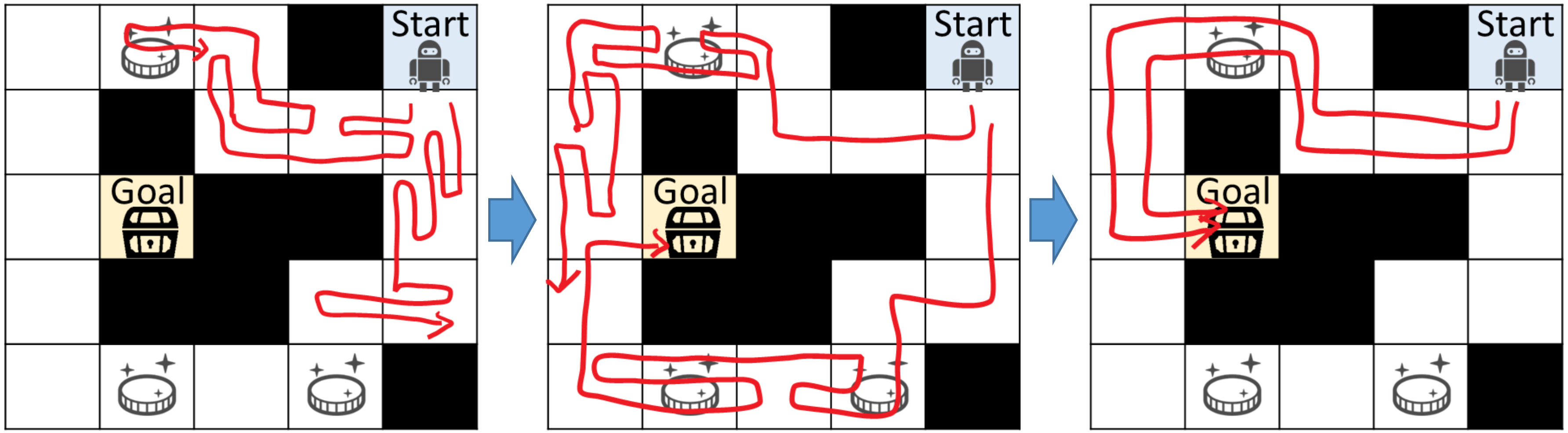
- Overview

Repeat the following two steps

S1. Collecting data through trial-and-error using the learning policy

S2. Update policy to maximize total rewards using collected data

- The concept of policy improvement during learning



Iterative process of data collection and policy updates gradually shifts from random actions towards maximizing total reward

Exercise 1

Let's actually run the Reinforcement Learning code to see how it learns and the results!

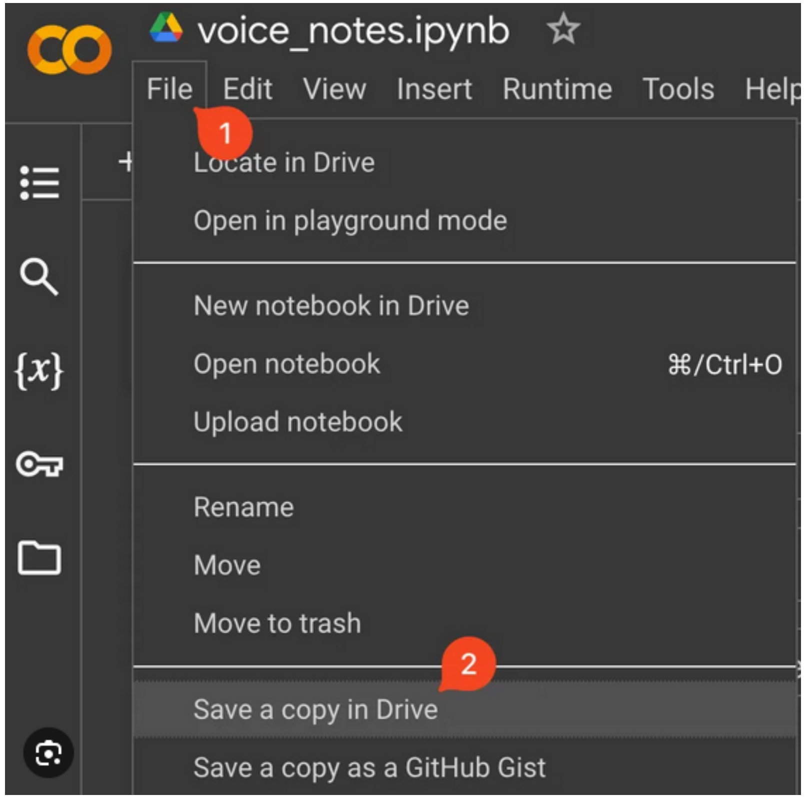
The objective is to check the correspondence between the reinforcement learning concepts described and the results of the code execution.

How to Access and Save the Practical Exercise

Access Google Colaboratory through the syllabus or the URL below

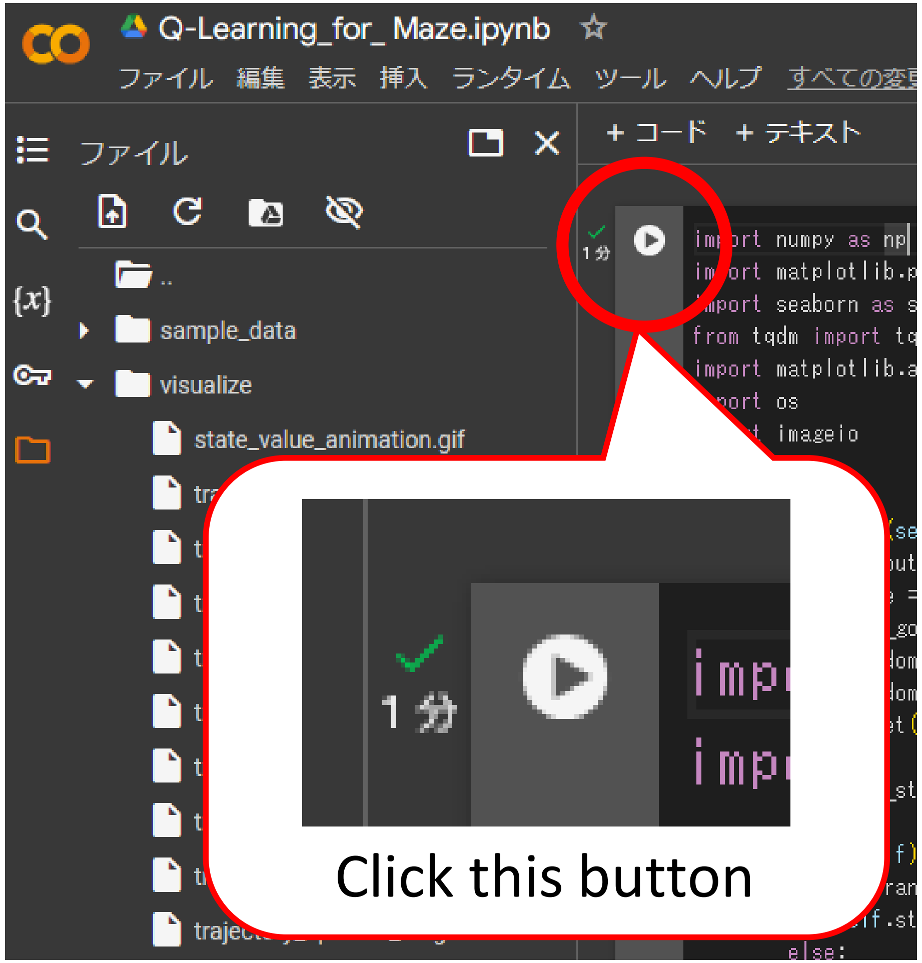
https://colab.research.google.com/drive/1VLud2mfjYeOthRVgUq8zrgRwlhDtgAk1?usp=sharing

Save to your Google Drive

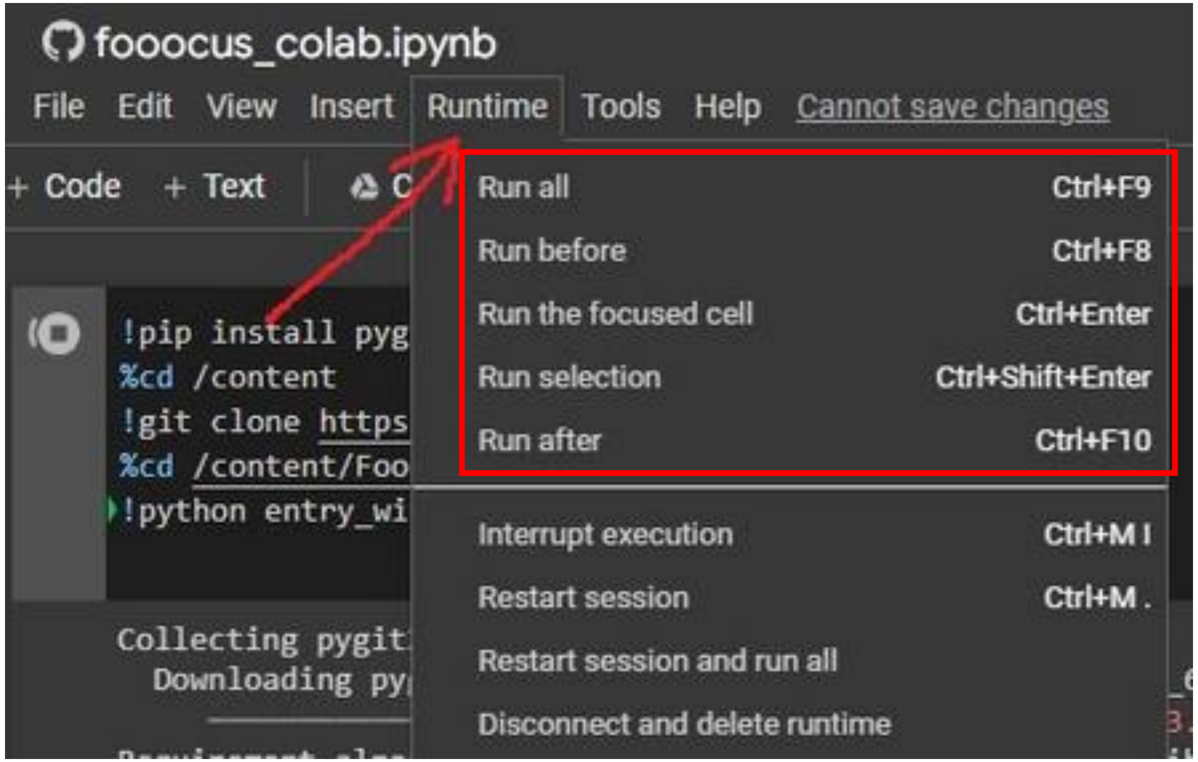


Code Execution

When executing only one code section



Other code execution



Learning Environment

Maze Task

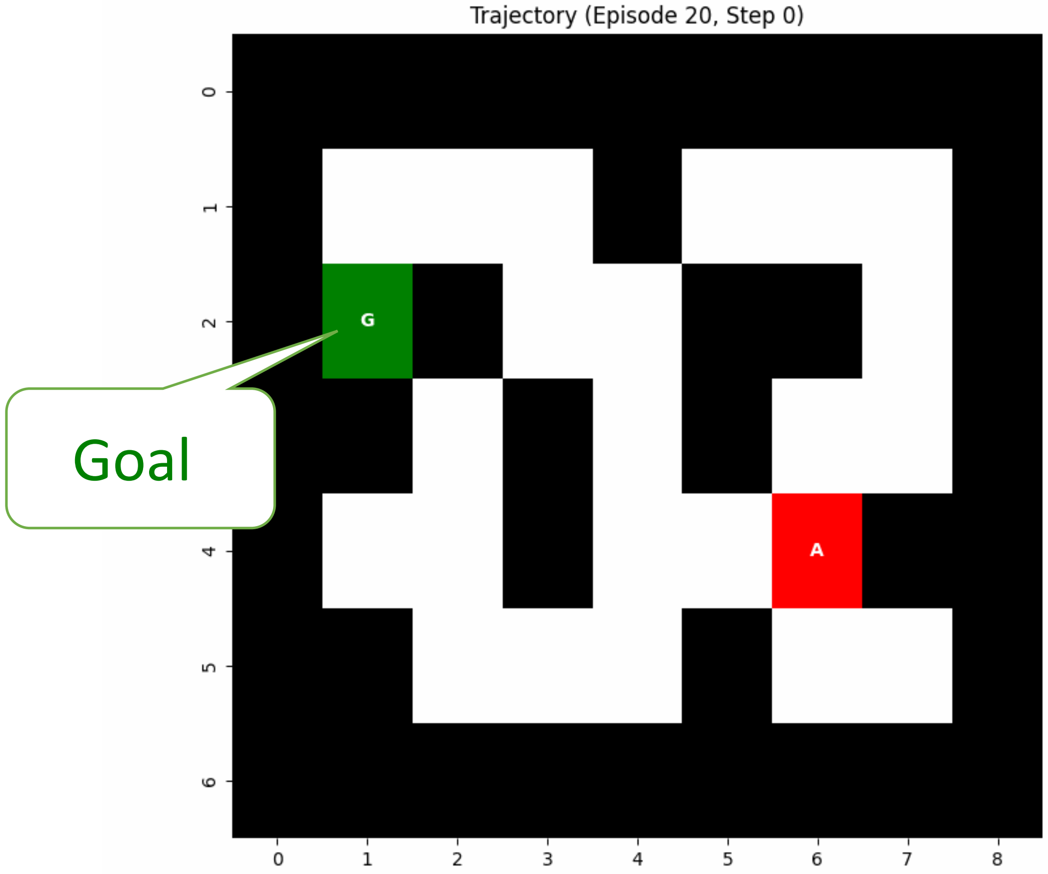
Task Objective: Learn the path from current position to goal position

State: Current coordinates, goal coordinates (4-dimensional)
Action: Up, down, left, right (4-dimensional)
Reward: +10 for reaching the goal, -0.1 otherwise

One step corresponds to one action selection

One episode ends when the trial is completed
Environment state is reset after each episode

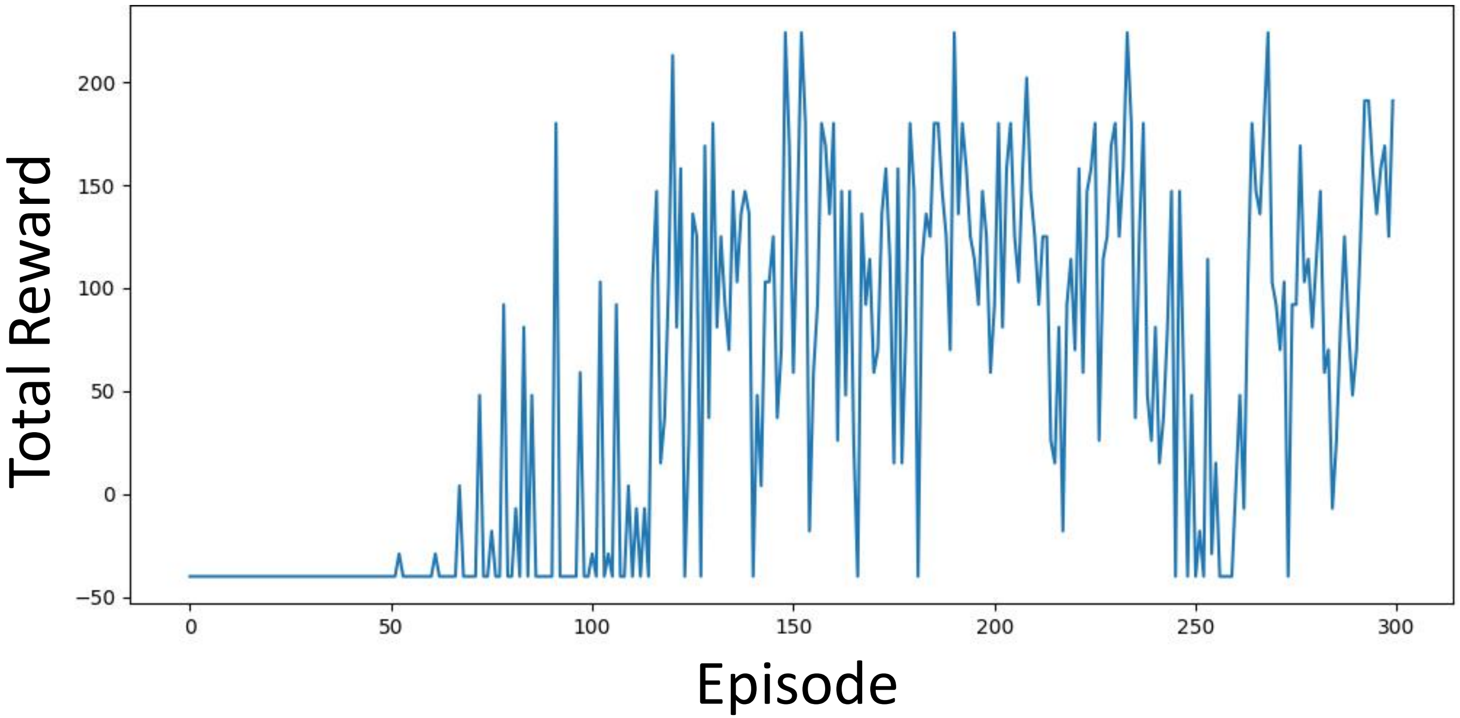
In this task, one episode consists of 40 steps



Exercise 1: Verification of Learning Curves

1. How Behavior Patterns Change from Early to Late Stages of Learning?

Learning curves: Displayed below after code execution



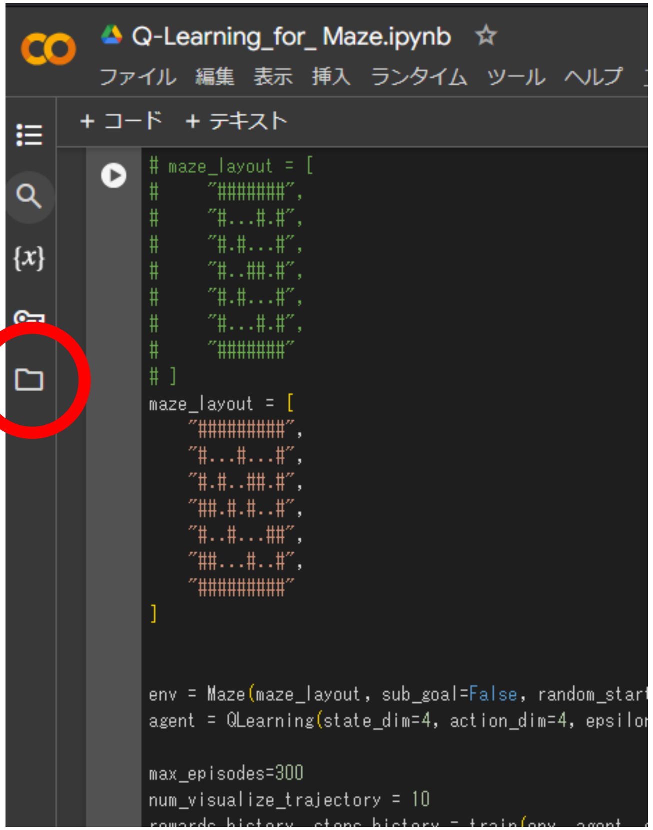
Plotting the changes in total rewards per episode

As learning progresses, total rewards (learning objective) increases

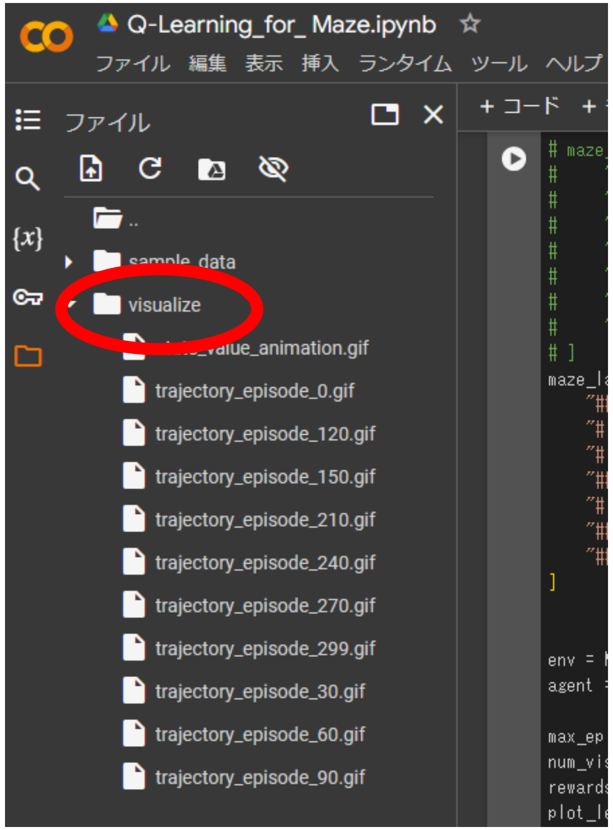
Exercise 1: Verification of Learning Trajectories

1. How Behavior Patterns Change from Early to Late Stages of Learning?

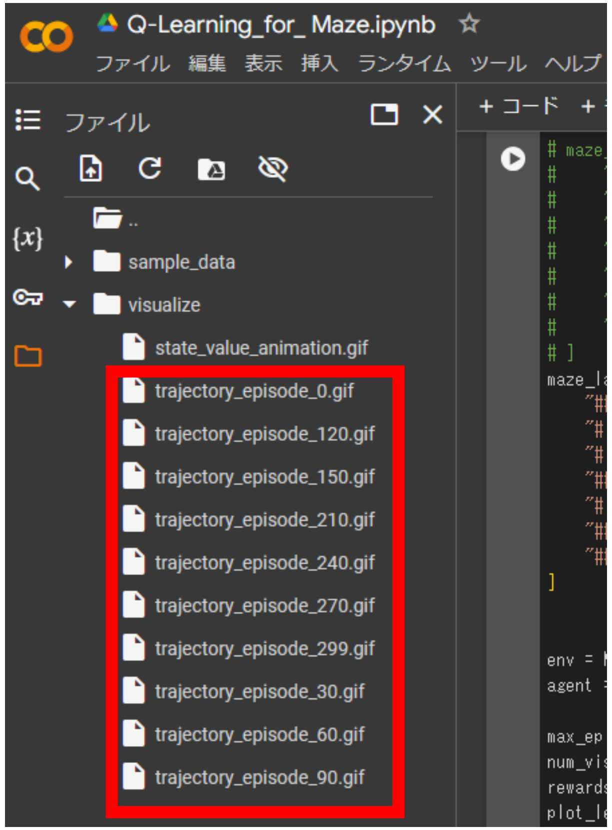
1. Open file



2. Open "visualize"



3. Open "trajectory animation"

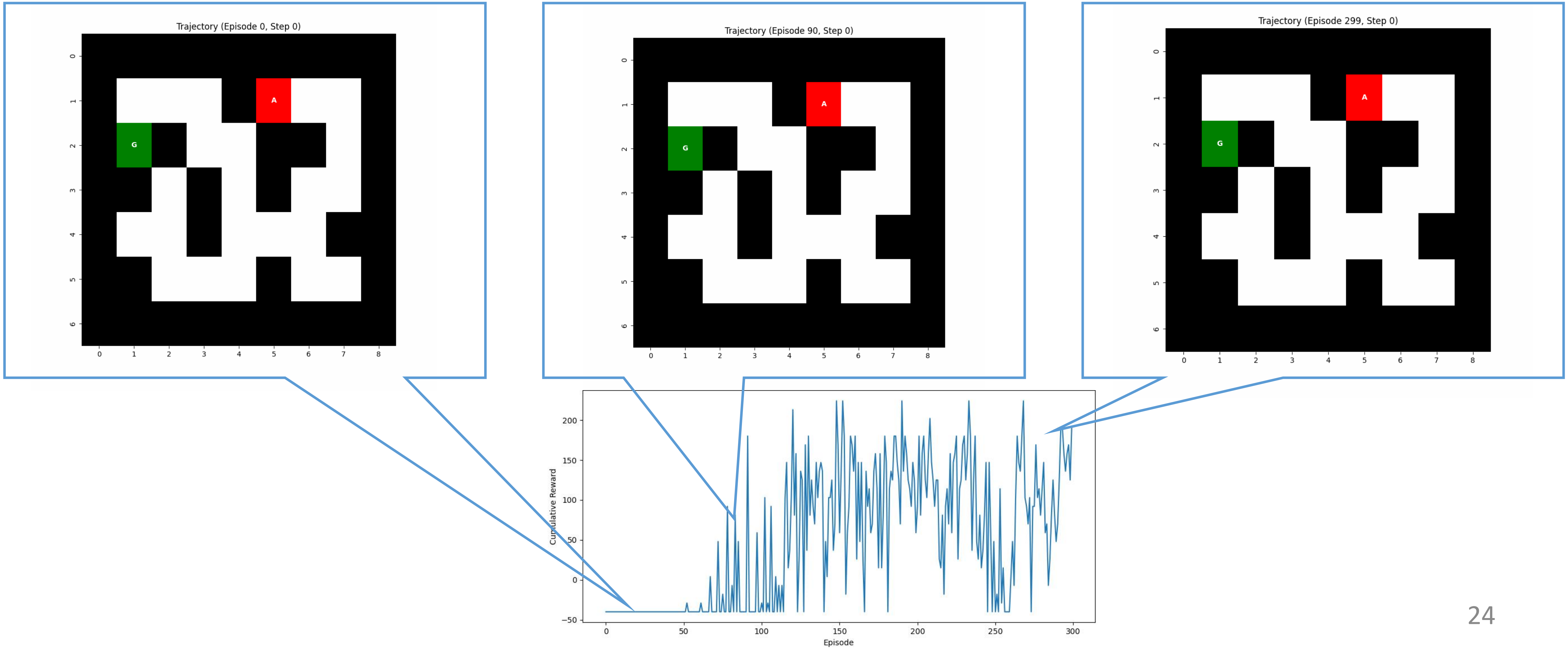


Check trajectories during learning at regular episodes

Exercise 1: Comparison of Trajectories and Curves

1. How Behavior Patterns Change from Early to Late Stages of Learning?

Let's check the correspondence between total reward and the learning trajectories



Exercise 1: Visualization of Value Function During Learning

2. How the Value Function is Updated?

Update equation for value function

Approximate Value

$$\hat{Q}(s,a) \coloneqq \hat{Q}(s,a) + \alpha\left(\boxed{r(s,a,s') + \gamma \max_{a'} \hat{Q}(s',a')} - \hat{Q}(s,a)\right)$$

Reward

Value of post-transition state s’

The value function $\hat{Q}(s,a)$ for state s and action a approximates the reward and value of next state s’

Since values are updated based on subsequent state values, states surrounding high-value states also become high in value

Let's examine how the value function values actually change

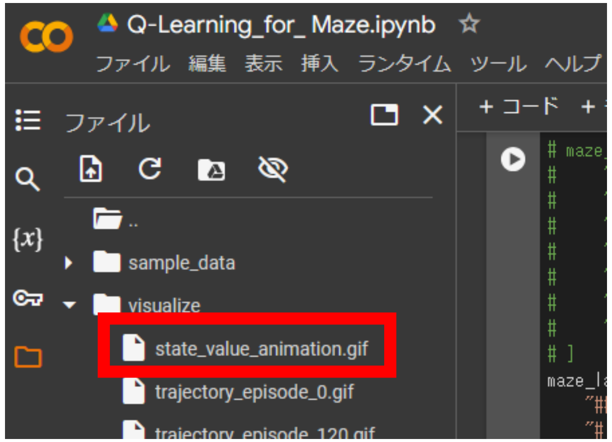
Exercise 1: Visualization of Value Function During Learning

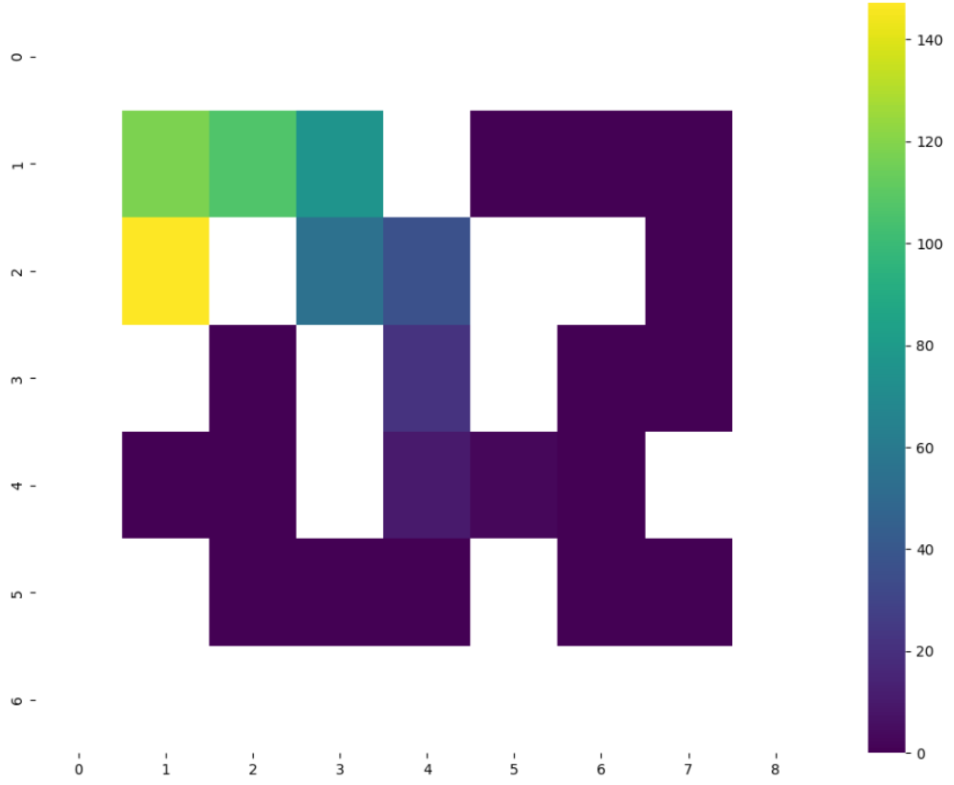
2. How the Value Function is Updated?

Visualization of the value $\max_{a'} \hat{Q}(s, a')$ of movable coordinates at regular episodes

Visualize the value of each state with a heat map

The "state_value_animation.gif" is Visualization results





Exercise 1: Examining Learning Costs

3. Which Requires More Training Samples: Randomizing Initial States or Randomizing Goal States?

Let's consider the forecast and the reasons for it.

4. Verify Your Answer to Question 3 Through Programming

Modify the following input variable at the bottom of the program:

```
env = Maze(maze_layout, sub_goal=False, random_start=False, random_goal=False)
```

Judge learning success by:

- Is the sum of rewards in the learning curve high?
- Can the final learning trajectory reach the goal?

If the number of samples is insufficient, please adjust the following variables:

```
max_episodes=300  # Number of trial-and-error episodes. Increase this value
num_visualize_trajectory = 1 # Change to 1 as visualization takes time
```

Mini-Report Assignment

- Consider the problems that can occur when combining with Deep Neural Network (DNN)

・Assignment

1. Problems occurring from combining RL and DNN
2. If you can afford it, please consider ideas for solving that problem as well

・There is a paper on the front table for your report, so please answer it and turn it in at the end of lecture

・The quality of the report content has nothing to do with the grade, so please feel free to consider it!

Please try about 30 minutes on Exercise 1 and Mini-Report Assignment

- Explanations will restart at 11:55

- Mini-Report Assignment
    1. Problems occurring from combining RL and DNN
    2. If you can afford it, please consider ideas for solving that problem as well

Exercise 1: Explanation

3. Which Requires More Training Samples: Randomizing Initial States or Randomizing Goal States?

Randomizing goal states requires more training samples. This is because the state space that needs to be explored becomes larger

- With random initial states but fixed goal, the state space to explore is only 2-dimensional (current coordinates)

- In contrast, with random goal states, the state space becomes 4-dimensional (goal coordinates plus current coordinates)

With max_episodes=10000, sufficient learning is possible even with random goal states

Contents

1. Reinforcement Learning Problem Settings and the Concept of the Learning Process

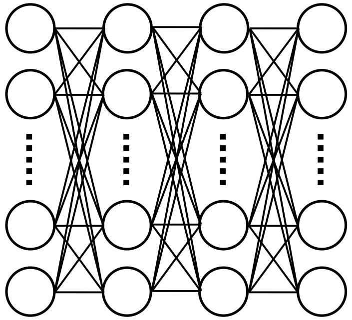
2. Deep Reinforcement Learning: Challenges and Solutions in Neural Network Integration

Approaches to Combining Deep Learning and Reinforcement Learning

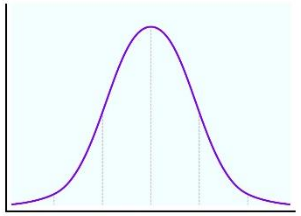
- Approximating Policies and Value Functions with Deep Learning

In the case of Gaussian policy $\pi(a|s)$

State s



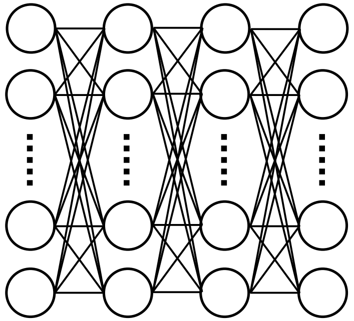
Mean $\mu(s)$
Variance $\sigma(s)$



Action a

In the case of value function $Q(s, a)$

State s



$Q(s, a_1)$
$Q(s, a_2)$
$\vdots$
$Q(s, a_{N-1})$
$Q(s, a_N)$

Outputting values for each action a_n conditioned on state s

All action values can be obtained in a single forward pass

Actions and values to be approximated are determined by the reinforcement learning algorithm

Reinforcement Learning System Changes with DNN

- Action Decision Process

Without DNN

low-dimensional state

Observation state → Task-dependent feature extractor → Acquire with RL behavioral rules → Task-dependent action generator → Action

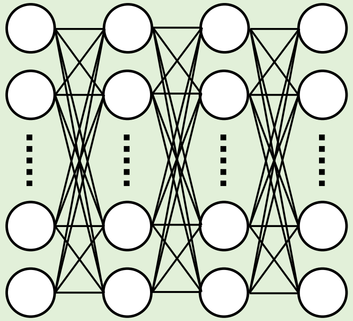
low-dimensional action

The performance of the system is highly dependent on the quality of the overall design

Since there are few pairs of states and actions to learn, the number of samples required is small

With DNN

Observation state →



→ Action

Capable of learning high-dimensional input/output policies appropriate for the task

Since there are huge pairs of states and actions to learn, the number of samples required is large

Reinforcement Learning System Changes with DNN

- Changes in the skills and capabilities required to develop systems involving RL

Without DNN

- Understanding RL performance
- Safe trial-and-error environment
- Knowledge to simplify learning tasks (required)

With DNN

- Understanding RL performance
- Safe trial-and-error environment
- Knowledge to simplify learning tasks (not required)

+

- Environment capable of collecting huge amounts of trial-and-error data
- Parameter tuning of RL and DNNs

Improved performance, but higher learning costs

Challenges for Combining RL and DNN

- Challenges in approximating policy and value function with DNN

C1. Dynamically changing target values to be approximated by DNN

E.g. target value of Q-learning

$$r(s,a,s') + \gamma \max_{a'} \boxed{\hat{Q}(s',a')}$$

Value function dynamically changing with learning

The updated value depends on a value function that changes dynamically with learning

C2. Overlearning occurs in regression with large network structures

The larger the network size, the more data is required

However, since RL updates policy based on experience data, in many cases there is not enough data for large networks

Main Ideas for Solutions

- Aligning RL updates with the approach of supervised learning

Since deep learning is a supervised learning algorithm, approximation performance is high when it is close to the problem setting of supervised learning

- Examples of solutions in Deep Q-Network [V.Mnih et al., 2013]

A1. Target network: gradually changing target values [V.Mnih et al., 2013]

A2. Experience replay: using a variety of experience data [L. Lin, 1992]

These two tricks are also used in current DRL algorithm

Explain the two tricks

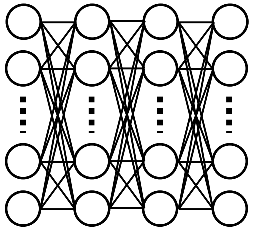
A1. Target Network: Gradually Changing Target Values

- Temporarily fix value neural network to calculate target values

1. Copy value neural network and prepare the network for fixing

$\hat{Q}(s, a; \theta)$

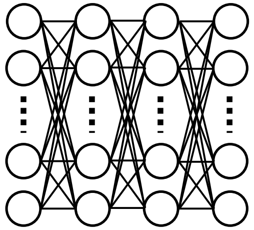
Network weights θ



$\rightarrow$

$\hat{Q}(s, a; \theta^{-})$

Copied network weights θ^{-}



2. Calculate target values with copied network θ^{-} and update network θ

$$r(s, a, s') + \gamma \max_{a'} \hat{Q}(s', a'; \theta^{-}) \xrightarrow{\text{Update}} \hat{Q}(s, a; \theta)$$

3. Copy the weights again after updating the value network a certain number of times

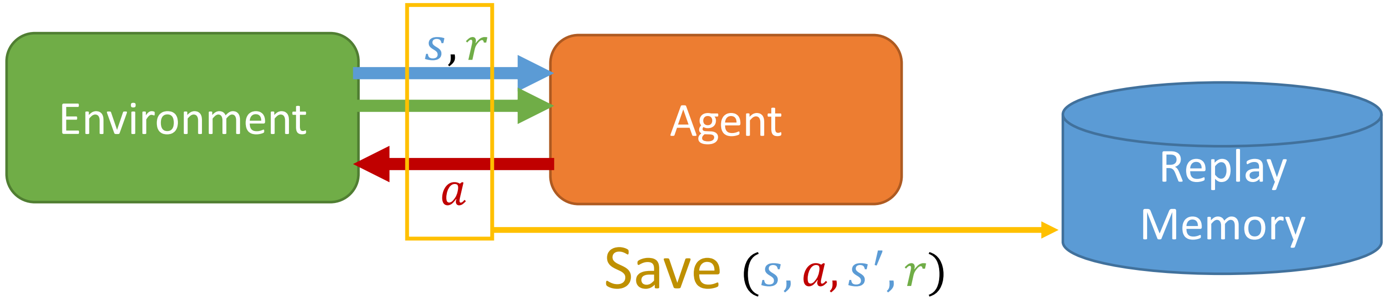
Network weights θ $\rightarrow$ Copied network weights θ^{-}

Stabilize the target value fluctuations by temporarily fixing value neural network

A2. Experience Replay: Using a Variety of Experience Data

- Store as much experienced data as possible and pick up data randomly during training

1. Store as much data as possible collected by agents



2. Randomly picks up stored data and updates value

Error with target value ： $r(s,a,s') + \gamma \max_{a'} \hat{Q}(s',a';\theta^-) - \hat{Q}(s,a;\theta)$

Avoid overlearning by updating from all collected data

Combining RL and DNN: Summary

- Challenges: approximate policy and value functions with DNN

C1. Dynamically changing target values to be approximated by DNN

C2. Overlearning occurs in regression with large network structures

- Solution: Aligning RL updates with the approach of supervised learning

DQN's example is handled by the following two tricks

A1. Target network: gradually changing target values

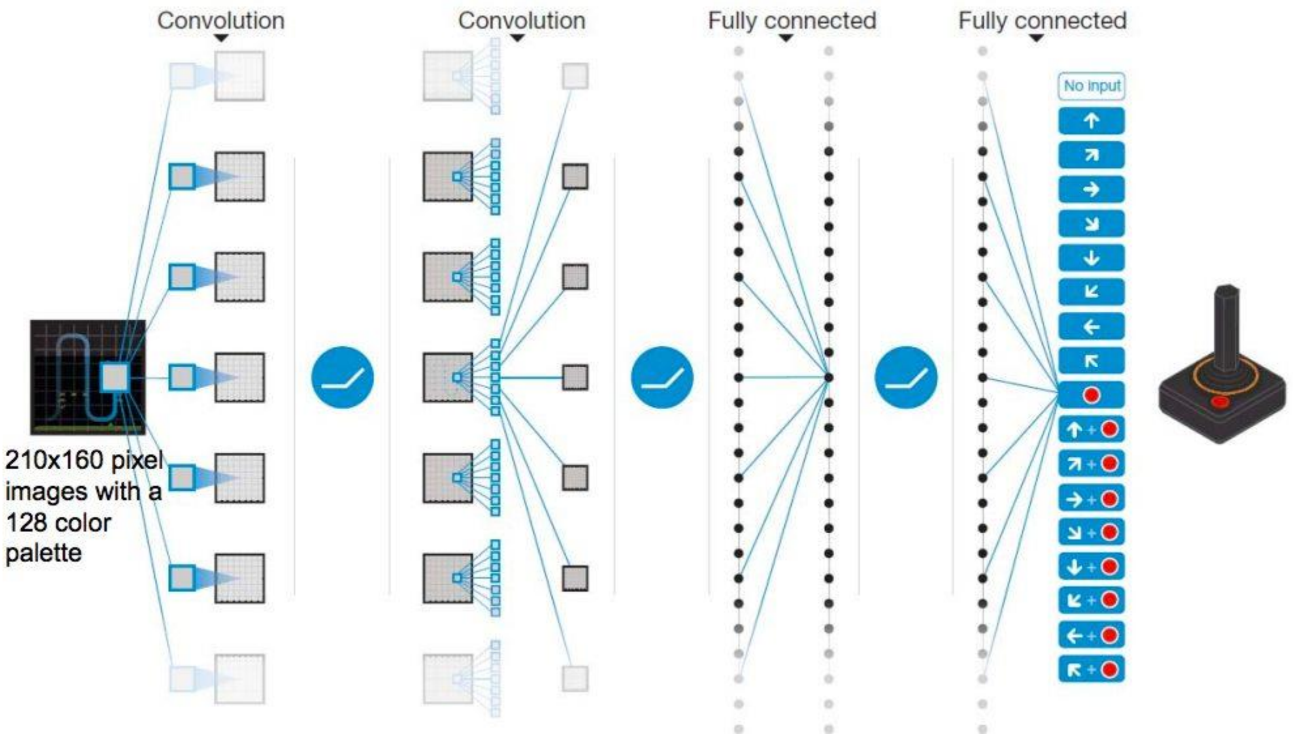
A2. Experience replay: using a variety of experience data

Deep Q-Network [V.Mnih et al., 2013]

- The research that triggered attention towards DRL

Approximate the value function of Q learning with Convolutional Neural Network (CNN)

It became possible to directly input images into the policy

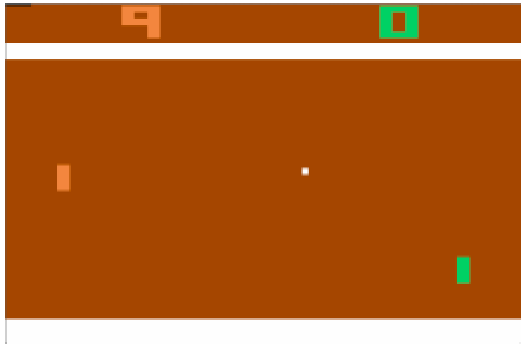


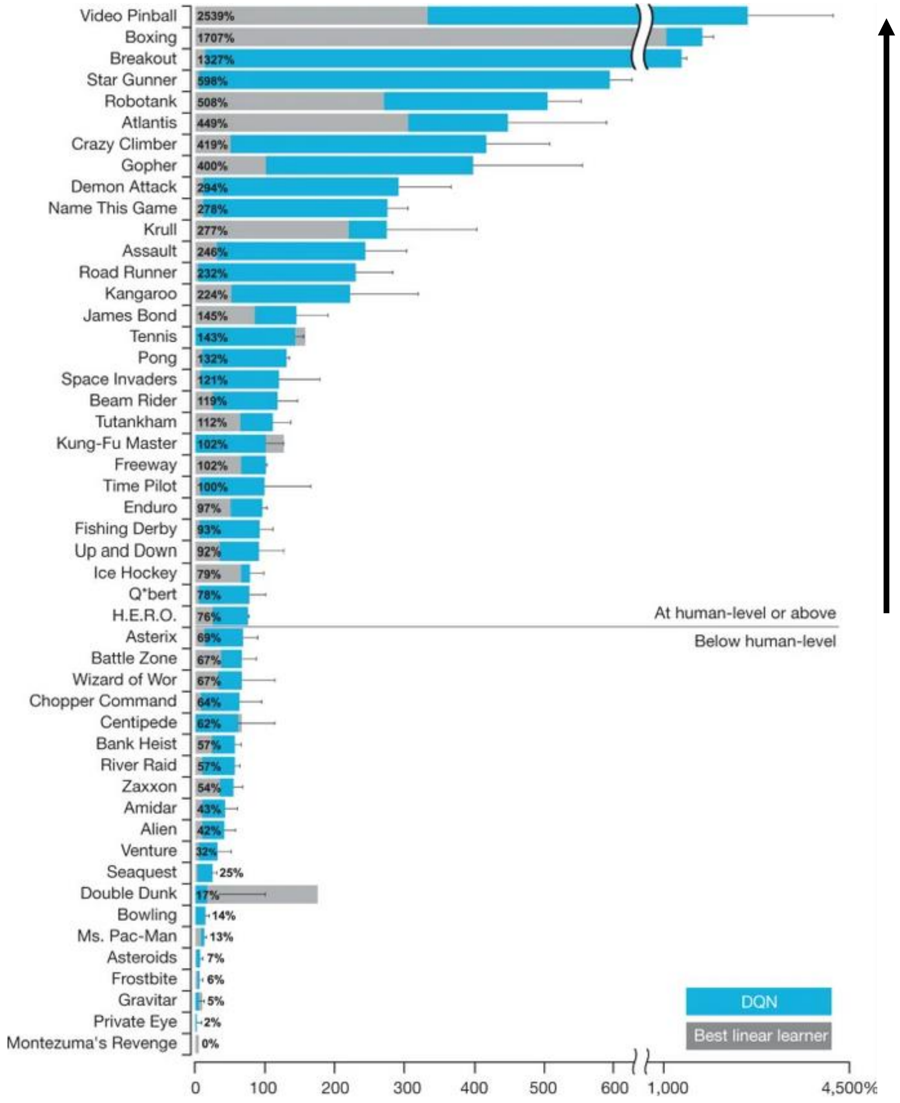
The research tested DQN's performance in 49 different video games

Deep Q-Network Performance

- More than equal to humans in 29 different games

Tested on 49 different Atari games





29 different games

Deep Q-Network Performance

- Video: Breakout

- Video: SPACE INVADERS

The strengths and weaknesses of DQN in different tasks

- Strength tasks

Many chances to be rewarded for random movements

Reflexive and simple movements

- Weaknesses tasks

Random movements do not yield rewards (e.g., Immediate failure game)

Many pairs of state actions to be learned

Similar trends in many other DRL

Some previous studies have focused on weaknesses and improved performance

Exercise 2: Deep Q-Network

Let's actually training Deep Q-Network!

Objective:

Confirm that DQN policies can be learned using vector and image states as input

Confirm the effectiveness of Experience Replay and Target Network used in DQN

Task: Reaching Task

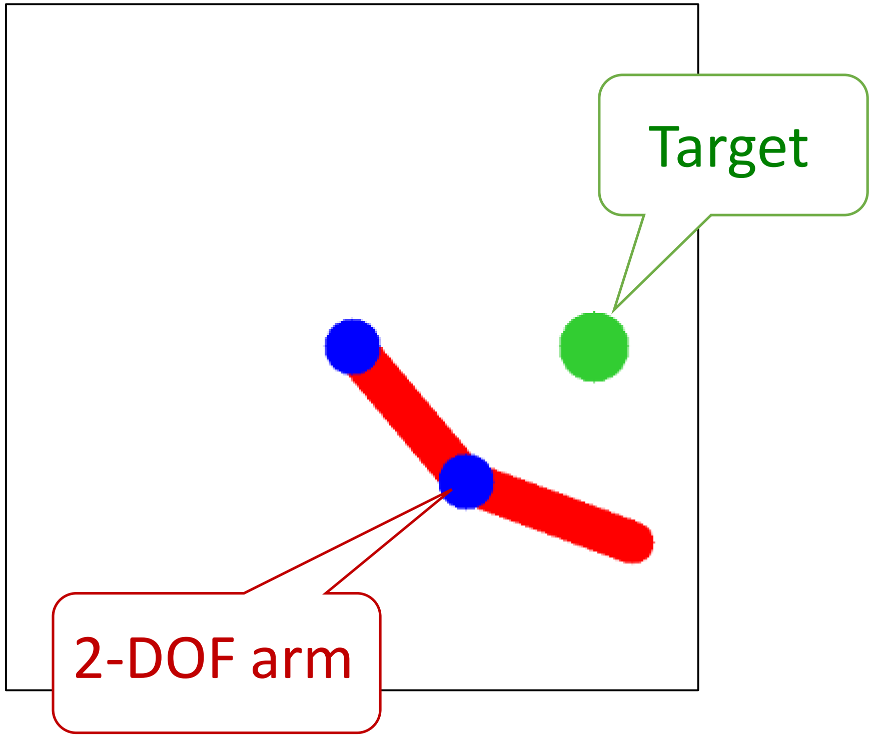
Task Objective: Minimize the distance between the target and the arm endpoint

State: Discrete joint states or image state

Actions: Which joint to move and by how much?
Differential actions consist of 5 types (5*2=10 actions)

Reward: Various reward functions to be tested

In this task, one episode consists of 30 steps



Exercise 2: Deep Q-Network

Access Google Colaboratory through the syllabus or the URL below

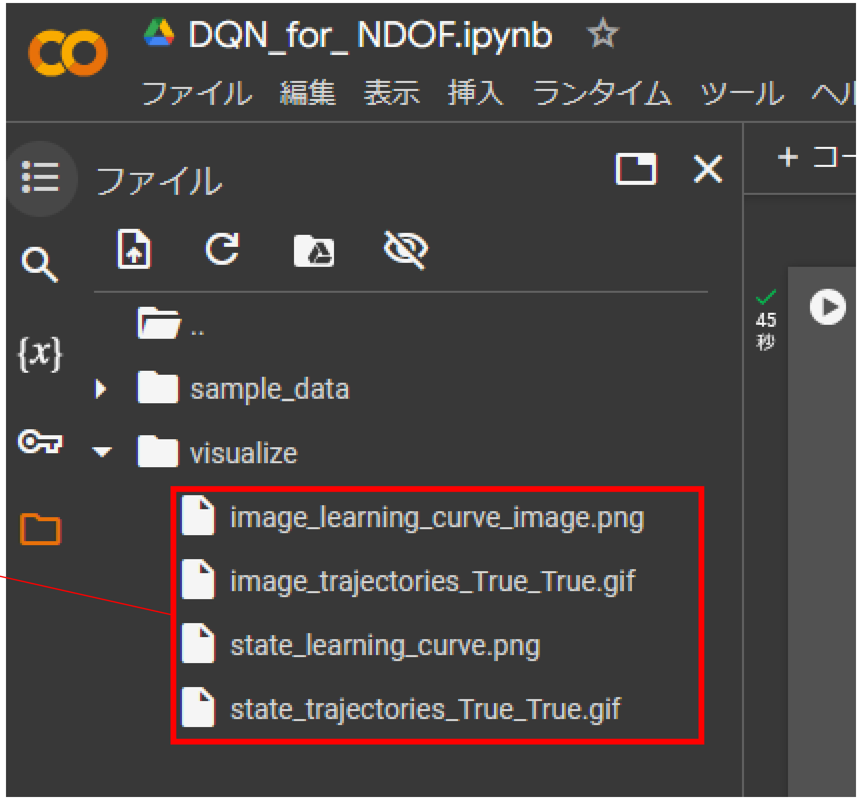
https://colab.research.google.com/drive/1vPZojYhlQ-hQ5DldSXsk1DnERli8wCT3?usp=sharing

2-1 Examining DQN Learning with Vector States and Image States

The state data format can be switched by changing the following variable

```
use_image_state = False#True
```

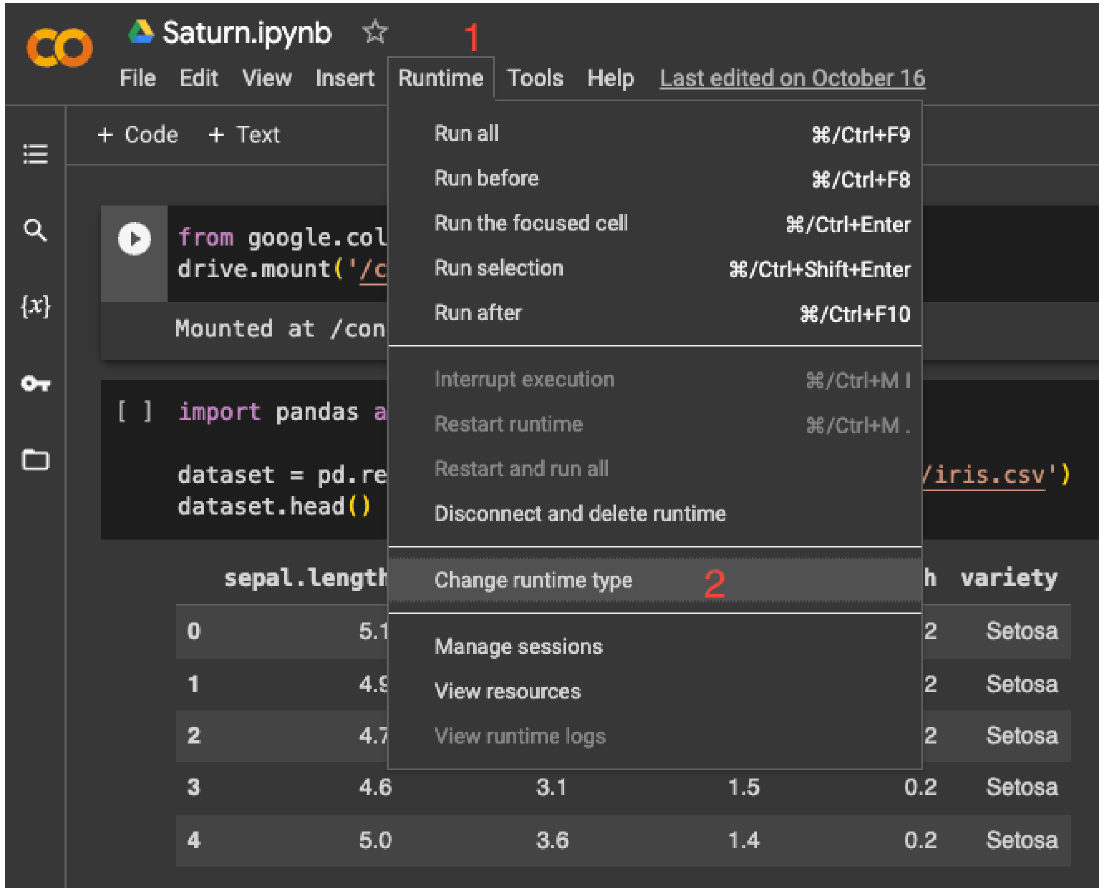
Check the learning curves and learning trajectories

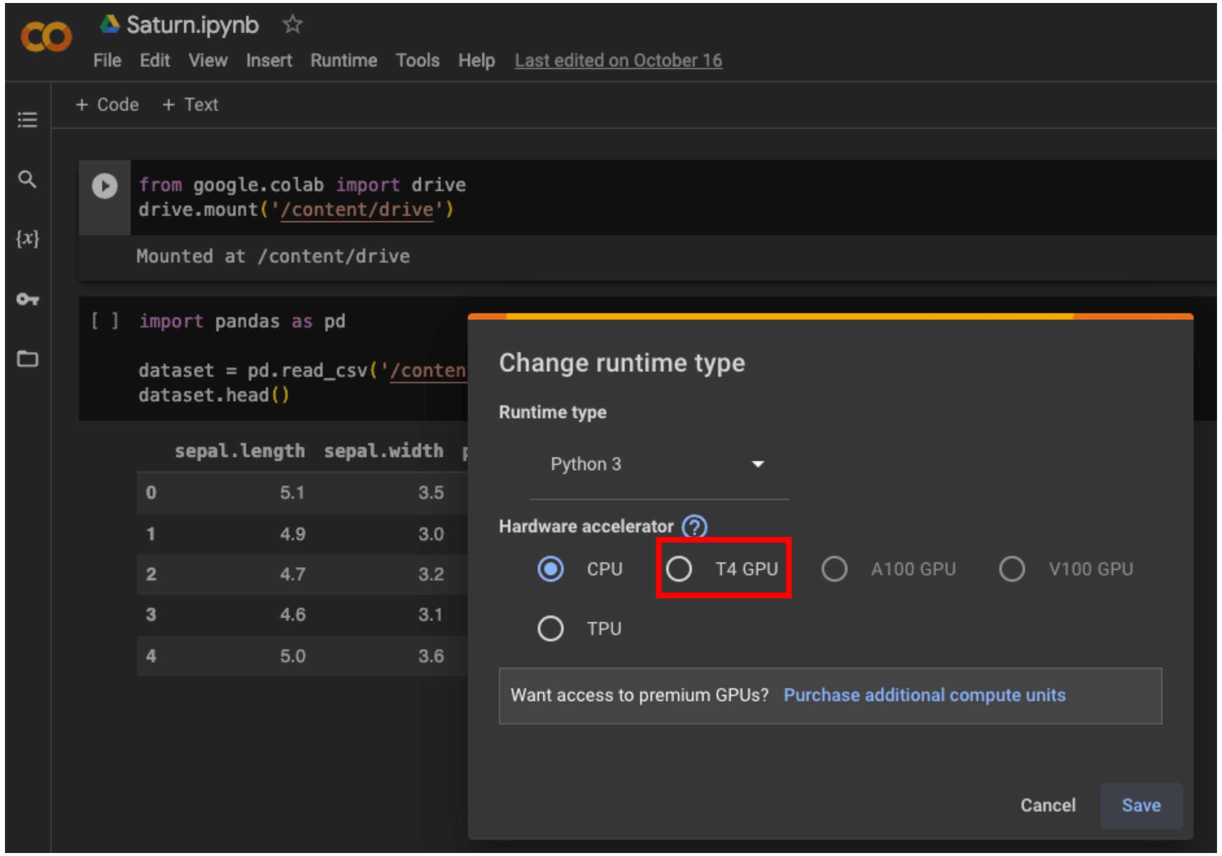


Exercise 2: Deep Q-Network

GPU Activation

Hardware acceleration changed from CPU to T4 GPU





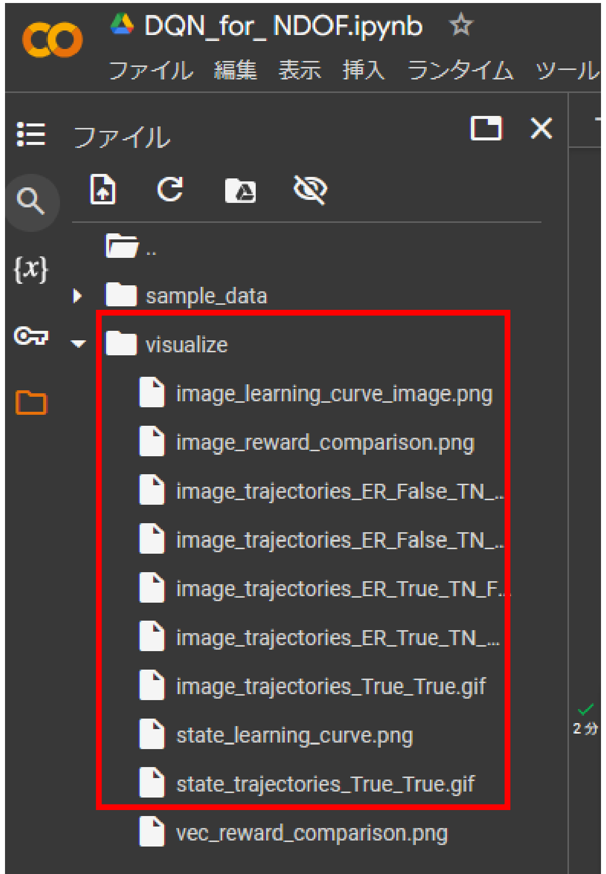
[https://saturncloud.io/blog/how-to-activate-gpu-computing-in-google-colab/]

Exercise 2: Deep Q-Network

2-2 Compare the Results of Vector States and Image States, and Analyze the Reasons for These Differences

Hint: Focus on the differences in learning conditions related to Experience Replay (ER) and Target Network (TN).

Specifically, compare “vec_reward_comparison.png” with “image_reward_comparison.png”, and compare “vec_trajectories_*.gif” with “image_trajectories_*”.gif

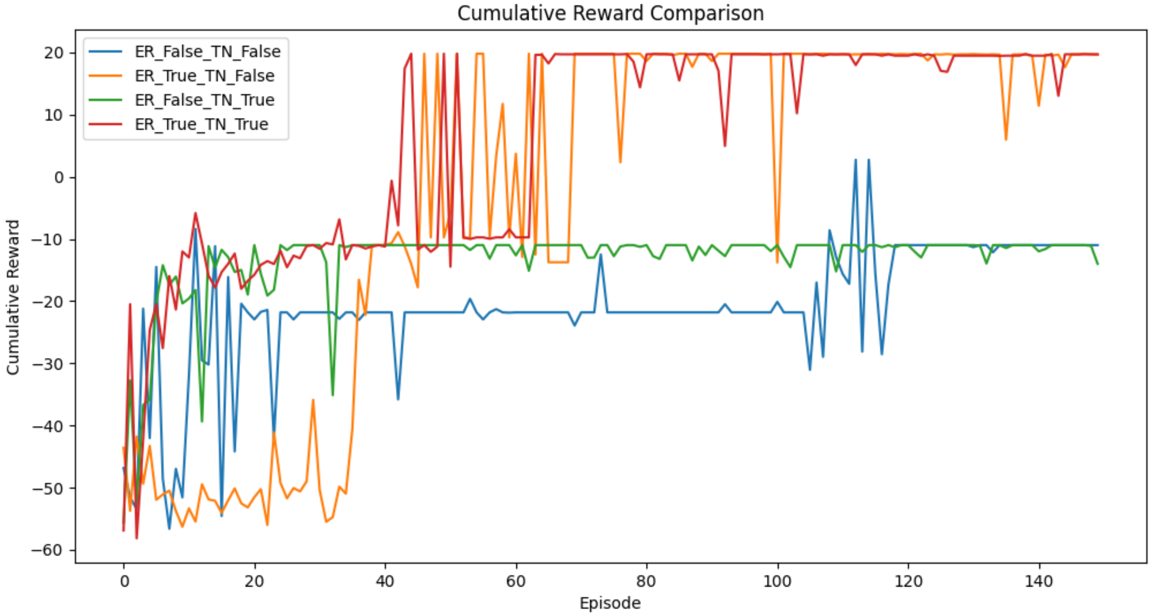


Please try about 10 minutes on Exercise 2

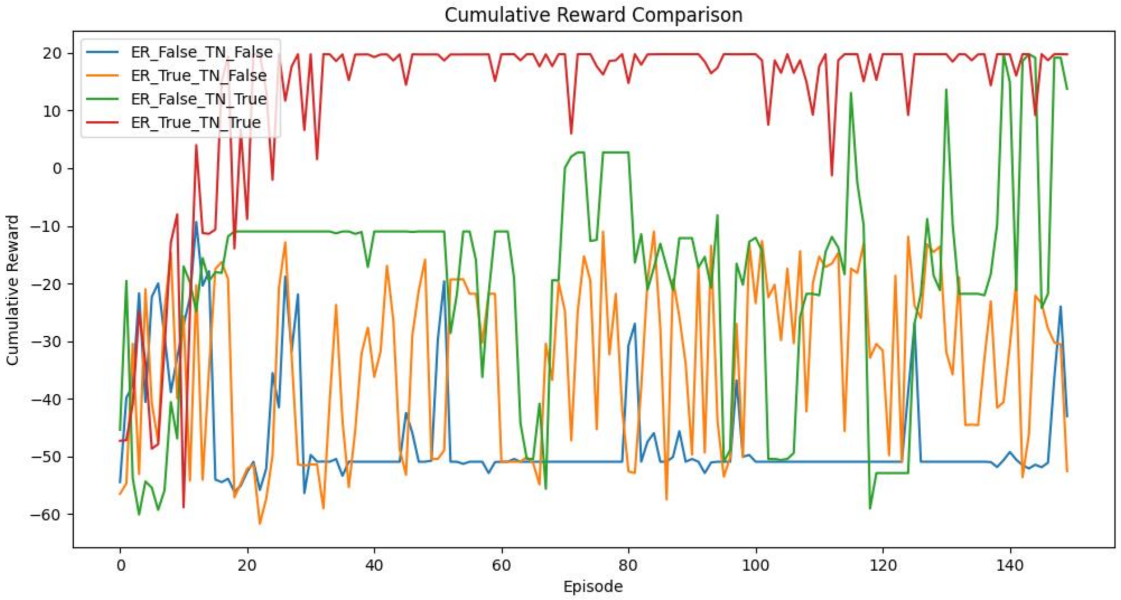
Example Answers for Exercise 2-2

2-2 Compare the Results of Vector States and Image States, and Analyze the Reasons for These Differences

Learning curve for vector input



Learning curve for image input



Comparison between Vector States and Image States:

With image states, the neural network is larger and learning is less stable

→ The stabilizing effects of ER and TN are more significant for image inputs

Summary

Explanation focusing on RL and DRL

- RL problem settings and the concept of the learning process

- Deep Reinforcement Learning: Challenges and Solutions in Neural Network Integration

- Through practical exercises, confirmed the correspondence between the explanation of reinforcement learning and the actual execution results

Please submit your mini-report to the front desk