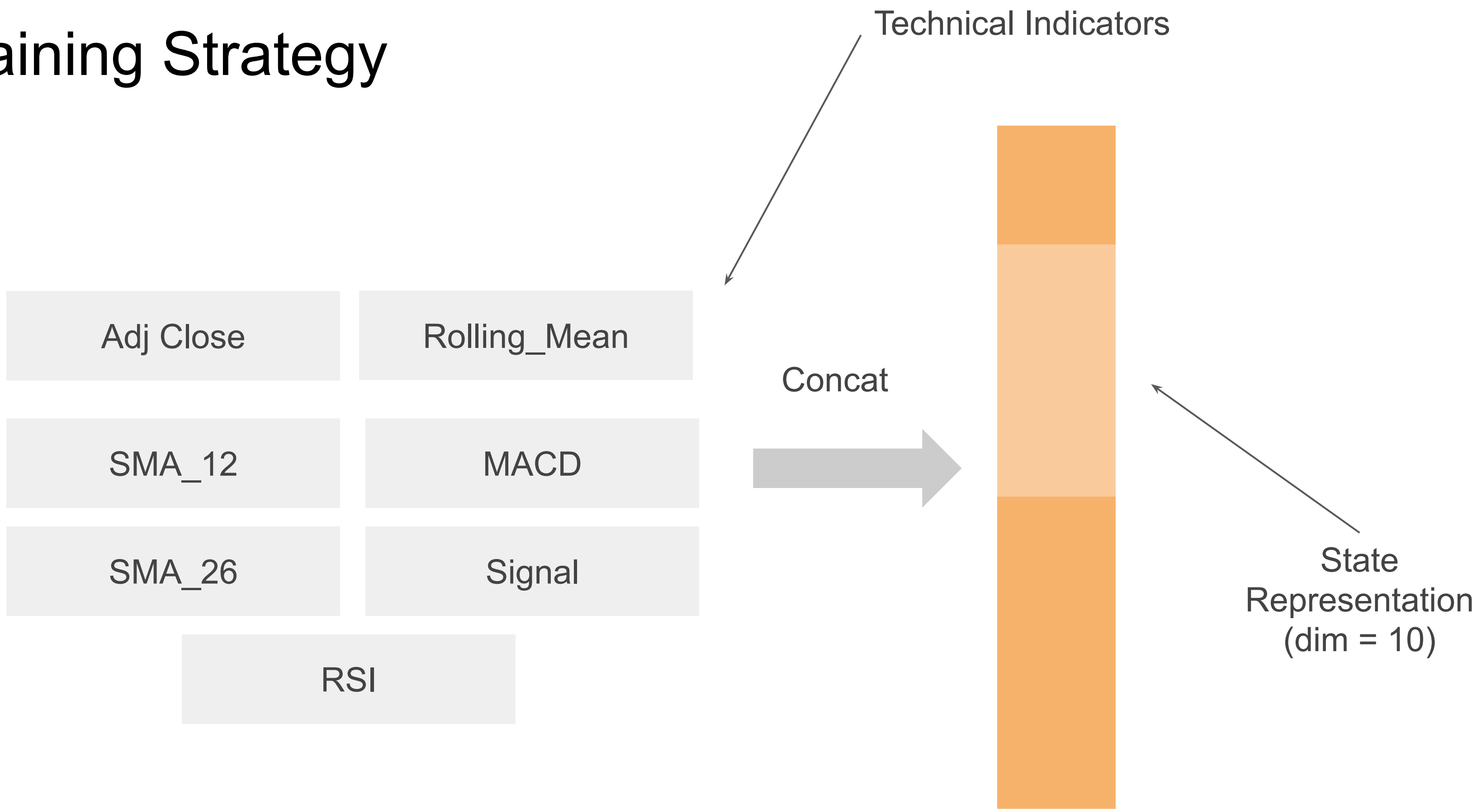
Fintech HW1 Repoert

r11944064 梁家綸

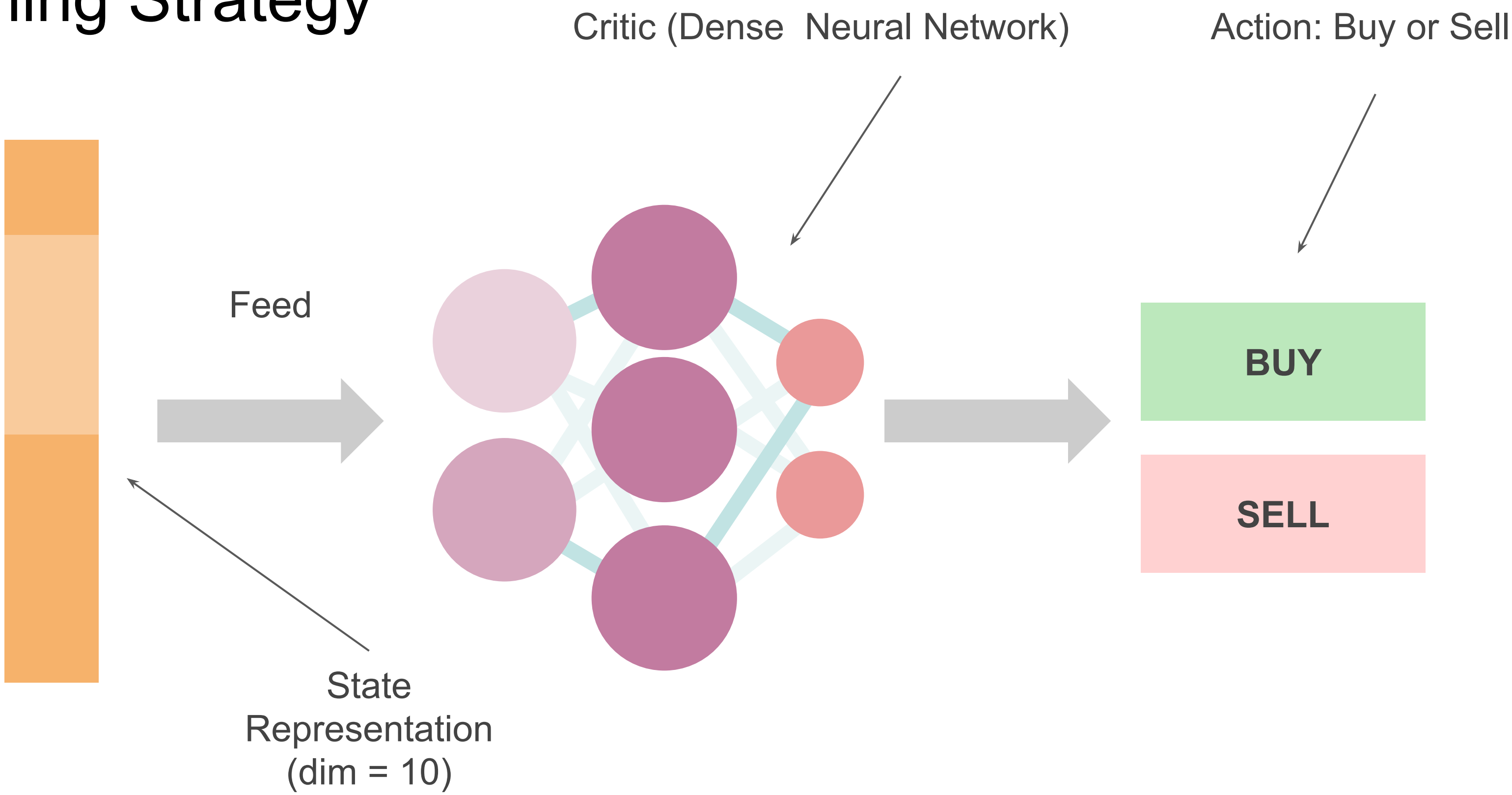
Summary

1. Github repository: https://github.com/b05505027/fintech_hw1
2. I trained a robo-advisor using **a DQN algorithm** with data from 0050.tw.
3. I tested the advisor on **the most recent 70 days of data** and trained it using **data from before those 70 days**.

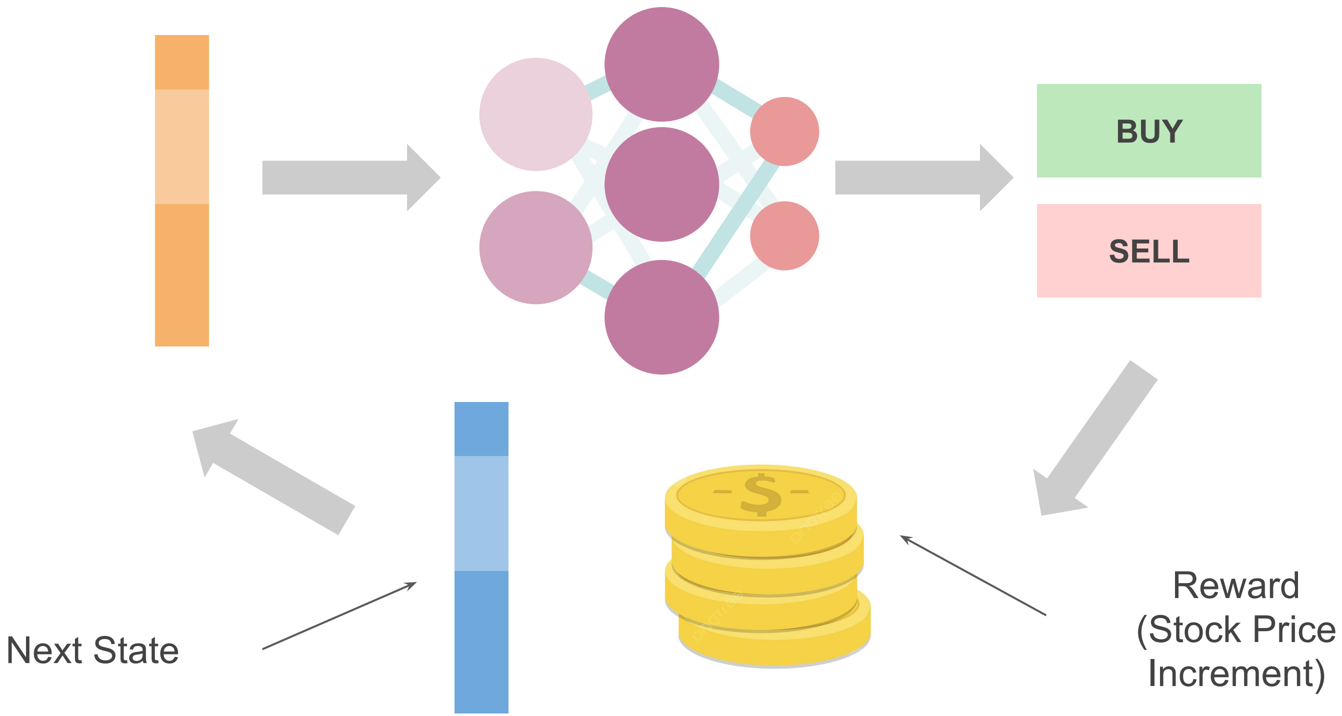
Training Strategy



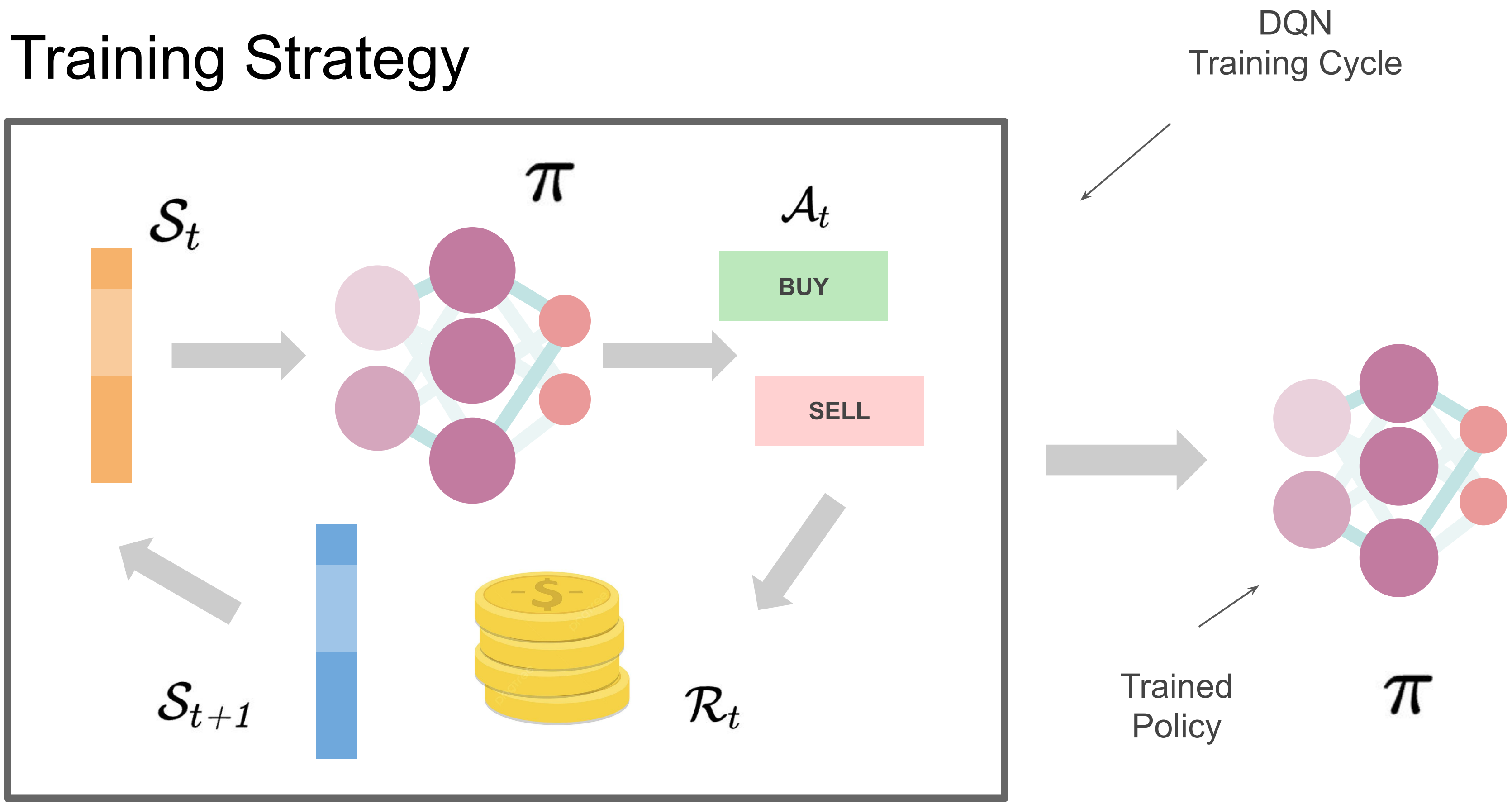
Training Strategy



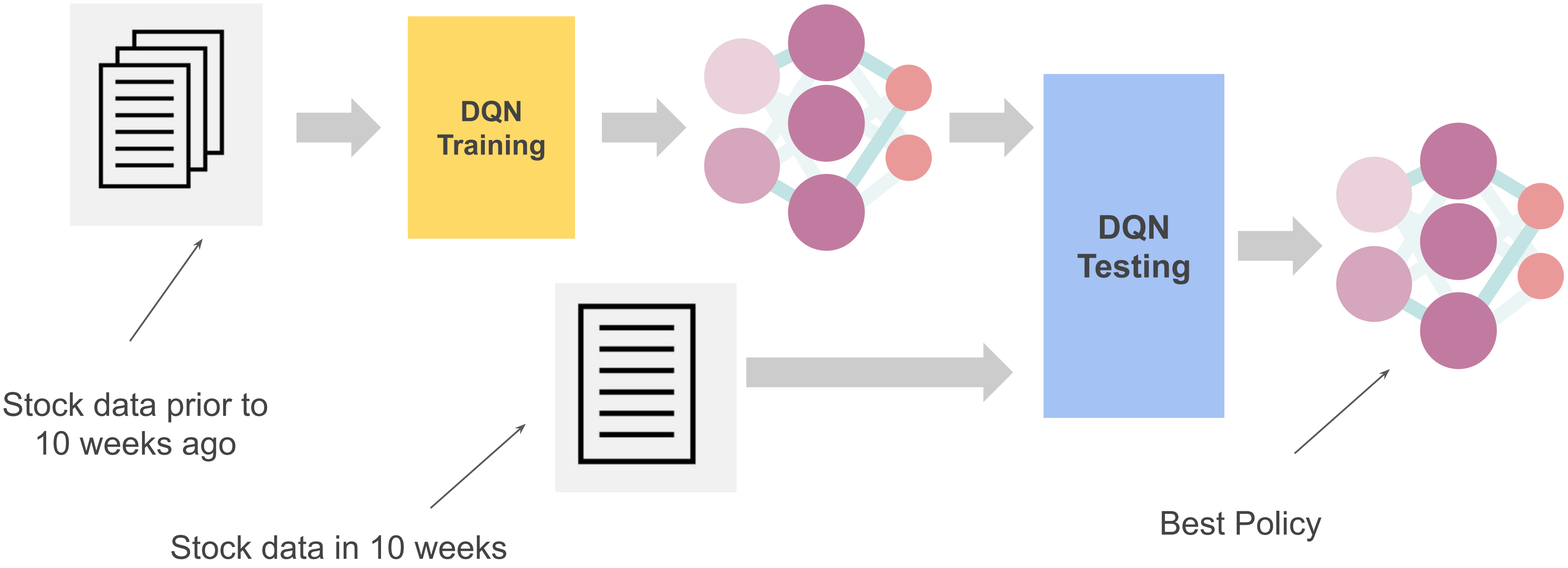
Training Strategy



Training Strategy



Training Strategy



Reward Function

Naturally, we aim to maximize the profit, and minimize the loss when holding stocks. So I choose the reward function:

$$r = 10 \cdot r_1 + 0.05 \cdot r_2$$

$$r_1 = \sum_{t=1}^{H} \gamma^{t-1} ln(G_t) \times \mathbb{1}[HoldingStock > 0]$$

$$G_t = \frac{StockPrice[t+1]}{StockPrice[t]}$$

$$r_2 = \sum_{t=1}^{H} \gamma^{t-1} HoldingStock \cdot (StockPrice_{t+1} - StockPrice_t)]$$

Algorithm

Algorithm 1 DQN Robo-Advisor

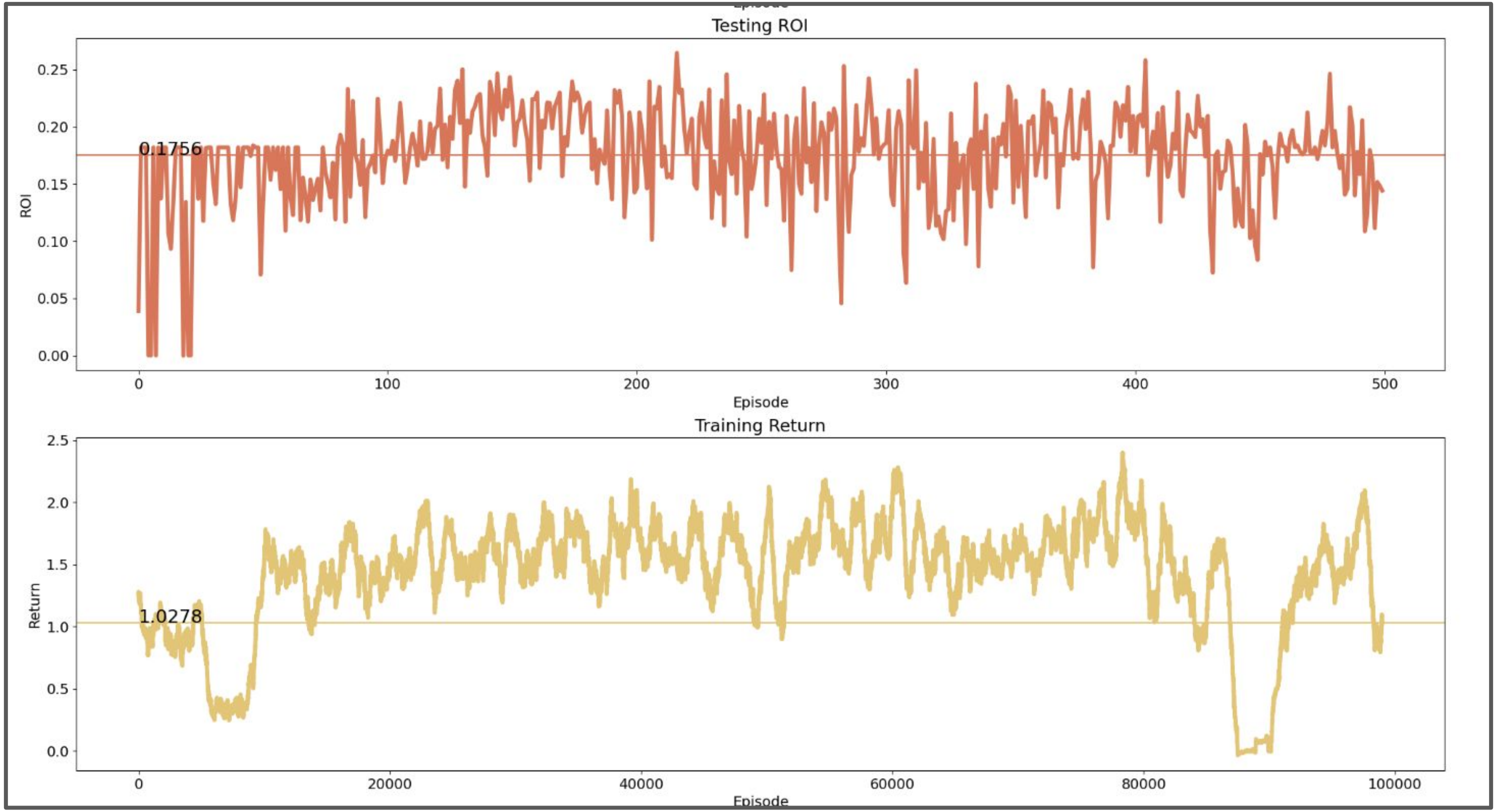
Initialize the critic network
Initialize the buffer_queue
exploration_rate $\leftarrow 1$
for $i = 0$ to episodes **do**
    randomly select a 70-days data from the historical data
    for $j = 0$ to 69 **do**
        get the state
        if random_number < exploration_rate **then**
            use the random binary action
        else
            predict the action through the state and critic
        end if
        exploration_rate $\leftarrow$ *exploration_rate* $\times 0.999$
        based on the action, get the reward and the next state
        store transition (s, a, r, s') to the buffer_queue
        if $j == 69$ **then**
            next_state $\leftarrow$ None
        end if
    end for
    sample a batch of data from the buffer
    update the critic using MSELoss between $r + Q(s', a')$ and $Q(s, a)$
end for
return the best policy

Parameters

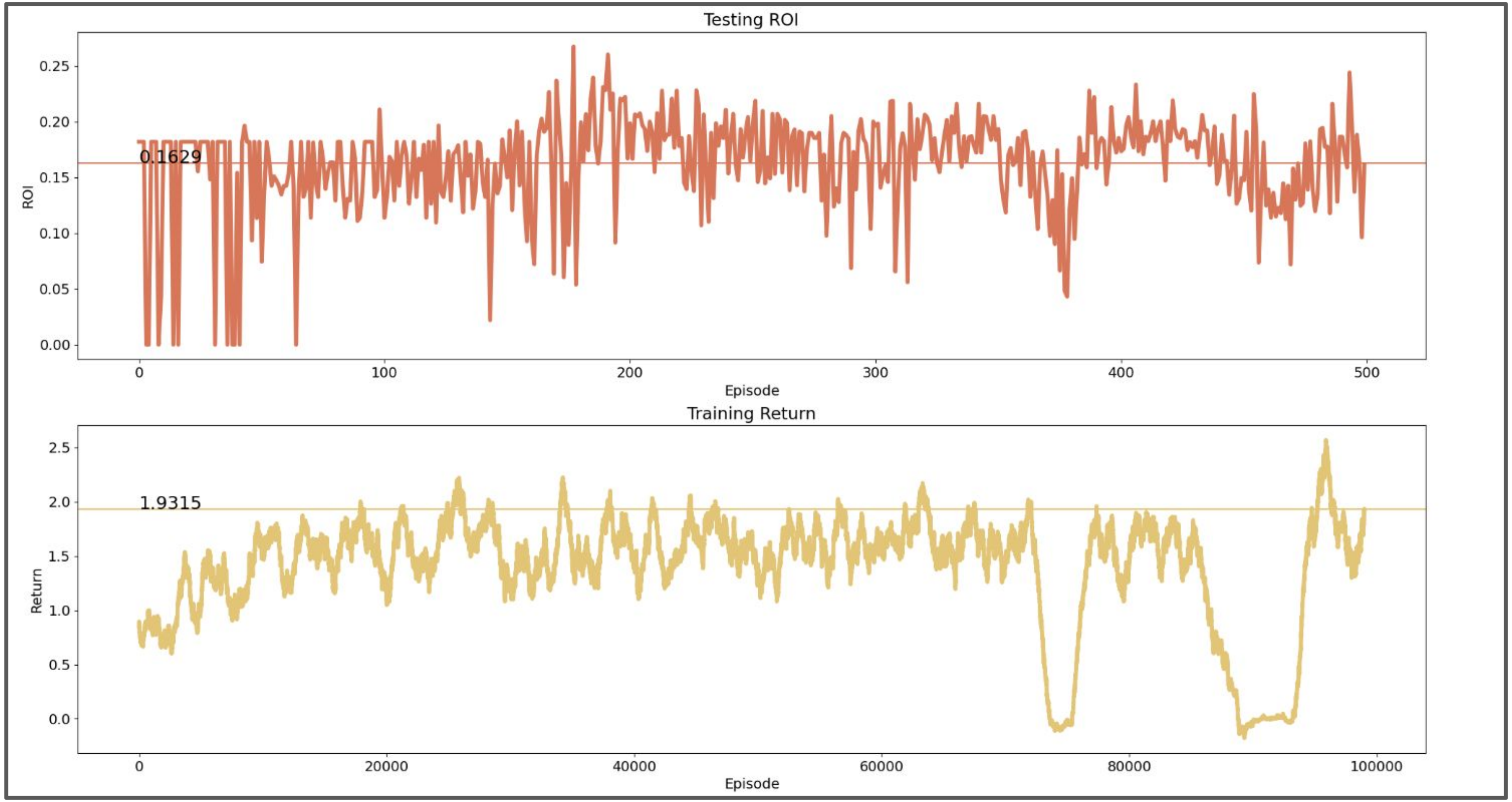
Table 1: DQN Robo-Advisor Parameters

Parameter	Value
Hidden Layer Size	64, 32 ; 400, 300
Input Size	7
Output Size	2
Normalization Method	Layer Normalization
Activation Function	ReLU
Optimizer	Adam
Learning Rate	1×10^{-3}
Training Episodes	100,000
Technical Indicators	Adj Close, MA12, SMA26, MACD, Signal, RSI, Rolling Mean
Batch Sizes	256, 128
Buffer Sizes	10000, 20000
Gamma	0.9, 0.95, 0.99

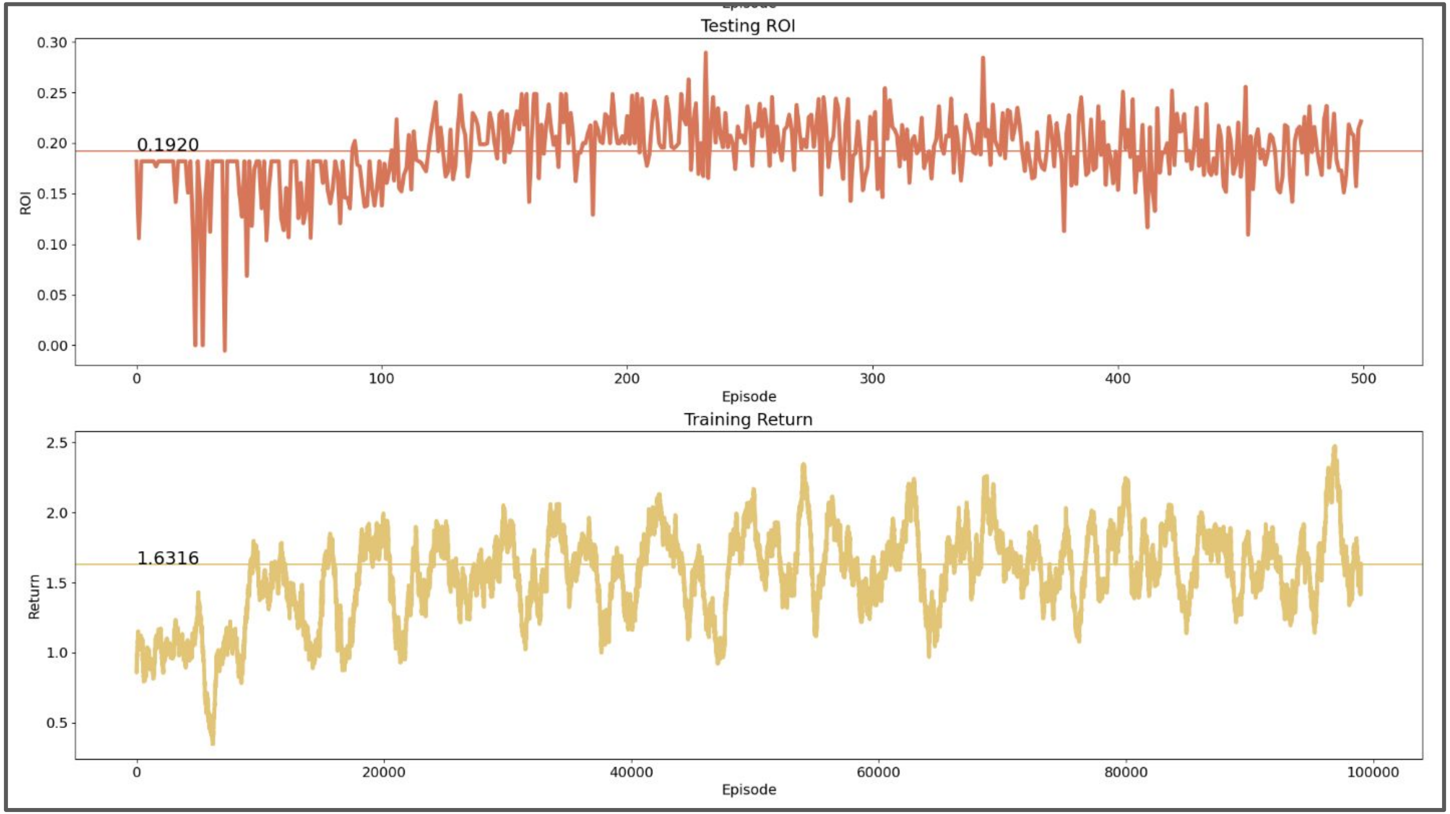
Training results: batch_size_128_buffer_size_5000_gamma_0.99



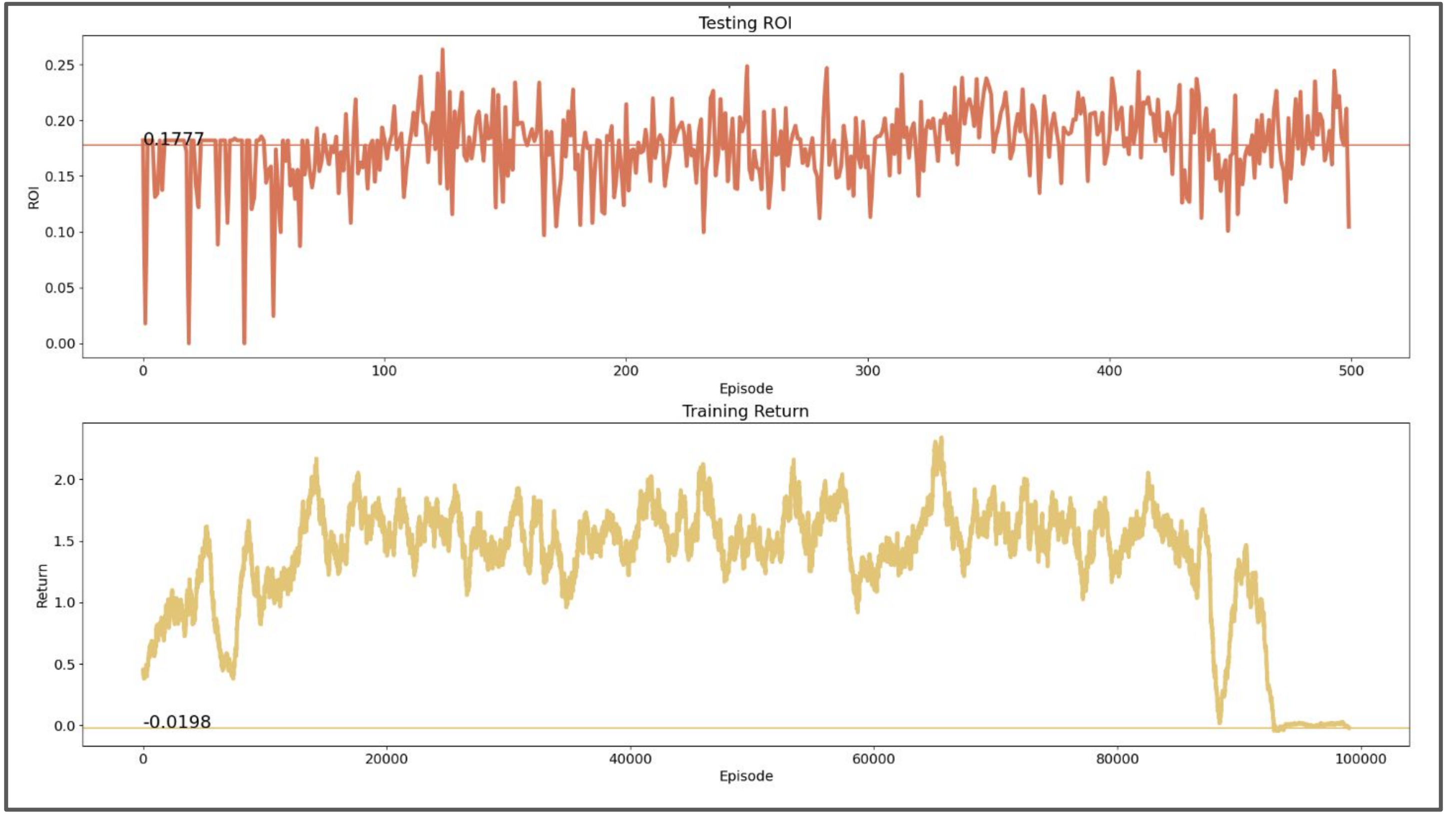
Training results: batch_size_128_buffer_size_5000_gamma_0.995



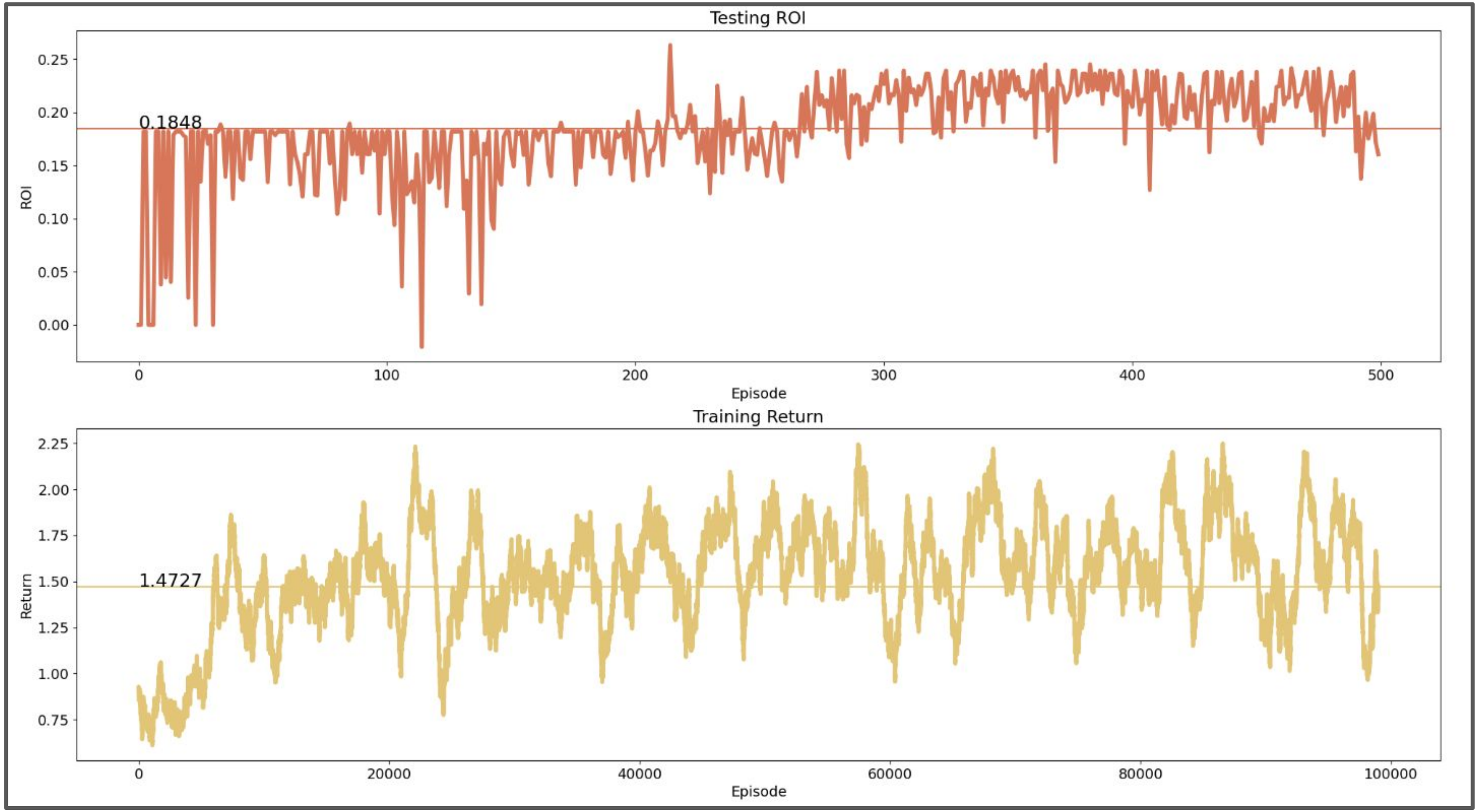
Training results: batch_size_128_buffer_size_5000_gamma_0.999



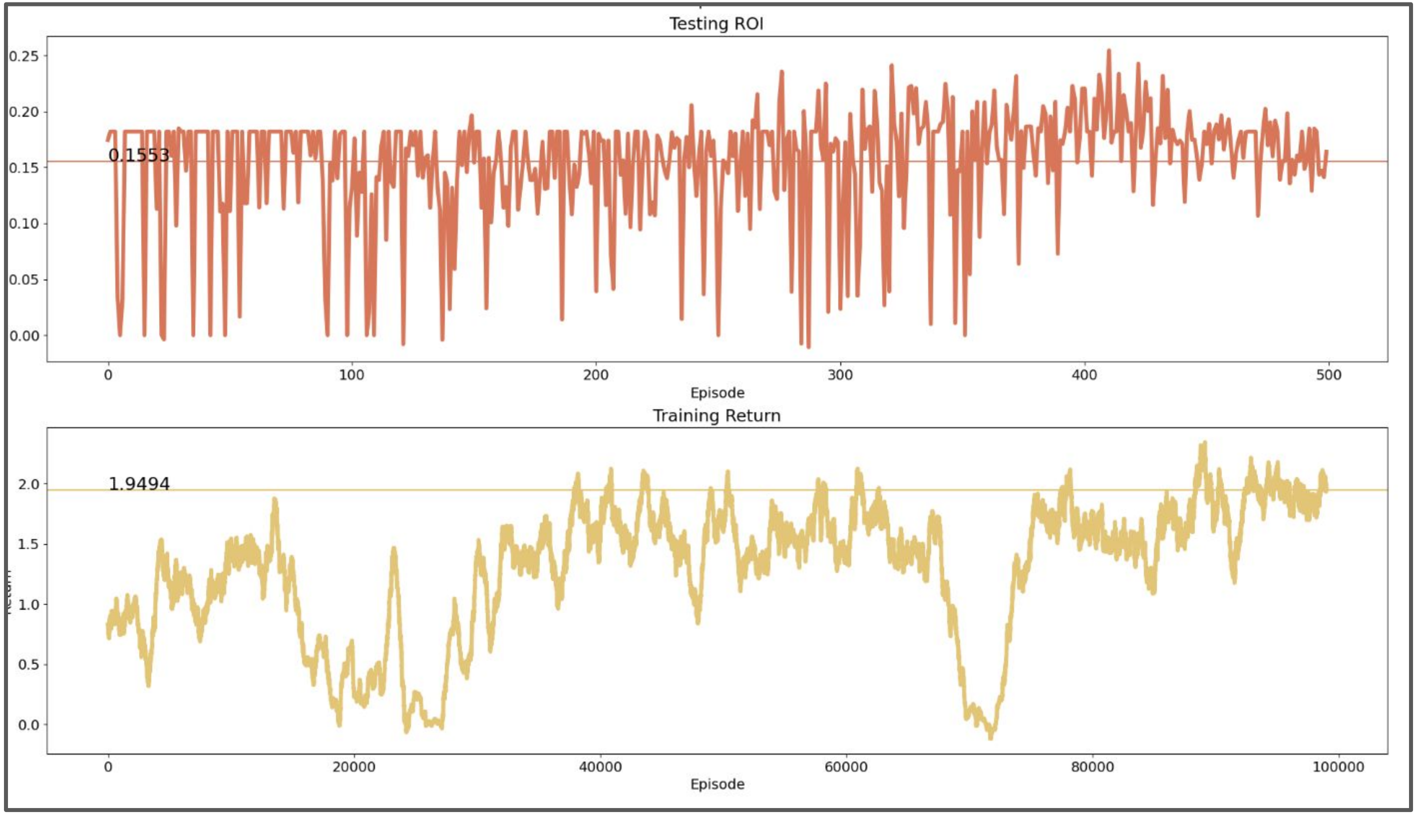
Training results: batch_size_256_buffer_size_10000_gamma_0.9



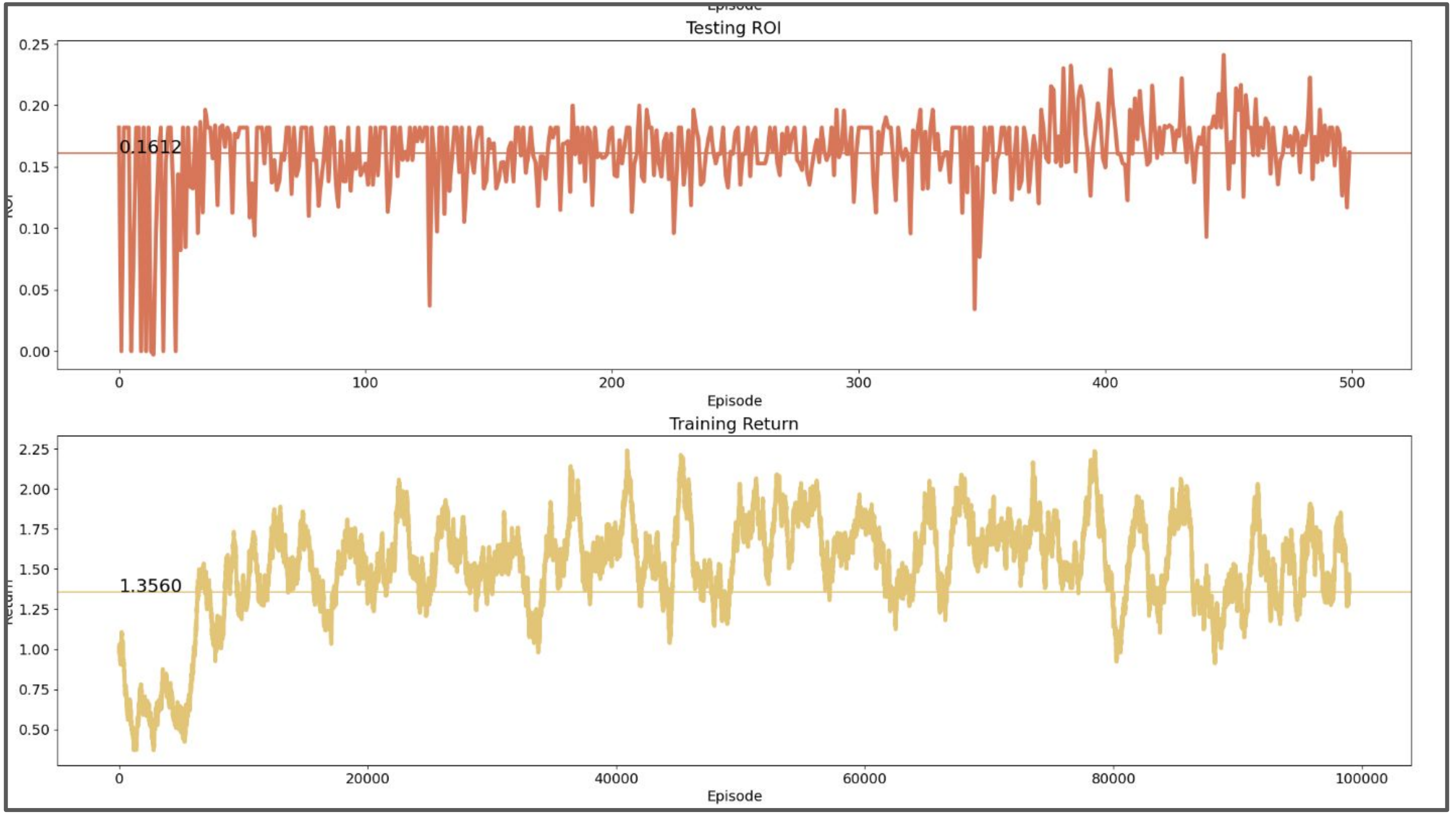
Training results: batch_size_256_buffer_size_10000_gamma_0.95



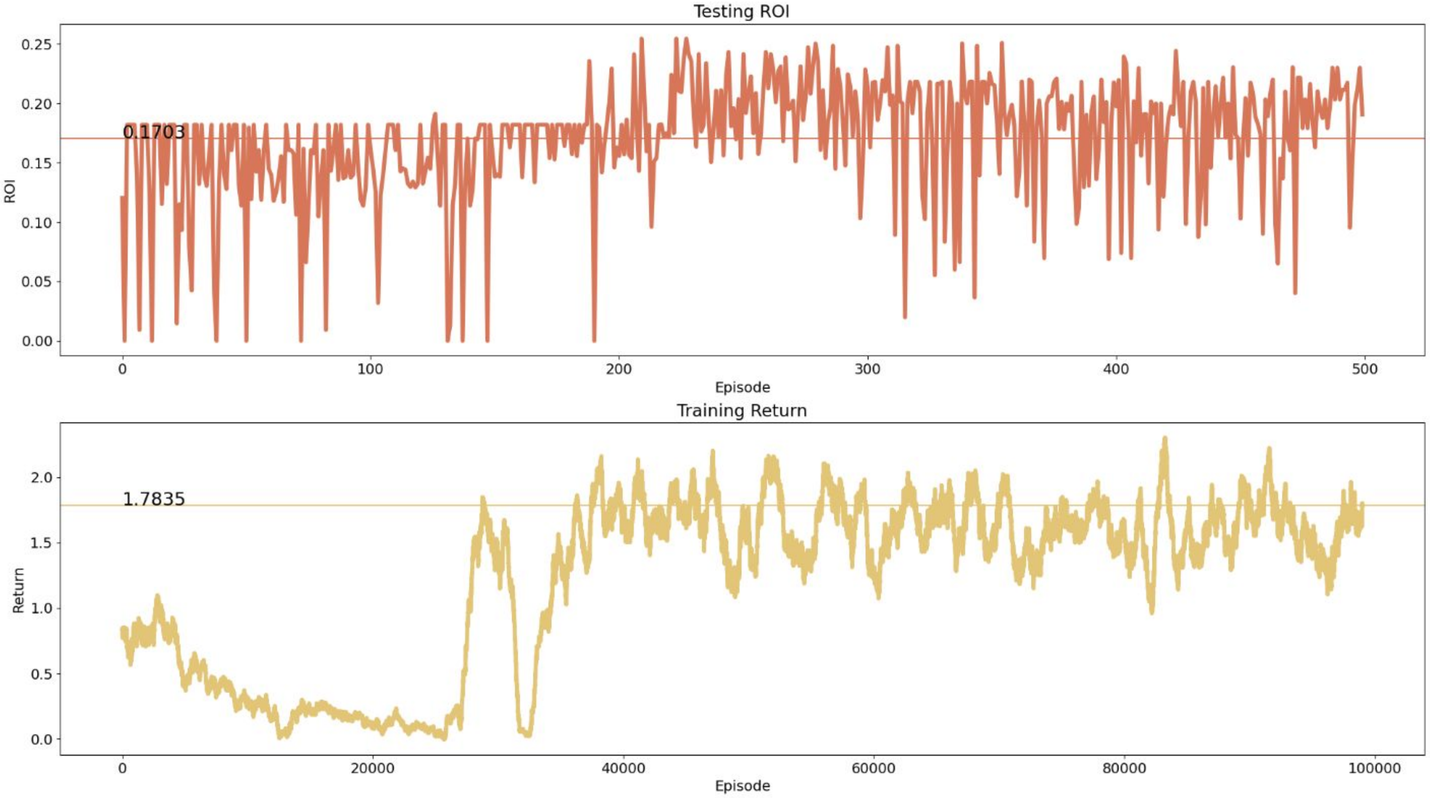
Training results: batch_size_256_buffer_size_10000_gamma_0.99



Training results: batch_size_256_buffer_size_20000_gamma_0.9



Training results: batch_size_256_buffer_size_20000_gamma_0.95



The Final Model

1. I choose the **batch_size_256_buffer_size_10000_gamma_0.95** as the final model.
2. For Simplicity, I didn't consider using the ensemble technique.
3. By executing the command python rrEstimate.py 0050.TW-short.csv, it can achieve **an approximate return rate of 110%**.

```
(fintech1) liangjialun@liangjialundeMacBook-Pro R11944064_t % python rrEs
timate.py 0050.TW-short.csv
rr=110.878849%
```