Final Project Report

[EE488] Introduction to Reinforcement Learning

Finance Trading Policy

- **Department:** School of Electrical Engineering
- **Student ID:** 20200625
- **Name:** Yejun Joo

---

1. Code Implementation Explanation

- Environment.py
  - self.observation_space
    - Used spaces from gymnasium library to define observation space. Observation state is set to a dictionary type, including information of agent itself and that from market.
  - self.action_dim
    - Also used spaces foro action space definition. Action space is 1-dimension array with n-elements. Here n corresponds to action dimension, which is same with the number of tickers. Each element of action means the number of stock to trade, while positive value meaning buying and negative value meaning selling stocks.
  - reset()
    - I designed the code to randomly select the intial balance. However, it is not continuously random. It is discretized in unit of $2500 to prevent unstable learing with high variance, resulting in long time to converge.
    - It resets the timestep, but for policies that are using state histories, timestep needs to be start after the length of history, meaning that it has to have enough history to recall.
  - step()
    - First it checks the last prices of each stock. Then for every stocks, it computes the maximum number of stock that can buy by comparing the last price and current balance. It also checks the currenct holding stocks to know the maximum number of stocks to sell.
    - Then it trades the stock based on the action, but here, if the action exceeds the possible trading number, it just clips the action to the maximum value. That's why at the test plot, the agent is keep selling 50 stocks even they don't have any.

        - I added trading fee to prevent the agent buying and selling meaninglessly.
        - I also check if it outputs impossible number of traiding(overflow).
- runner.py
    - make_env()
        - By using gymnasium library, I flattend the observation to easily feed it in to PPO network. Also normalized the observation state to make the network learn well. I later save the observation statistics file and load in tester.py code.
        - Using RescaleAction(), I can make the PPO actor net output values between -1 to 1, but still actually using those value in range of maximum tradable stock numbers.
    - I first store every transition data into PPO using Storage class, then update after every epoch to enable stable learning.
- tester.py
    - Its a code to test the policy. It make the environment similar way with runner does. It loads pre-trained policy and also the pre-saved observation state statistics for normalization.
- ppo.py
    - It has Actor for actor network, Critic for critic network, Storage to save all the transition data in buffer and then rollout later to update ppo.
    - _train_step()
        - It computes deltas(td error) and advantage from transition data in compute_returns() function in Storage class.
        - Then, my rolling out these datas, it computes the ratio of action log probability between old policy and updated policy, which means the importance factor. Then it compute the surrogate loss by multiplying advantage, then clip the loss. Finally adding the entrophy loss, we step the optimizer.
        - I added the value loss here, but it is independent with the ppo loss, and it only back propagates to value net.
- dqn.py
    - DQN is only possible for discrete actions, and the number of discrete actions explode if there are many tickers. So first, I discretize the actions into 21 steps, which means with maximum traiding limitation of 50 stocks, each step is a 5 unit trade.
    - For multi ticker system, I implemented several action net heads, independent network for each tickers. This makes the DQN algorithm possible for large number of discrete actions.

---

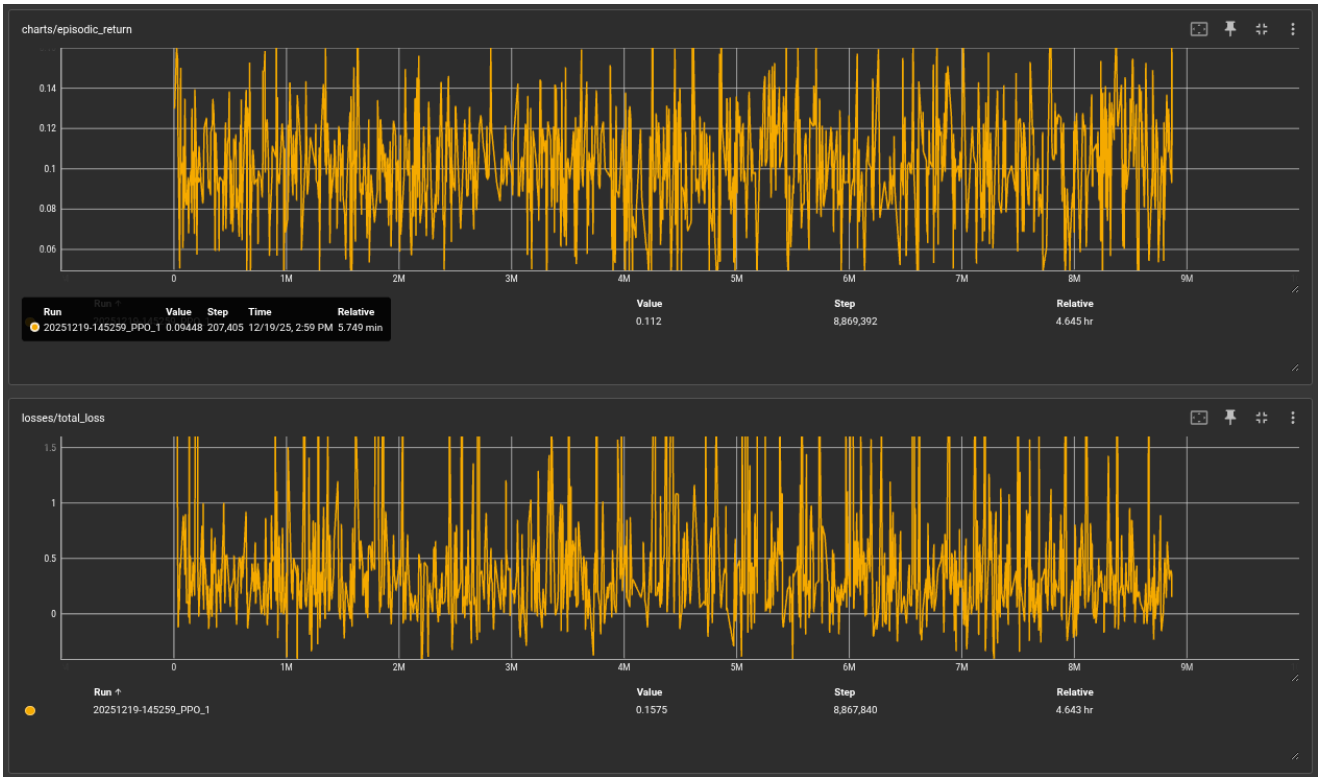
2. Experiments - Performance Upgrade, Single Ticker

Baseline Setup

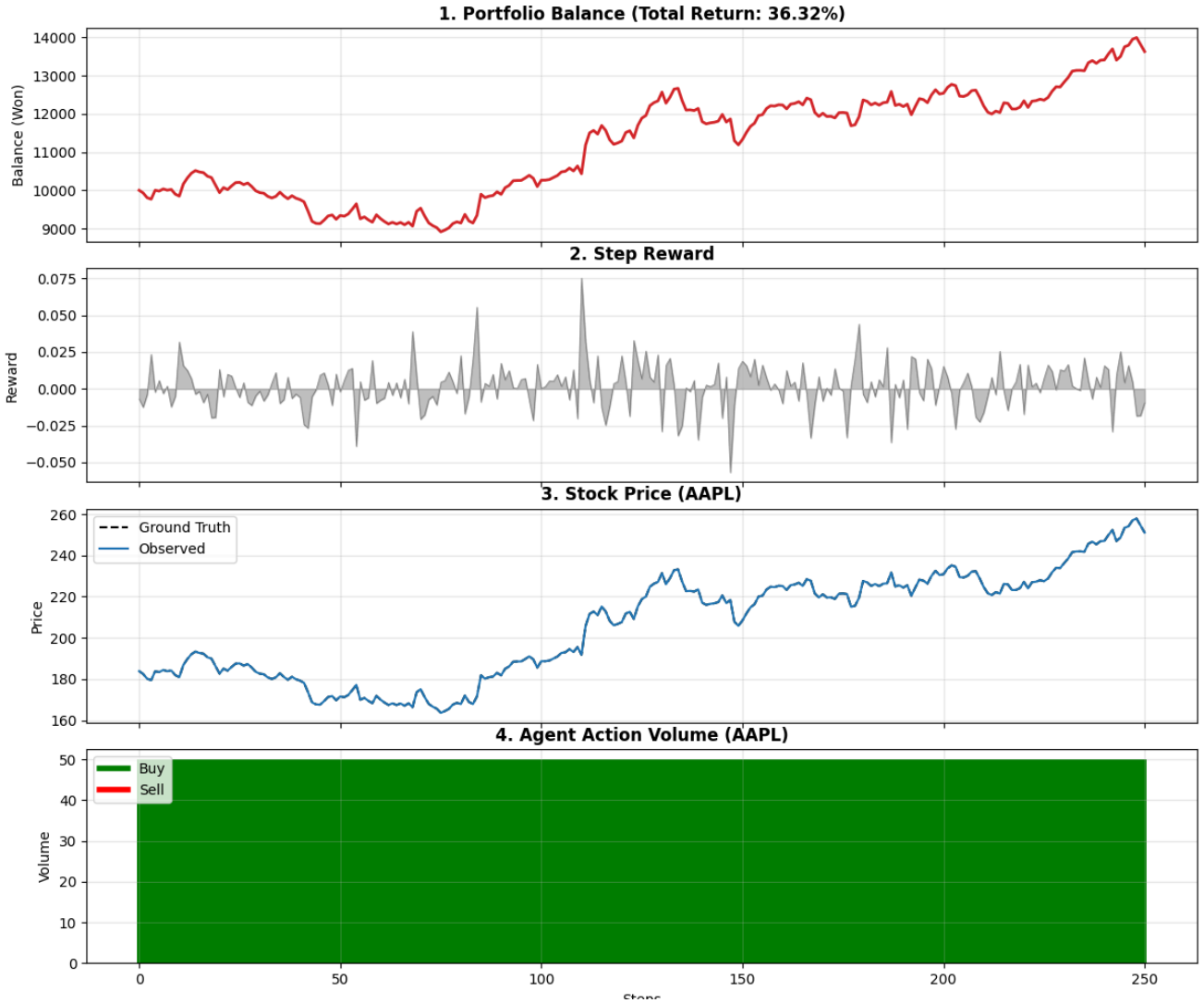
- Observation States
  - agent
    - current balance
    - currently holding stock's number
  - market
    - open price
    - high price
    - low price
    - close price
    - volume
- Reward Function
  - 1 day portfolio difference devided by initial balance
- Dataset
  - 2019-2023 APPL stock
- Algorithm
  - PPO

Results

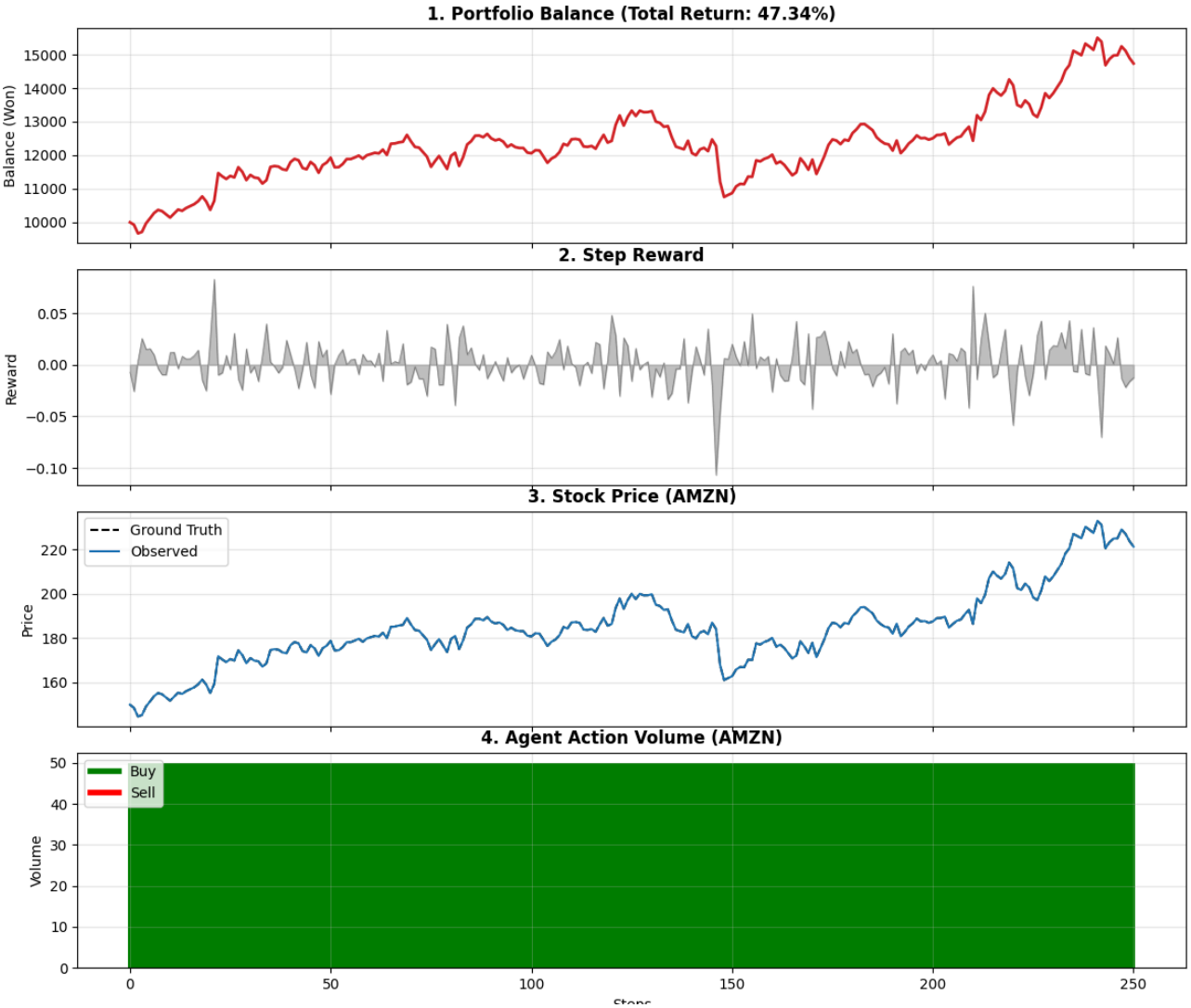
- **Learning graph**



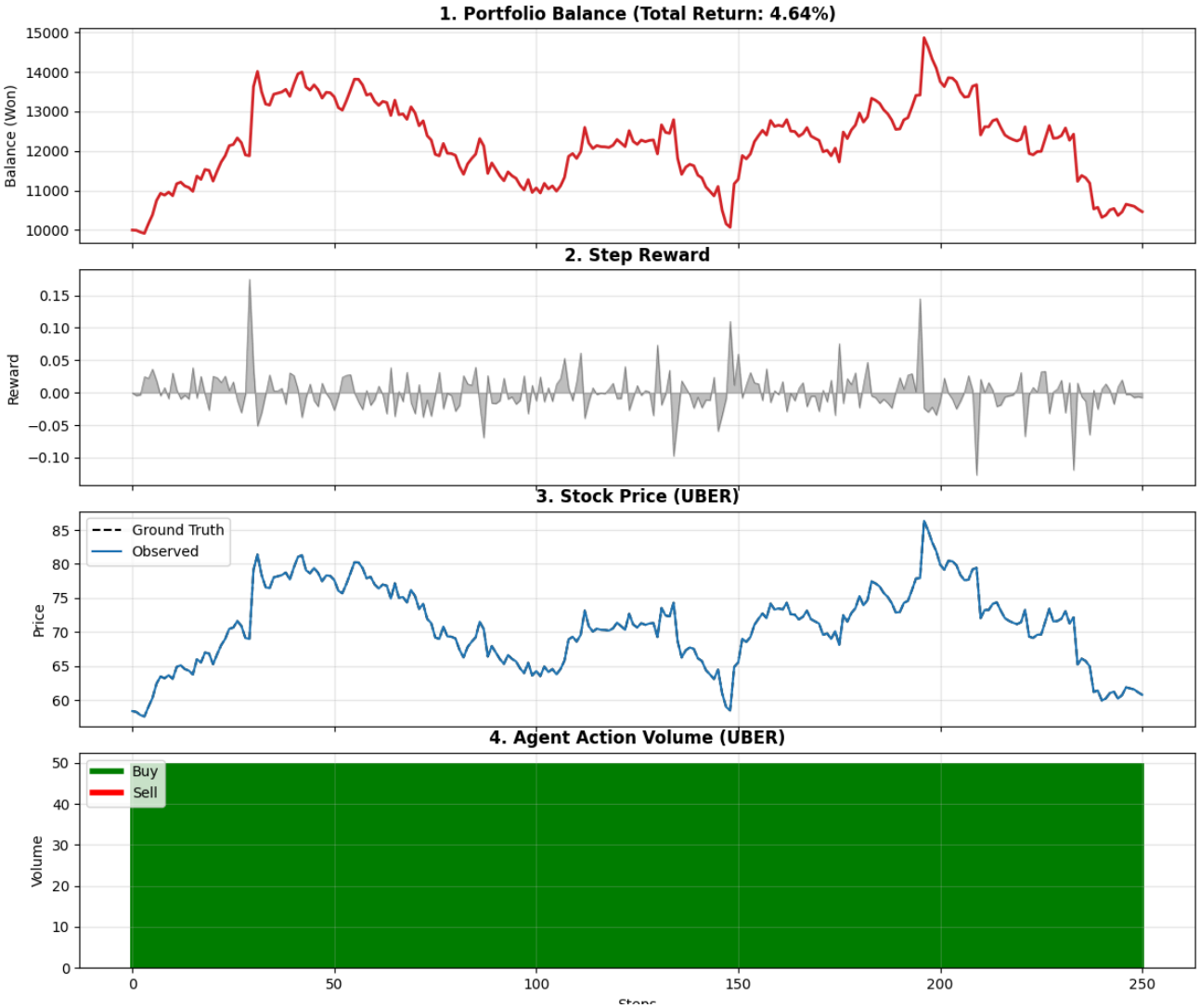
- **Same stock; APPL 2024**



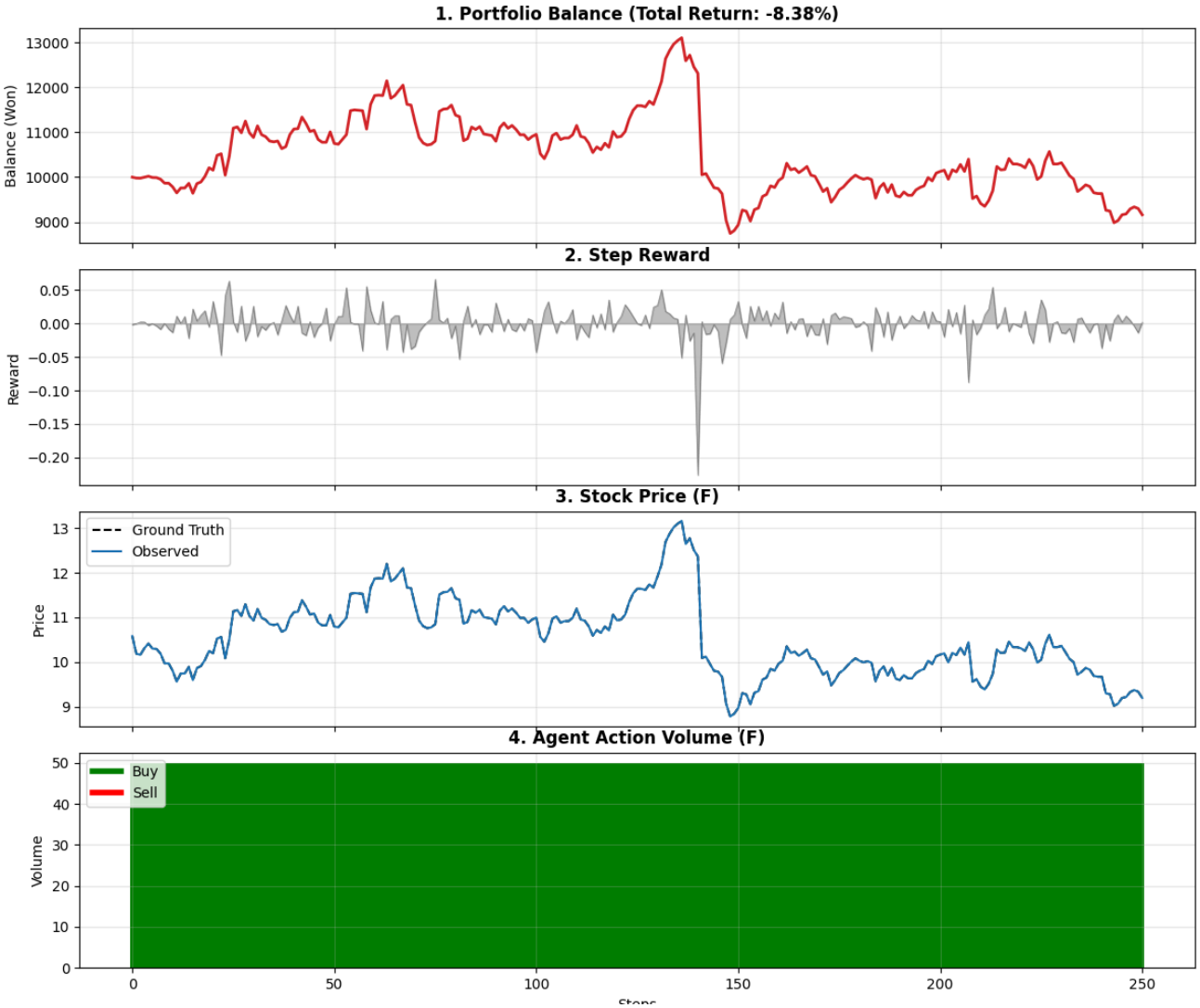
- **Different stock; AMZN 2024

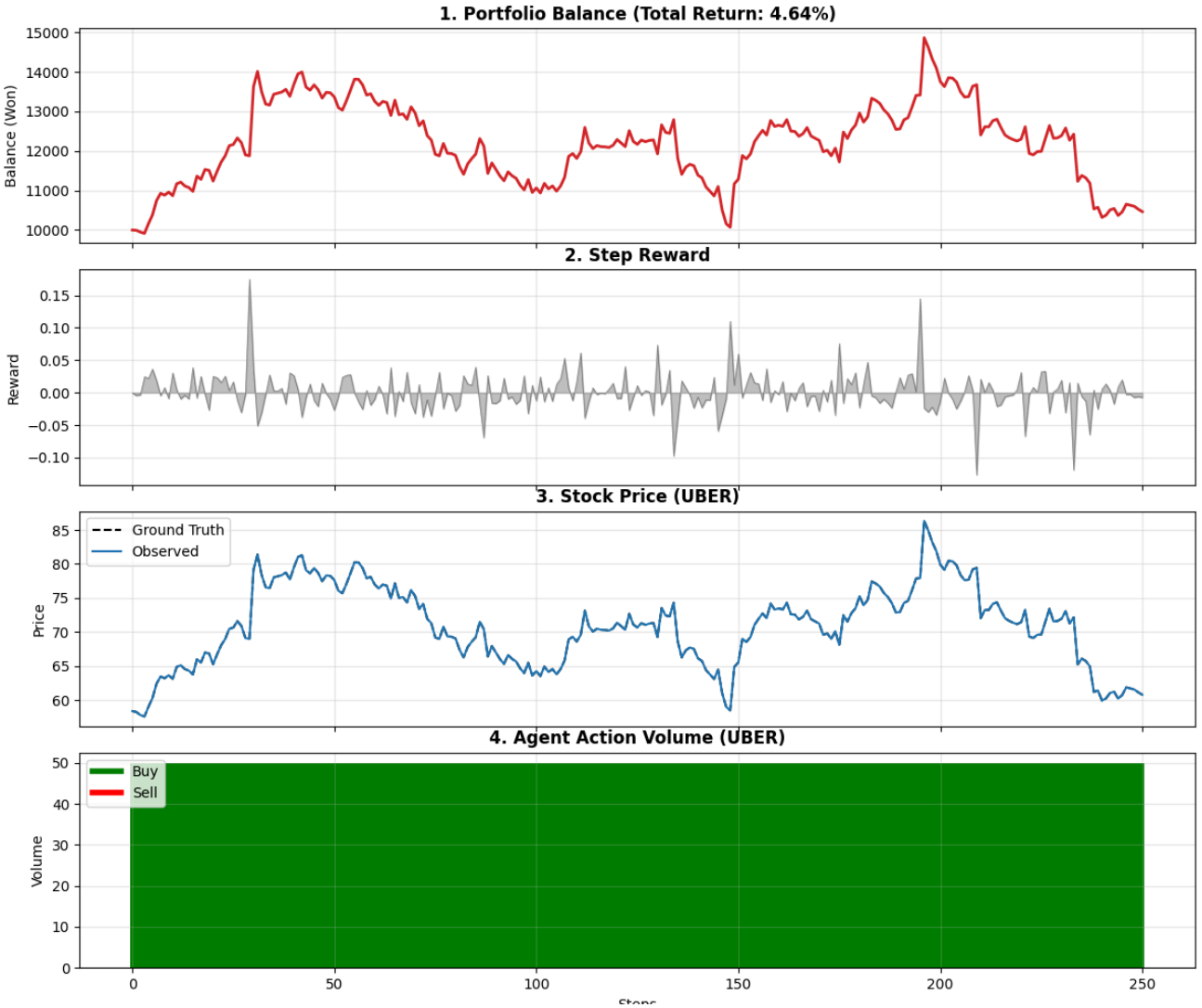


- **Different stock; UBER 2024**



- **Different stock; F 2024**


1. Portfolio Balance (Total Return: -8.38%)
Balance (Won)
13000
12000
11000
10000
9000
2. Step Reward
Reward
0.05
0.00
−0.05
−0.10
−0.15
−0.20
3. Stock Price (F)
Ground Truth
Observed
Price
13
12
11
10
9
4. Agent Action Volume (F)
Buy
Sell
Volume
50
40
30
20
10
0
0
50
100
150
200
250



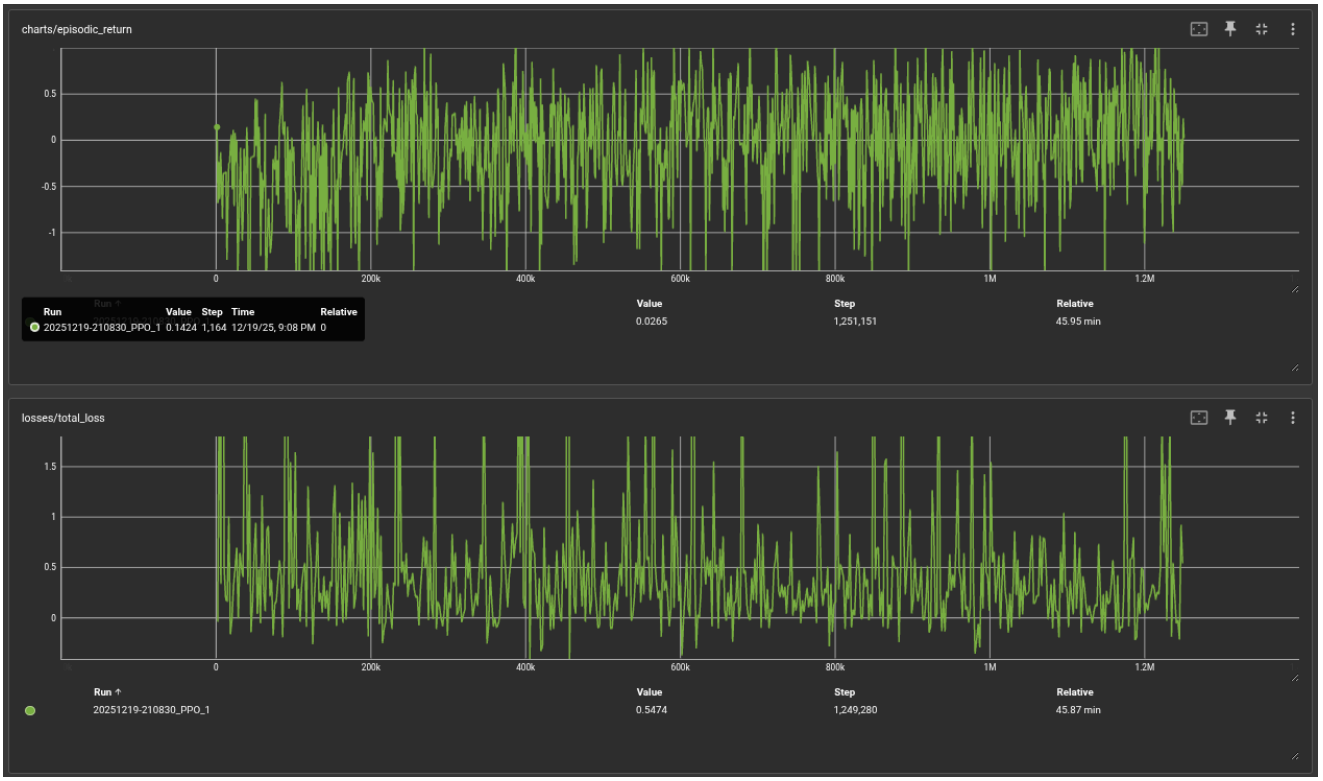
Advanced Setup

- Observation States
  - stack of observation history for 5 timestep
  - agent
    - current balance devided by initial balance
    - number of holding stocks
  - market
    - price over maximum price ever seen
    - target price orver actual price
    - open price
    - high price
    - low price
    - close price
    - volume
    - Relative Strength Index(RSI)
    - Trend Placeholder

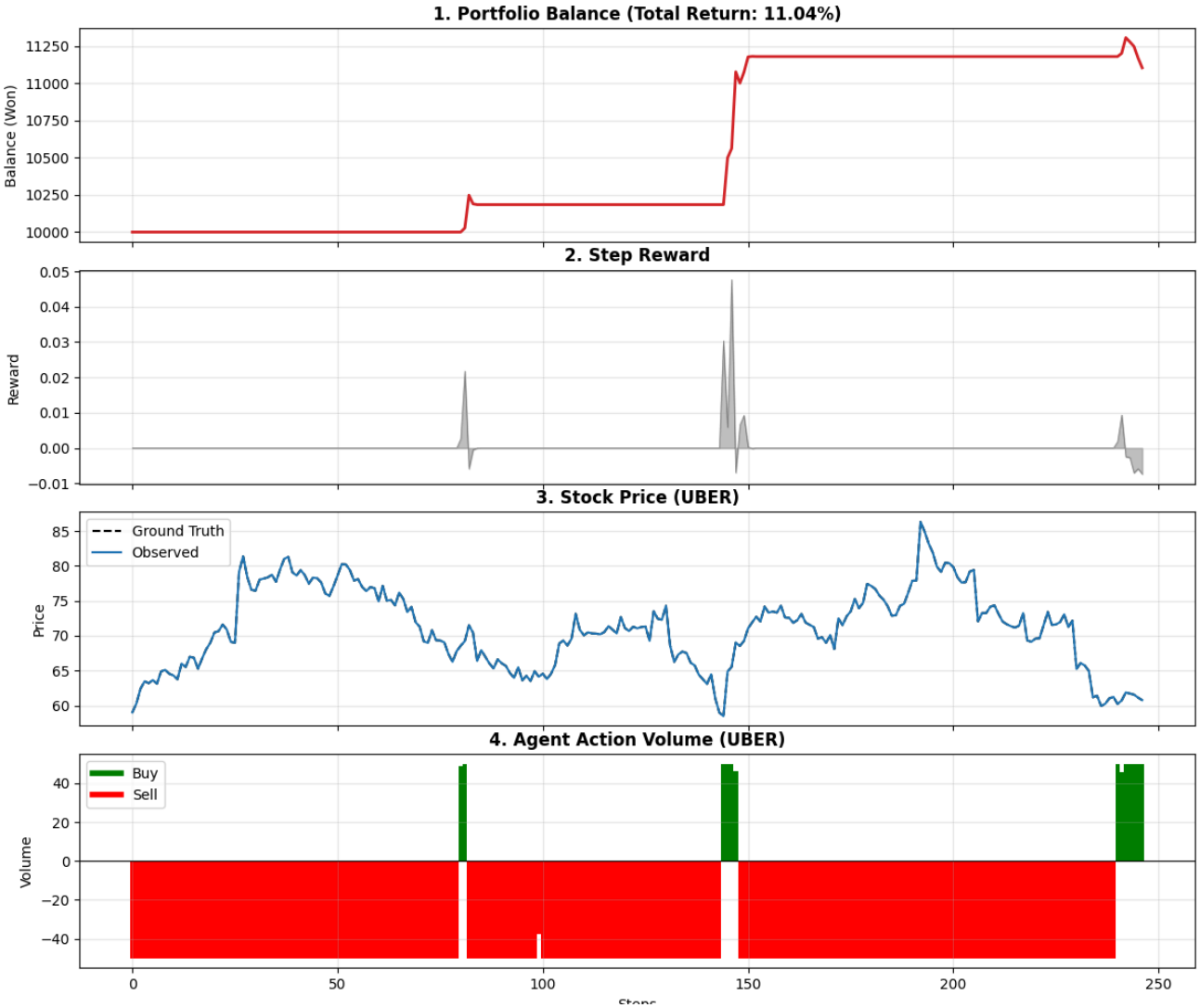
    - P/E Ratio
    - Divedend Yield
- Reward Function
    - Log return
    - Action penalty
- Dataset
    - 2019-2023 UBER stock

Results

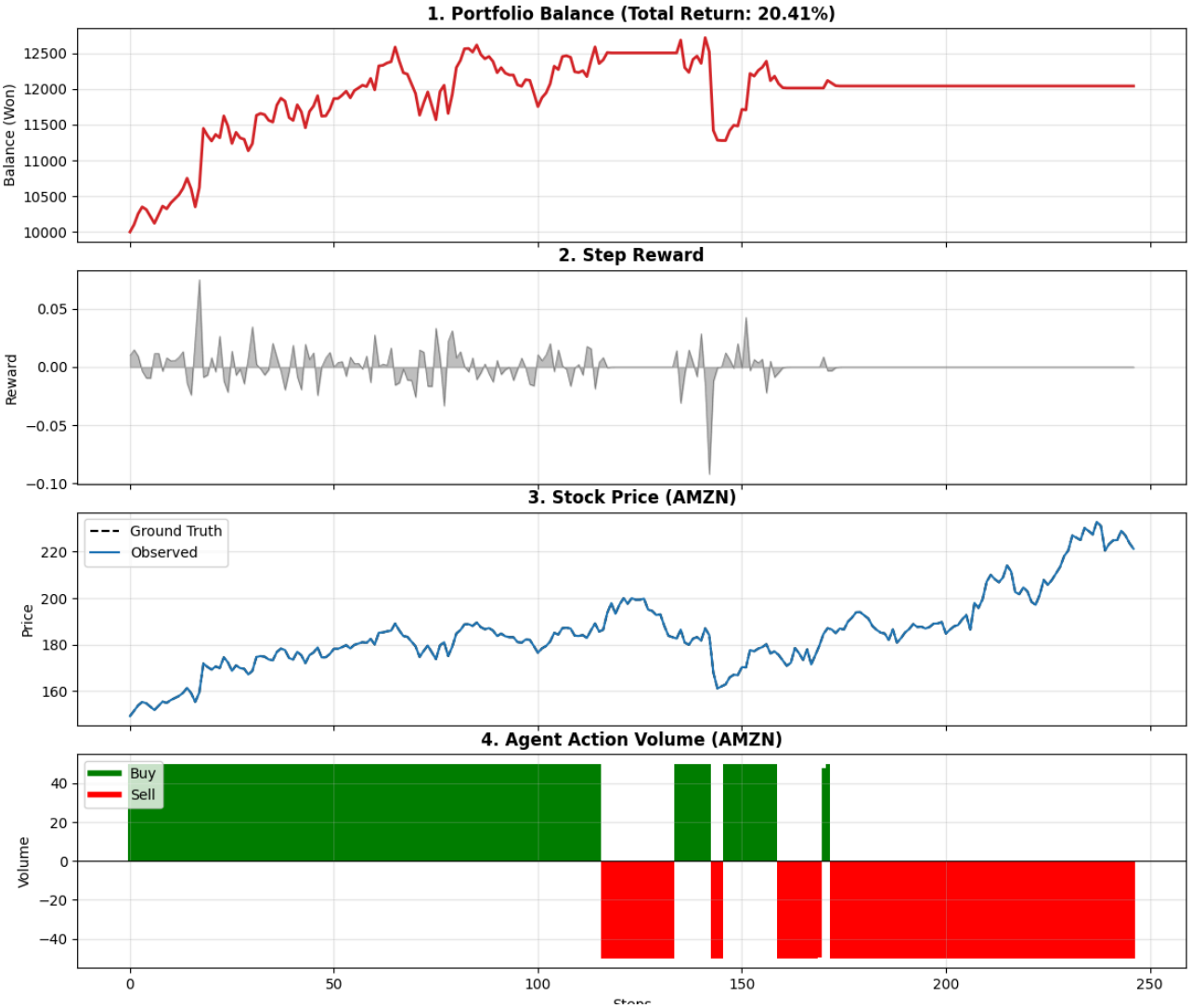
- **Learning graph**



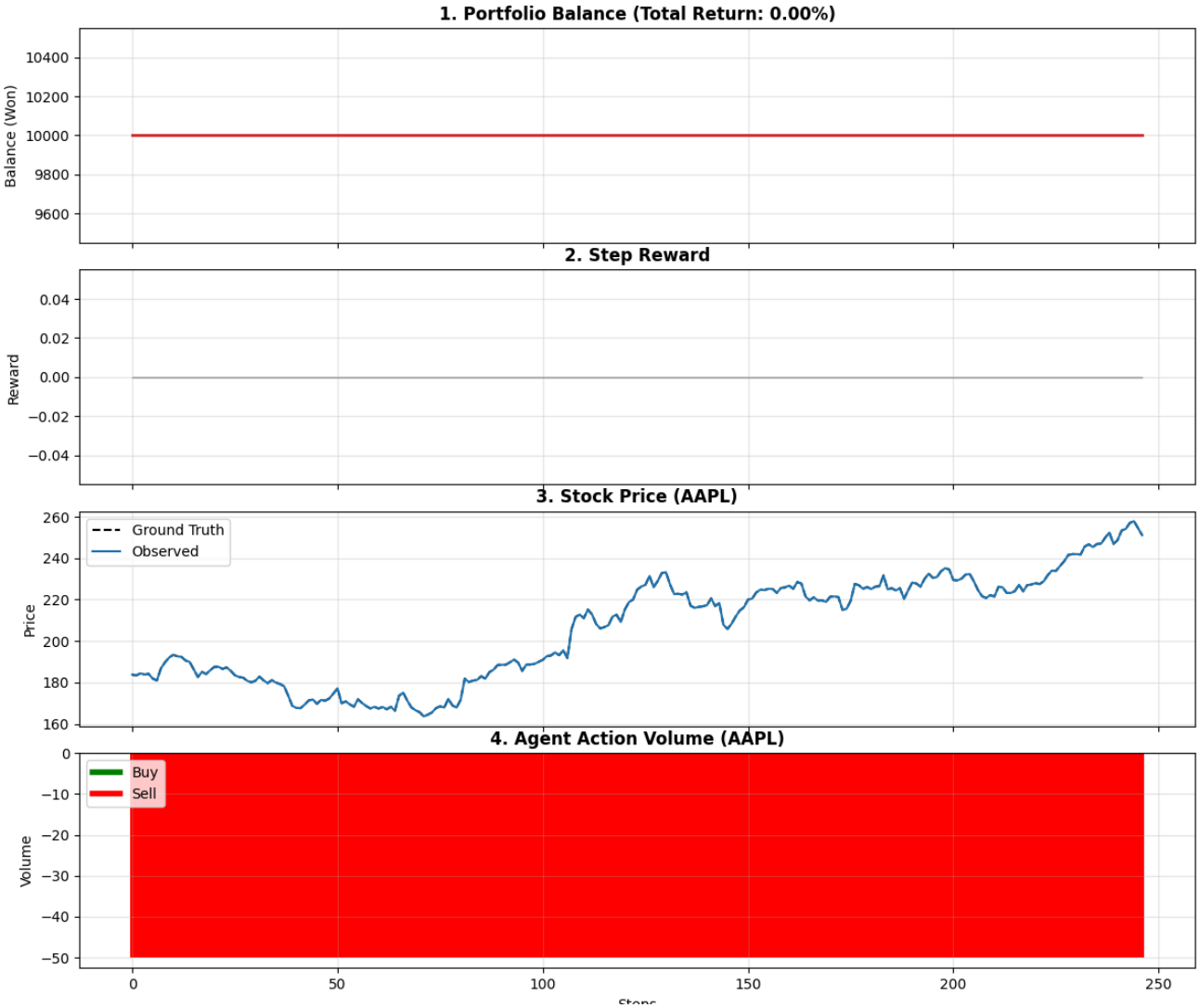
- **Same stock; UBER 2024**



- **Different stock; AMZN 2024**

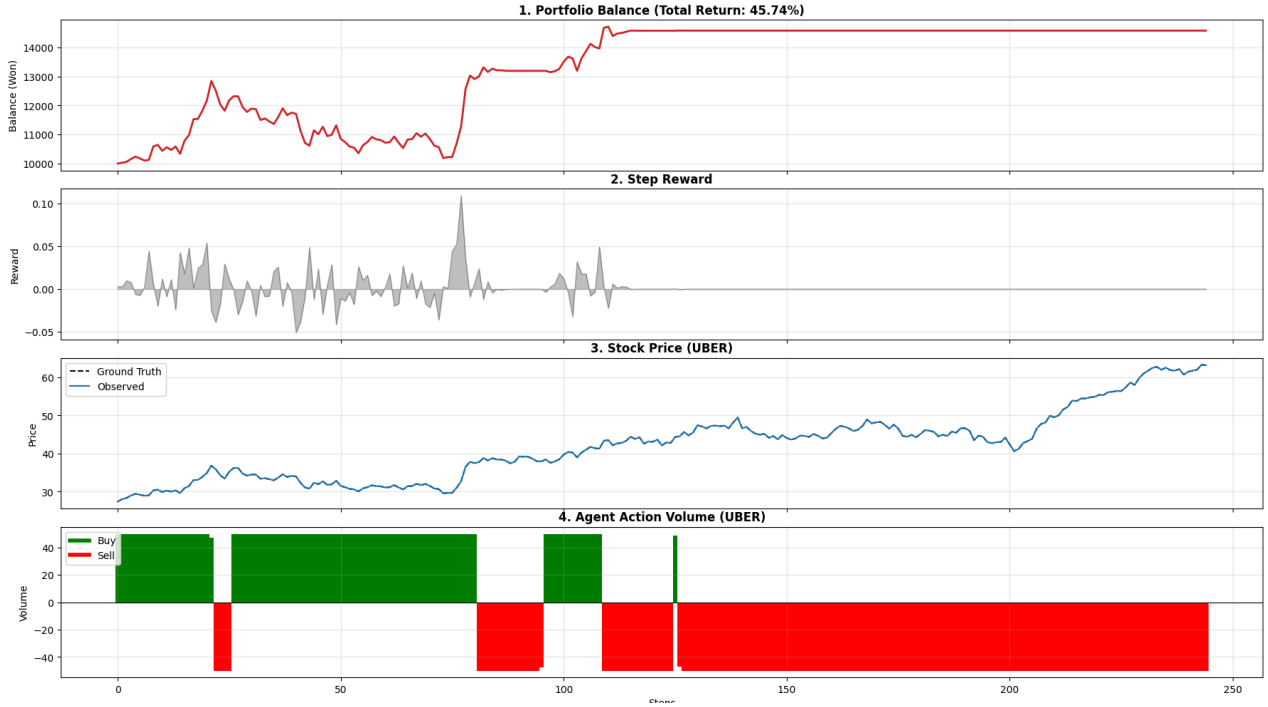


- **Different stock; APPL 2024**



- **Different stock; F 2024**

- **Same stock, same period; UBER 2023**



Discussion

- At first I learned the policy only using tech-related stocks, which are all increasing highly between 2019-2023. The tendency also continued in 2024. So by training the policy using tech-relating monotonously increasing stocks, the policy might not learn about the rule of up and down, but just buy and wait. It can make the policy impossible to handle with non-tech stokcs. As you can see in the baseline policy, it works pretty well with tech-related rising stocks, but it fails with Uber and Ford.
- However, the advanced policy learned more general policy with knowing about the tendency of the stock price, using the history of observation, non-monotonous data, and from various market related values. So it behaves better for Uber, F 2024 datasets.
- By knowing these values, the agent doesn't just predict the stock by looking at the current price, but actually understanding the value of the company.
- Analysts' price standard, Relative Strength Index(RSI) and P/E Ratio can also let the policy know about if the stock is over-evaluated or under-evaluated.
- I add the action penalty term to make the action smoother, in intention of making the policy to do a long-term investment. However, it didn't work well, but it actually somehow increased the tendency to stay at fully buying all stocks, or fully selling all stocks.
- Here, by the way, even the policy's action is to sell 50 stocks, they can only sell they are holding, so the value of the graph is not directly influencing the environment, but it's being clipped.

---

3. Experiments - Performance Upgrade, Multi Ticker

Baseline Setup

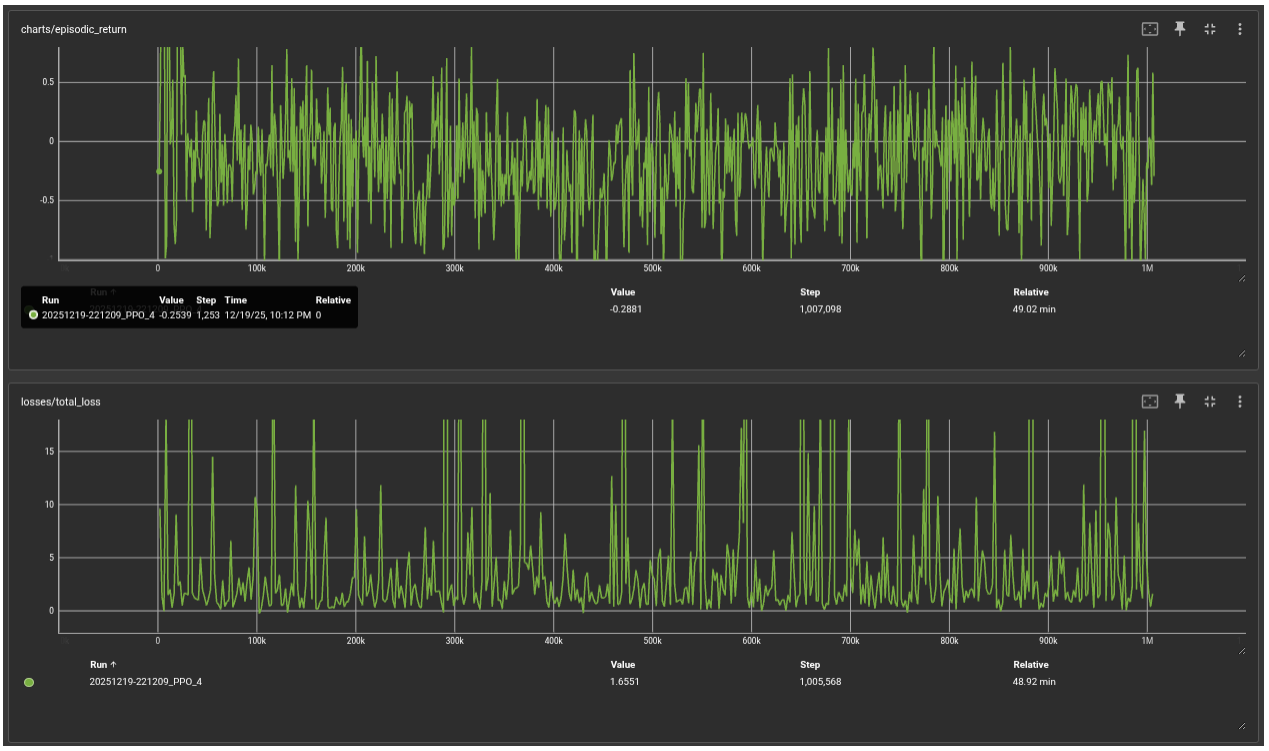
- Same as Single Ticker's

Advanced Setup

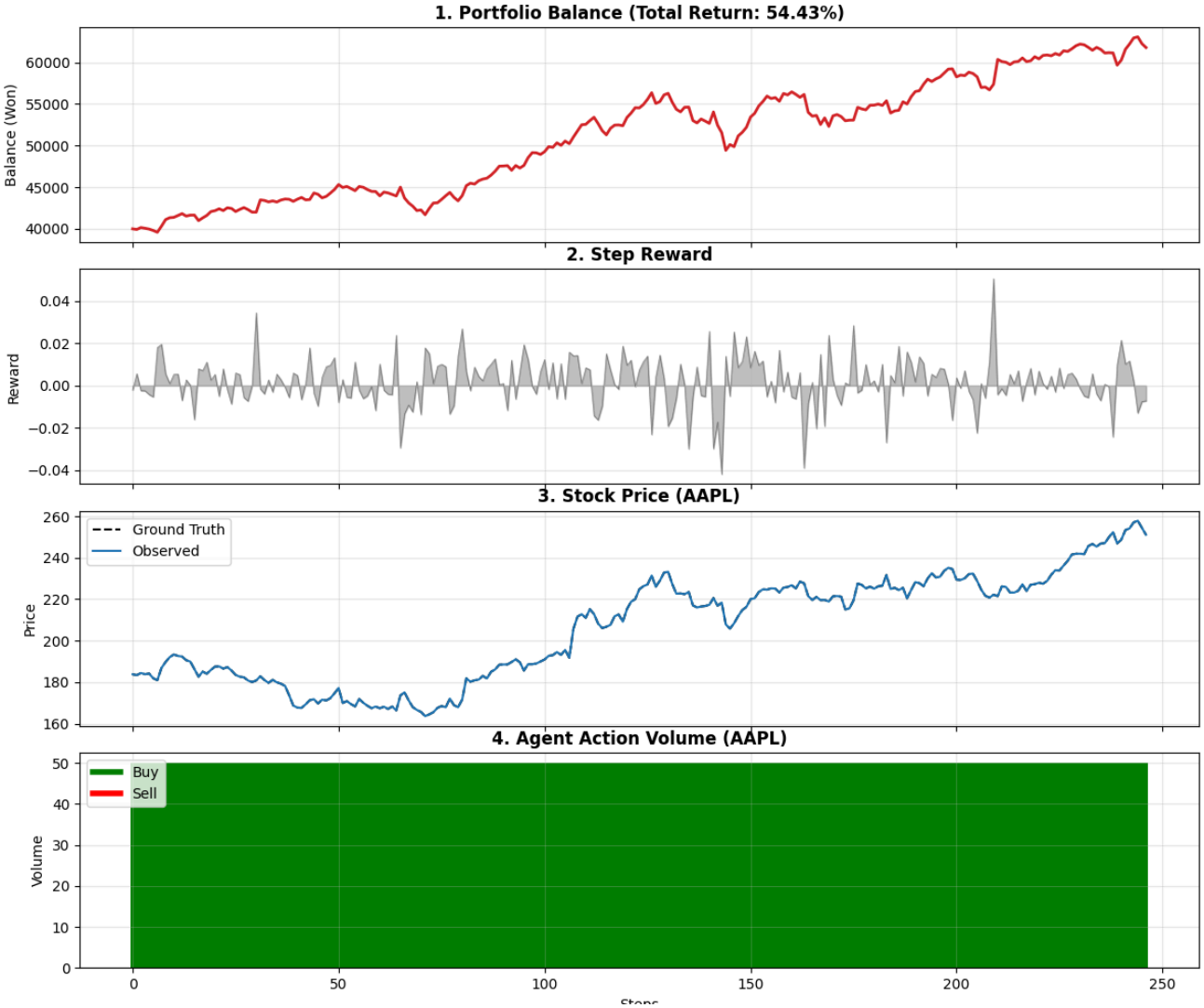
- Same as Single Ticker's

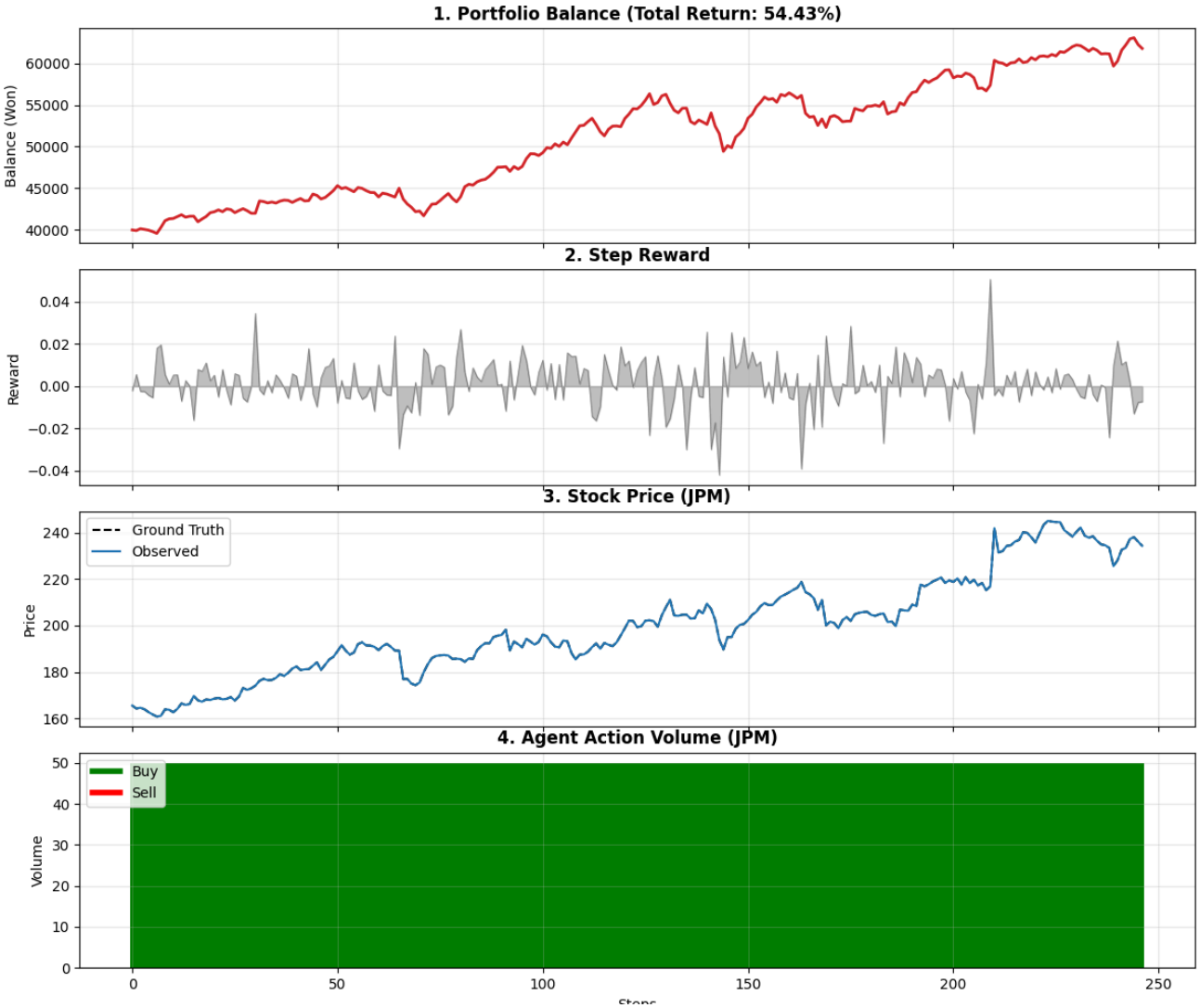
Results

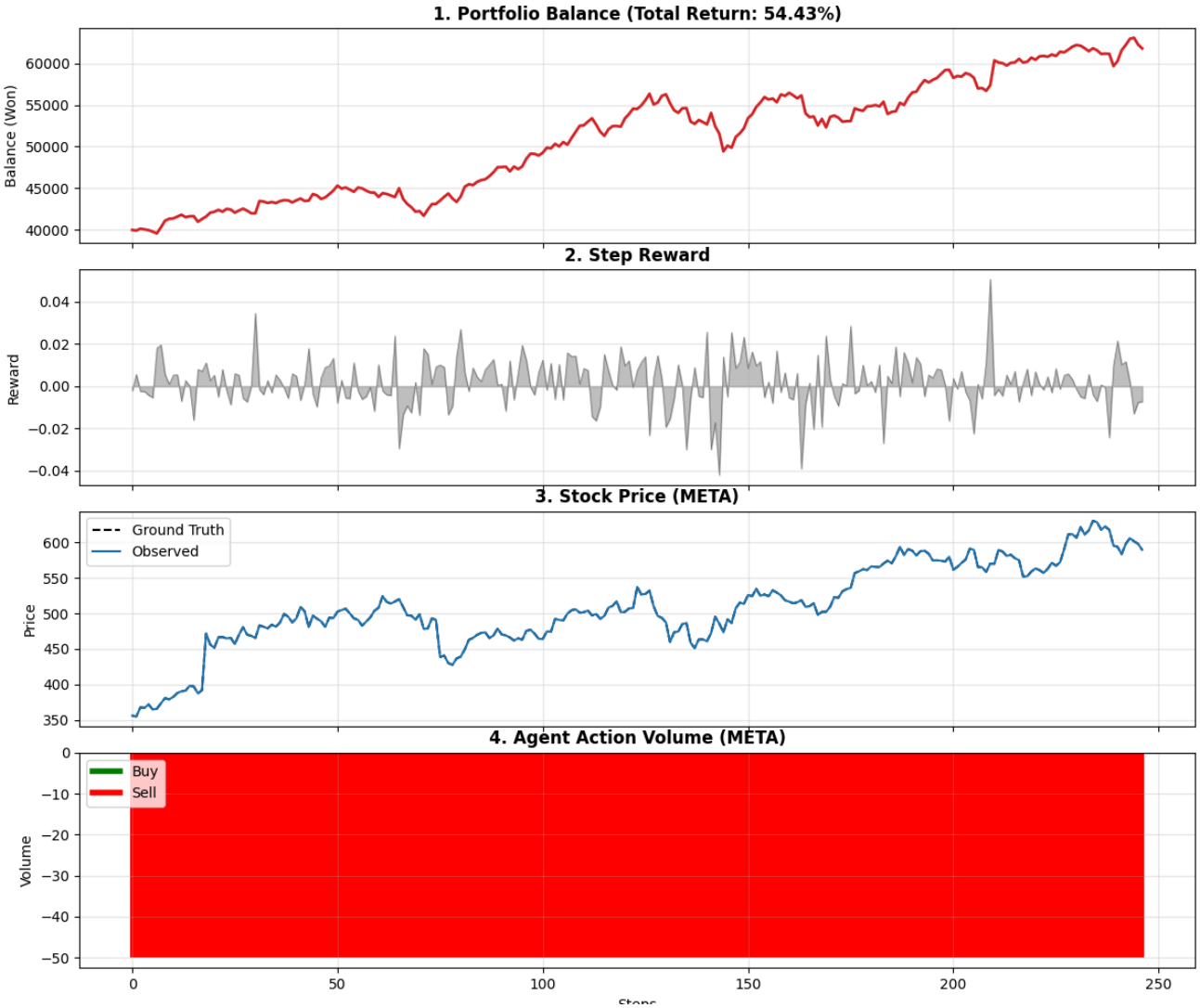
- **Learning Graph**

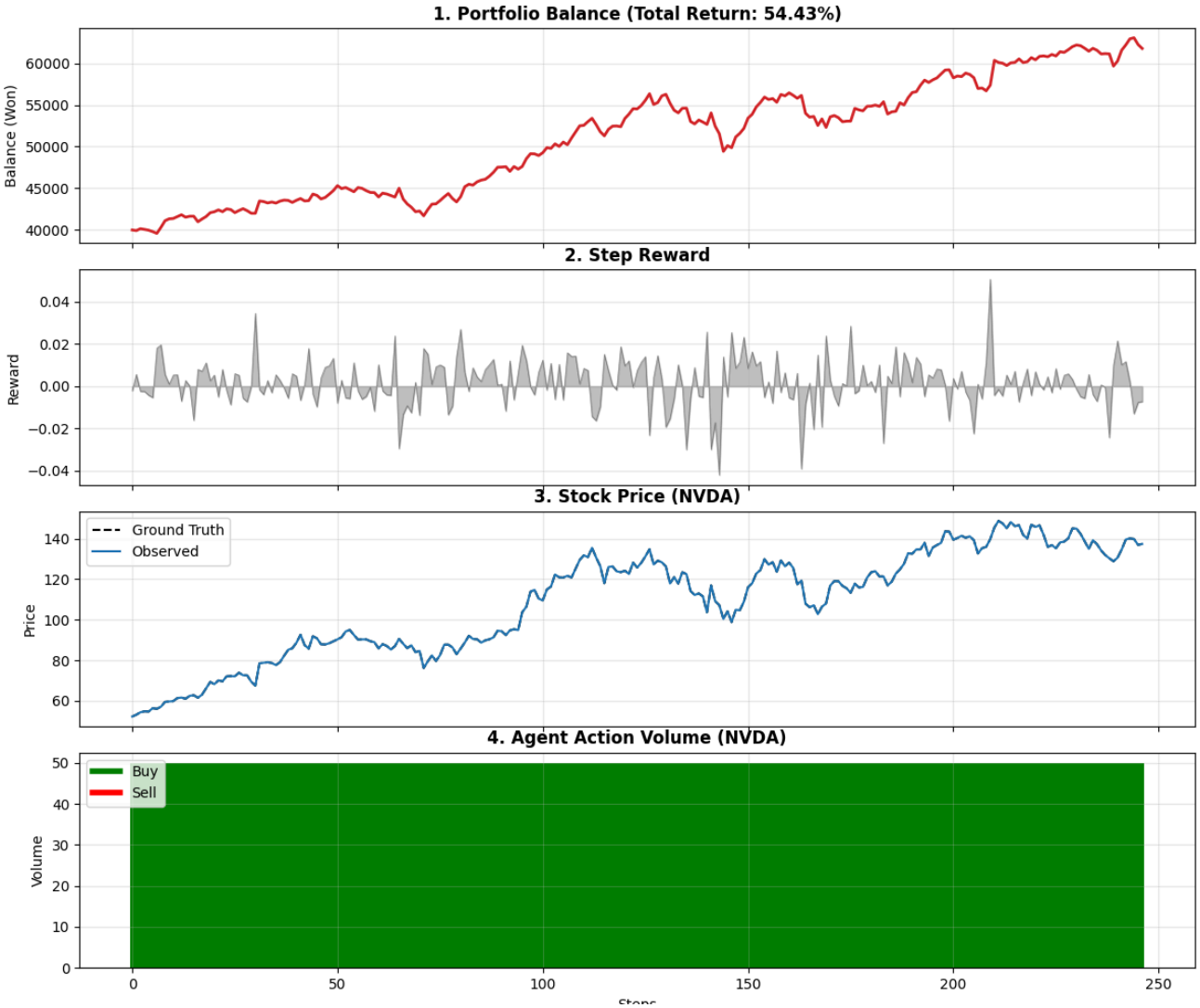


- **Exp1. All same stocks**

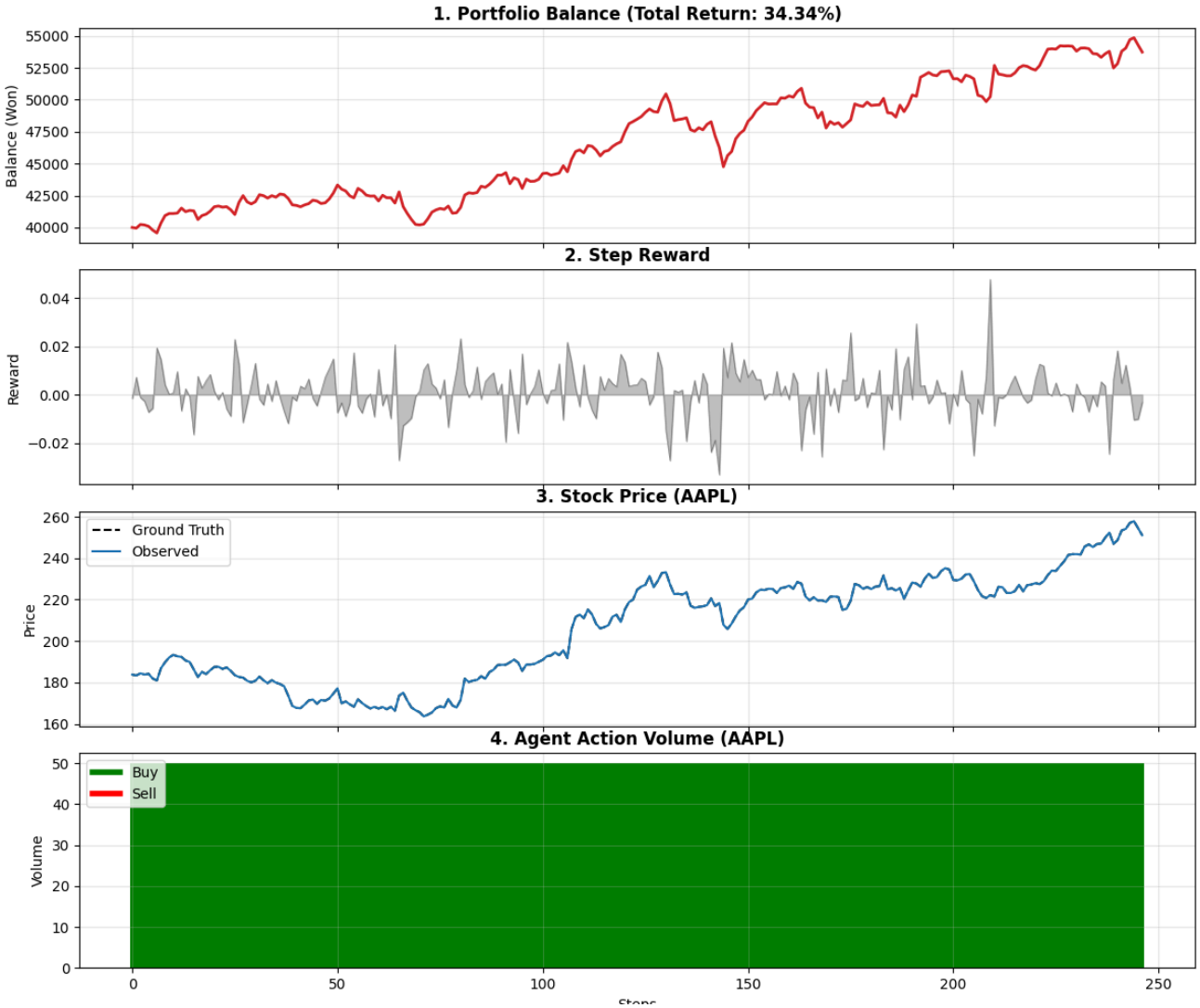


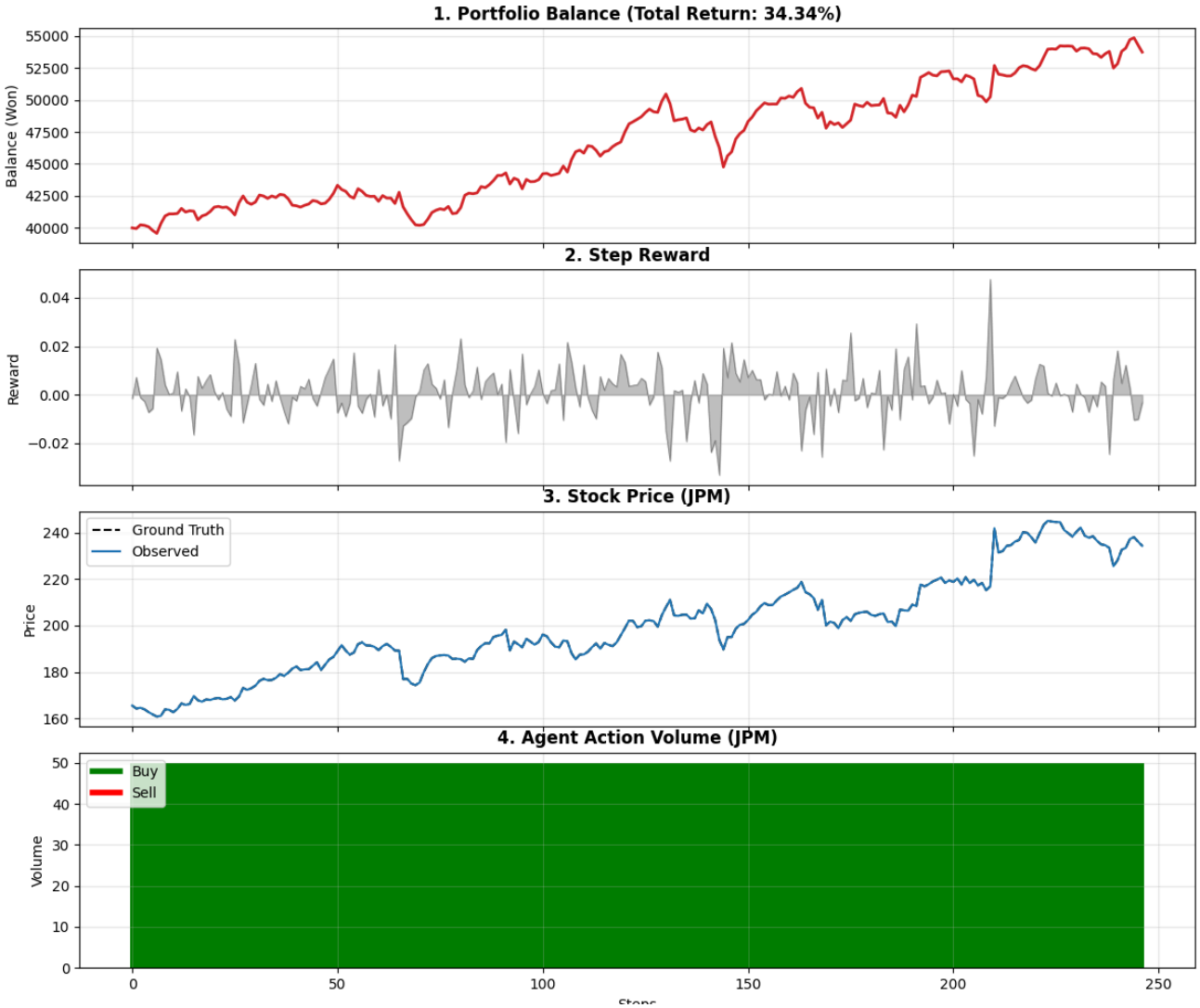

1. Portfolio Balance (Total Return: 54.43%)
Balance (Won)
60000
55000
50000
45000
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
−0.04
3. Stock Price (JPM)
Ground Truth
Observed
Price
240
220
200
180
160
4. Agent Action Volume (JPM)
Buy
Sell
Volume
50
40
30
20
10
0
0
50
100
150
200
250

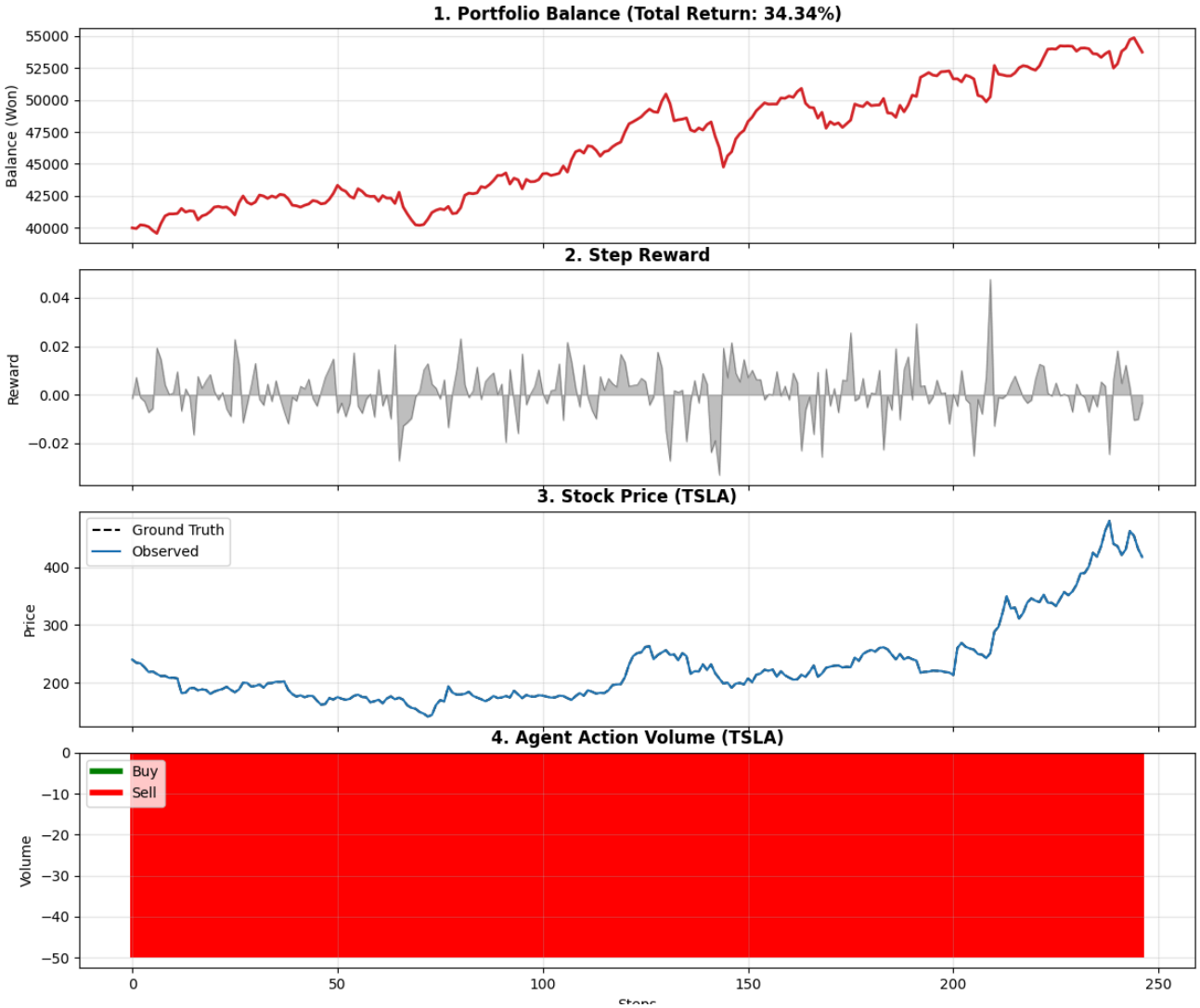

1. Portfolio Balance (Total Return: 54.43%)
Balance (Won)
60000
55000
50000
45000
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
−0.04
3. Stock Price (META)
Ground Truth
Observed
Price
600
550
500
450
400
350
4. Agent Action Volume (META)
Buy
Sell
Volume
0
−10
−20
−30
−40
−50
0
50
100
150
200
250

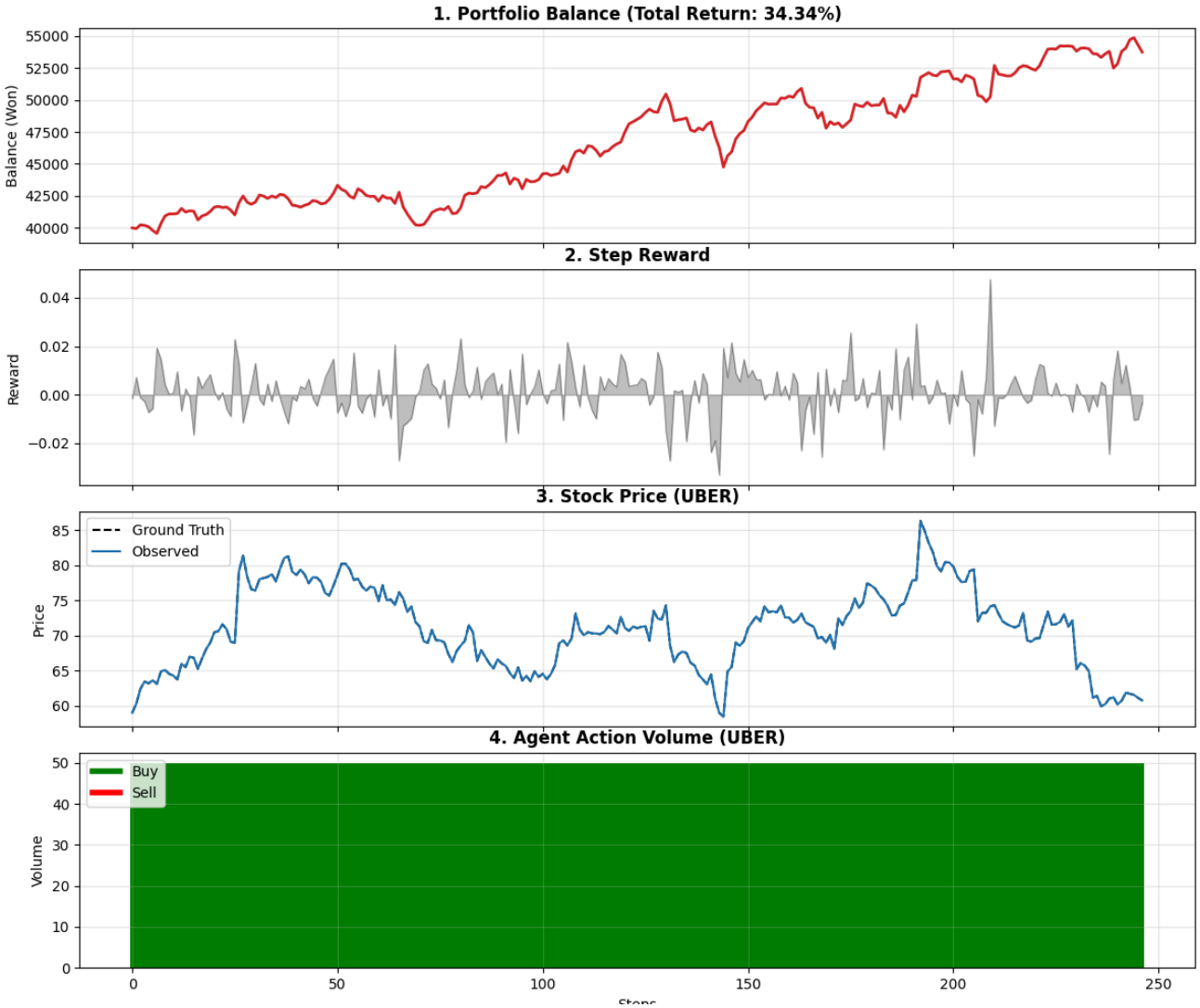


- **Exp2. 2 trained stocks + 2 unseen stocks**

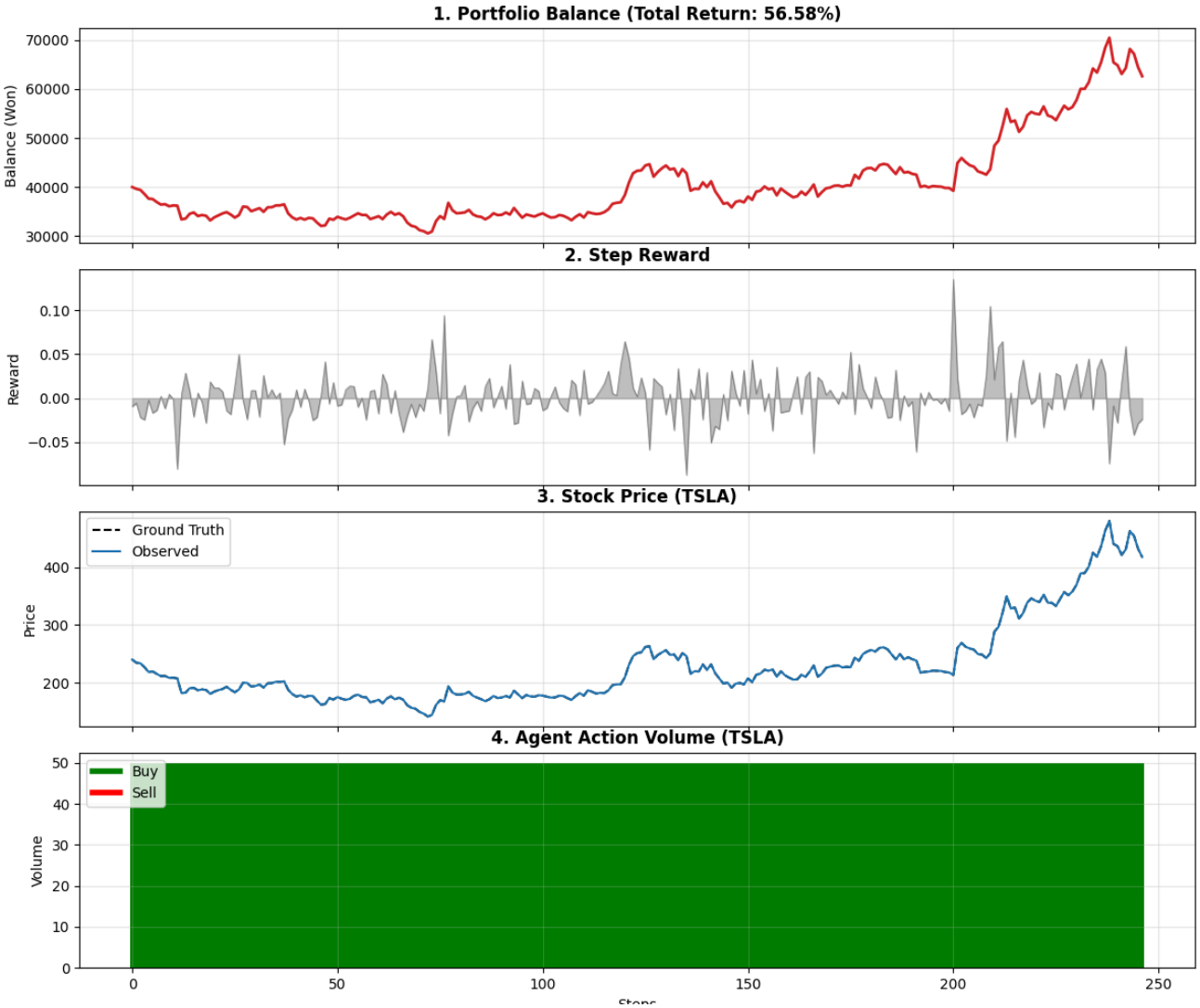

1. Portfolio Balance (Total Return: 34.34%)
Balance (Won)
55000
52500
50000
47500
45000
42500
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
3. Stock Price (AAPL)
Price
Ground Truth
Observed
260
240
220
200
180
160
4. Agent Action Volume (AAPL)
Volume
Buy
Sell
50
40
30
20
10
0
0
50
100
150
200
250

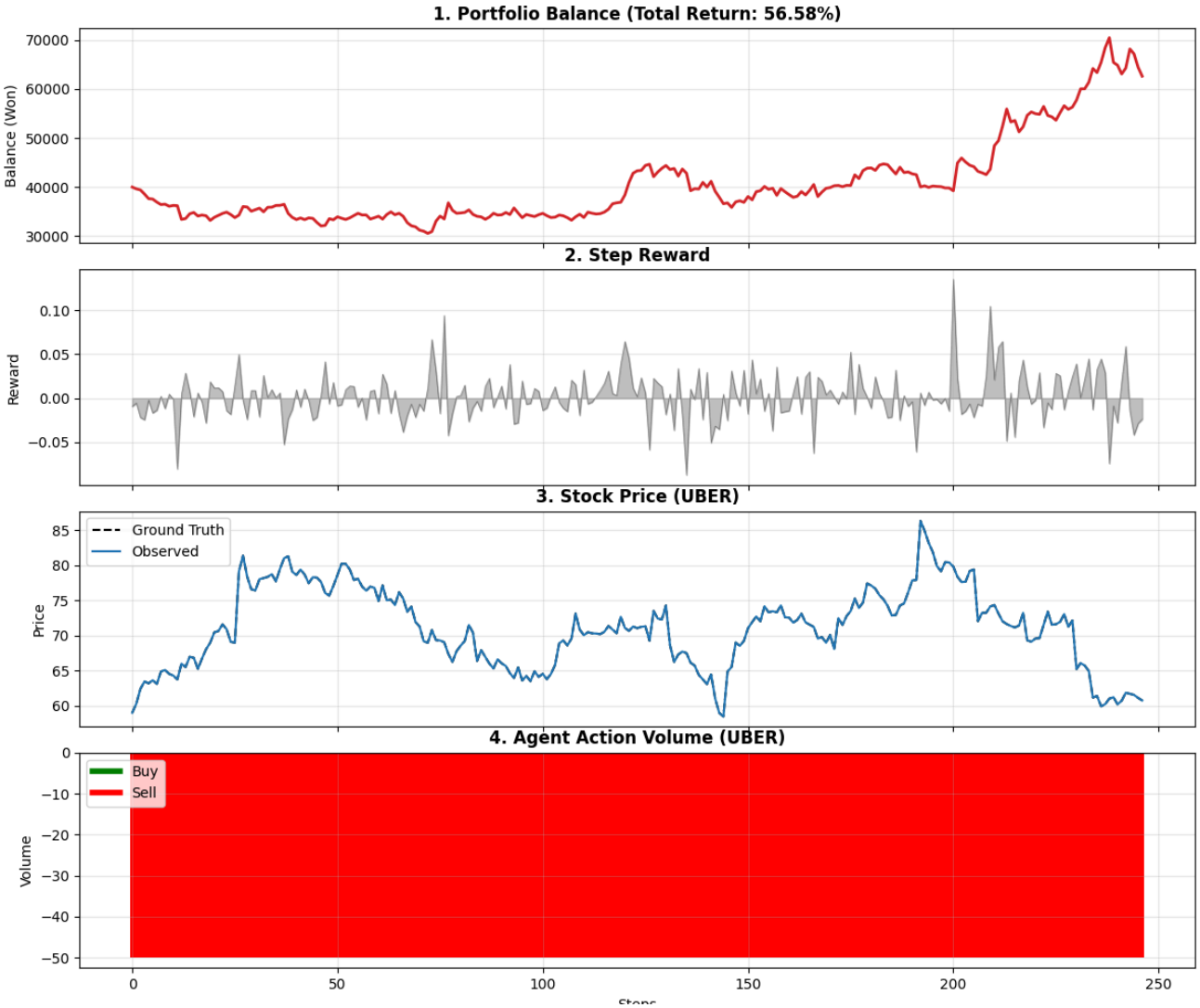

1. Portfolio Balance (Total Return: 34.34%)
Balance (Won)
55000
52500
50000
47500
45000
42500
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
3. Stock Price (JPM)
Ground Truth
Observed
Price
240
220
200
180
160
4. Agent Action Volume (JPM)
Buy
Sell
Volume
50
40
30
20
10
0
0
50
100
150
200
250

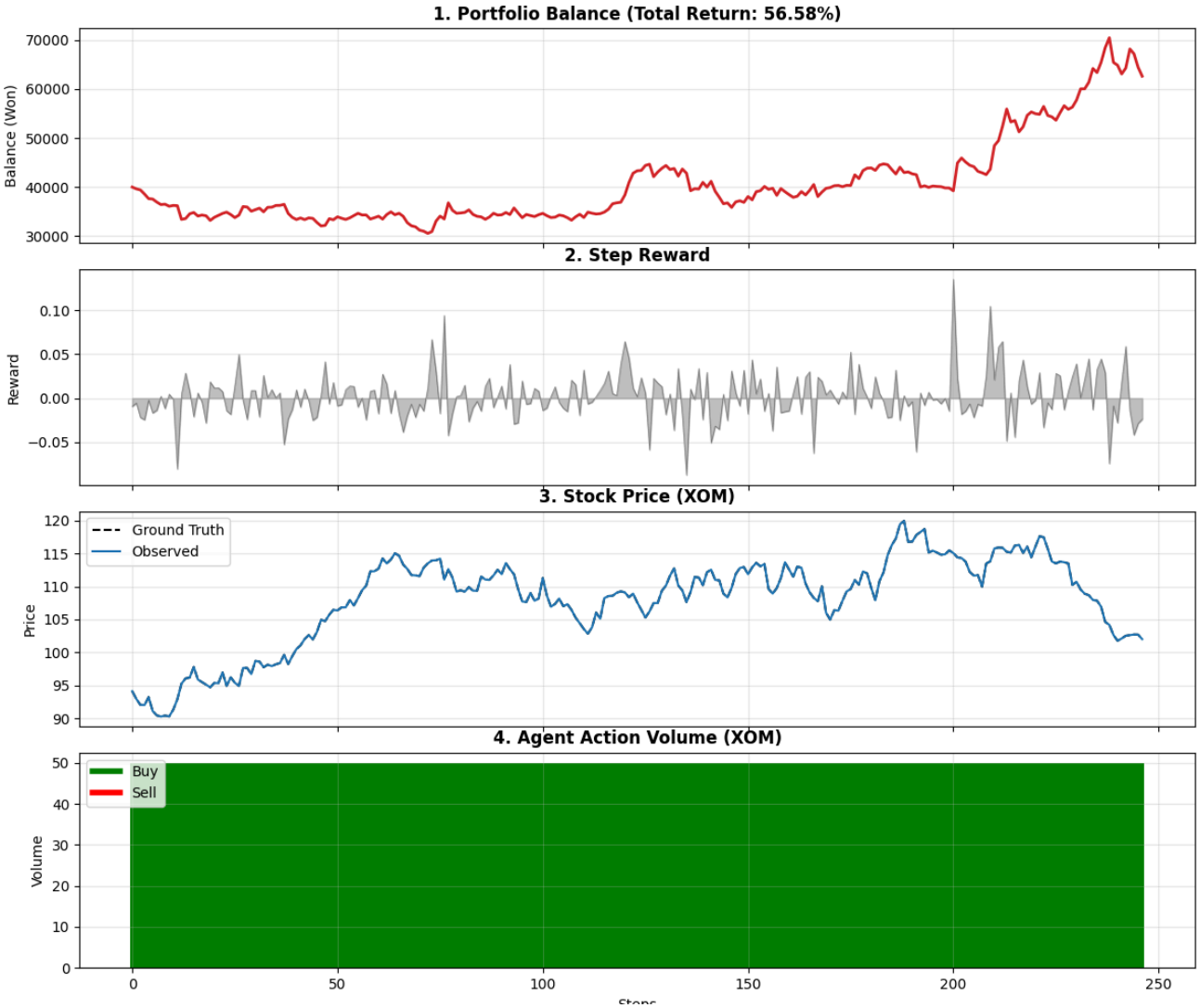

1. Portfolio Balance (Total Return: 34.34%)
Balance (Won)
55000
52500
50000
47500
45000
42500
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
3. Stock Price (TSLA)
Ground Truth
Observed
Price
400
300
200
4. Agent Action Volume (TSLA)
Buy
Sell
Volume
0
−10
−20
−30
−40
−50
0
50
100
150
200
250


1. Portfolio Balance (Total Return: 34.34%)
Balance (Won)
55000
52500
50000
47500
45000
42500
40000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
3. Stock Price (UBER)
Ground Truth
Observed
Price
85
80
75
70
65
60
4. Agent Action Volume (UBER)
Buy
Sell
Volume
50
40
30
20
10
0
0
50
100
150
200
250

- **Exp3. All unseen stocks**


1. Portfolio Balance (Total Return: 56.58%)
Balance (Won)
70000
60000
50000
40000
30000
2. Step Reward
Reward
0.10
0.05
0.00
−0.05
3. Stock Price (TSLA)
Ground Truth
Observed
Price
400
300
200
4. Agent Action Volume (TSLA)
Buy
Sell
Volume
50
40
30
20
10
0
0
50
100
150
200
250


1. Portfolio Balance (Total Return: 56.58%)
Balance (Won)
70000
60000
50000
40000
30000
2. Step Reward
Reward
0.10
0.05
0.00
−0.05
3. Stock Price (UBER)
Ground Truth
Observed
Price
85
80
75
70
65
60
4. Agent Action Volume (UBER)
Buy
Sell
Volume
0
−10
−20
−30
−40
−50
0
50
100
150
200
250



Discussion

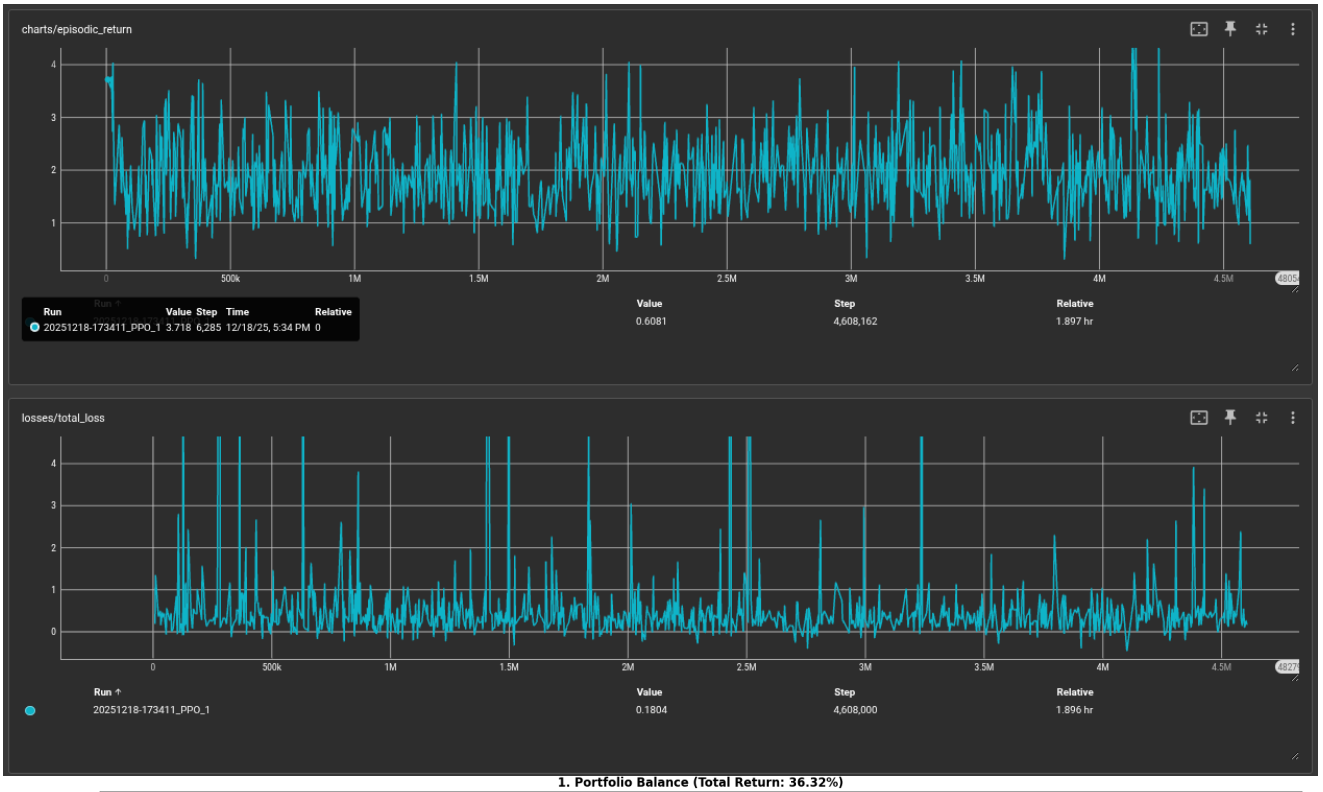
- The improvement of the performance based on the change of states and reward would be similarly discussed as previous discussion. So I want to talk the overall tendency of all policies in this whole project.
- We are giving very small dataset to each policy. Just 1 or up to 4 stocks, and up to 10years (about 2500 trading days). And the datasets that I selected are all tech-related dataset, and recent years, almost all tech-related stocks are growing. Somes are even sky-rocketing in several days.
- It means, we are giving too-biased dataset to policies, so the policy learns to just buy all at the first time, then keep it, which might be a fair stategy within recent tech-related stocks.
- However, it fails to learn the actual principal of trading, considering the value of the company, or interpret and predicting the rise or fall of the stock price in long term or even in short term.
- So I would like to leave this as a future work. The main thing to try to improve the performance of the policy, and making the policy to act diversely based on the observation state, the dataset needs to be more larger with various tendencies.

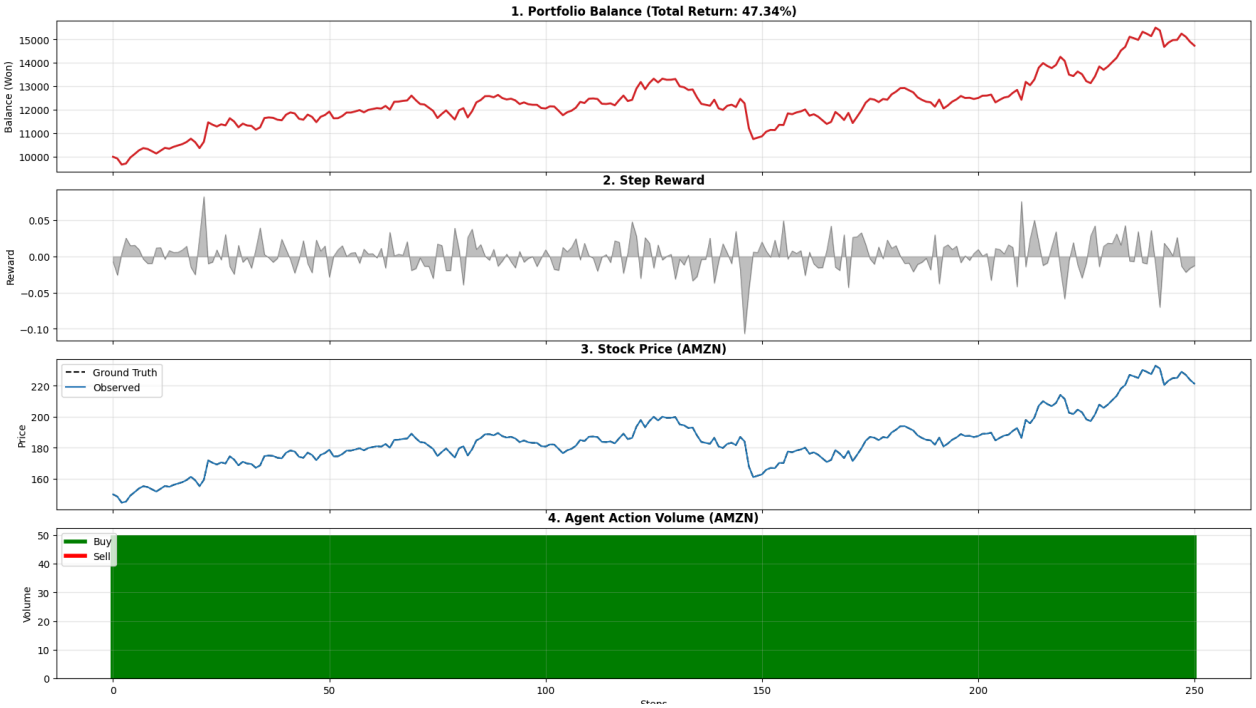
4. Experiments - Algorithm Comparison

Setup

- Same as baseline

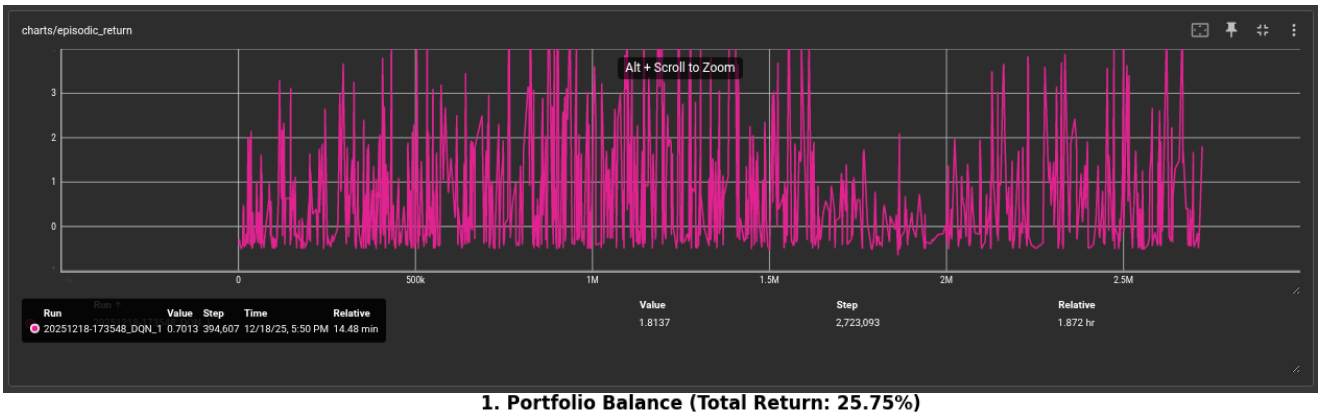
PPO Results

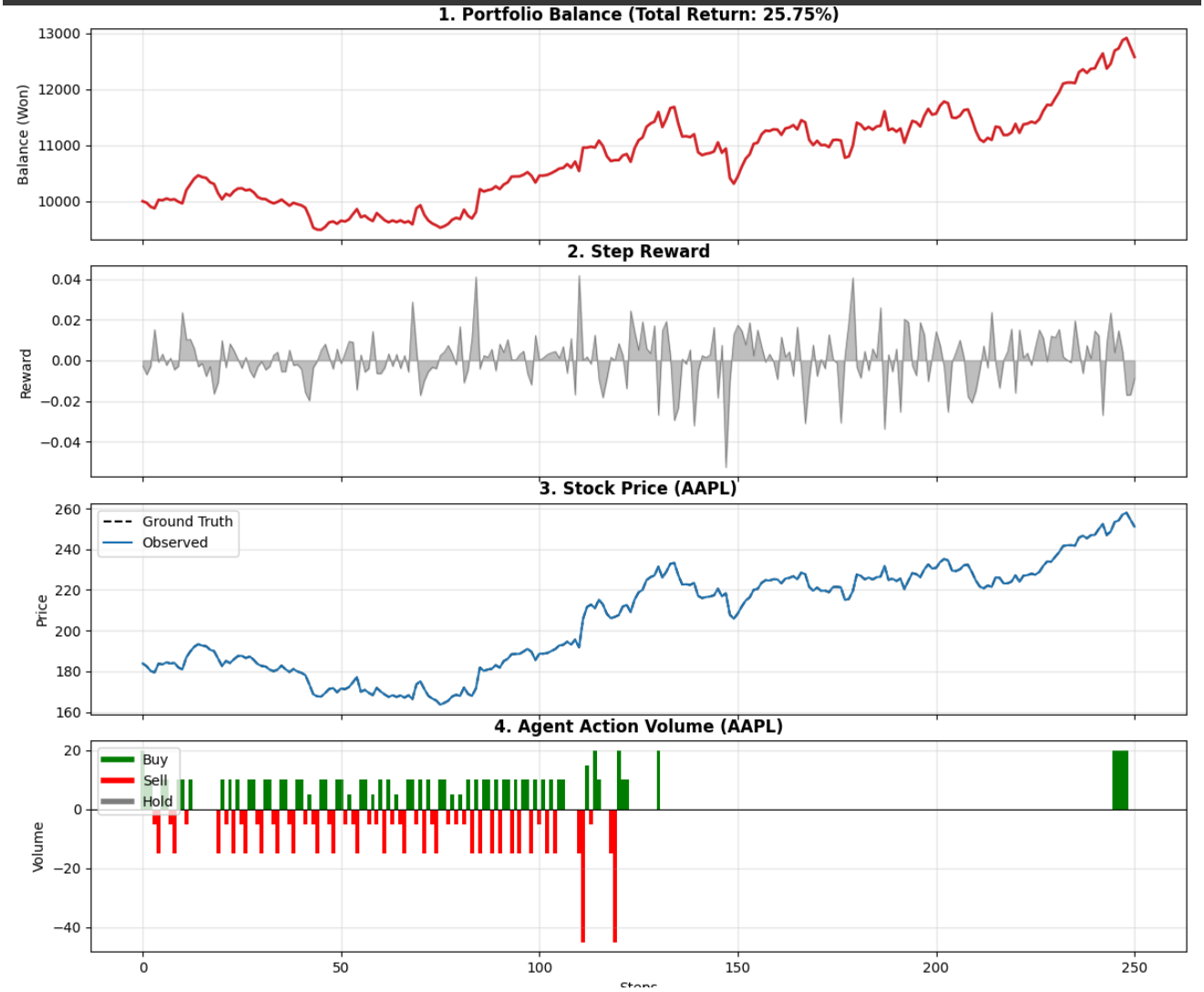


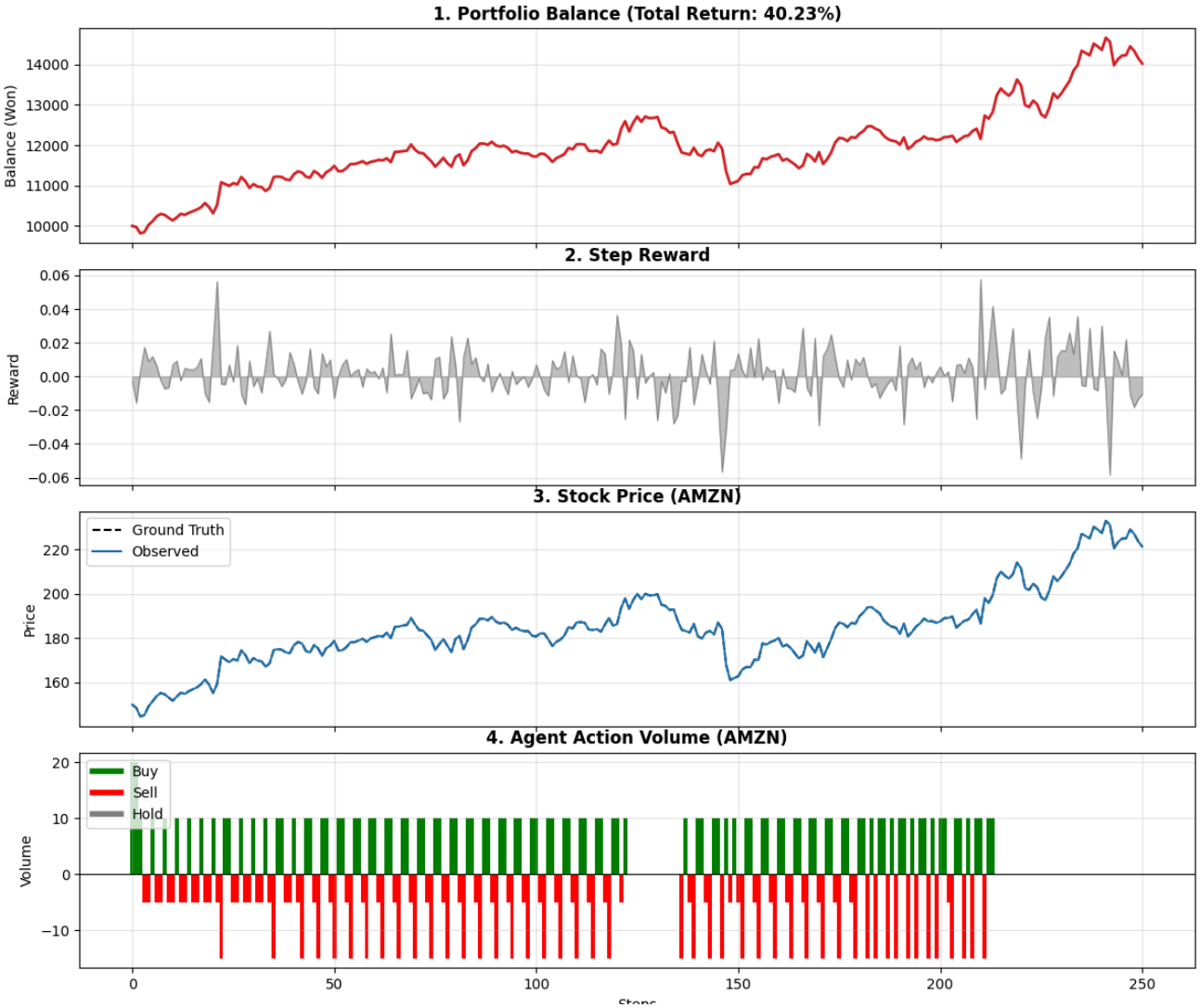

1. Portfolio Balance (Total Return: 47.34%)
Balance (Won)
2. Step Reward
Reward
3. Stock Price (AMZN)
Ground Truth
Observed
Price
4. Agent Action Volume (AMZN)
Buy
Sell
Volume
Steps

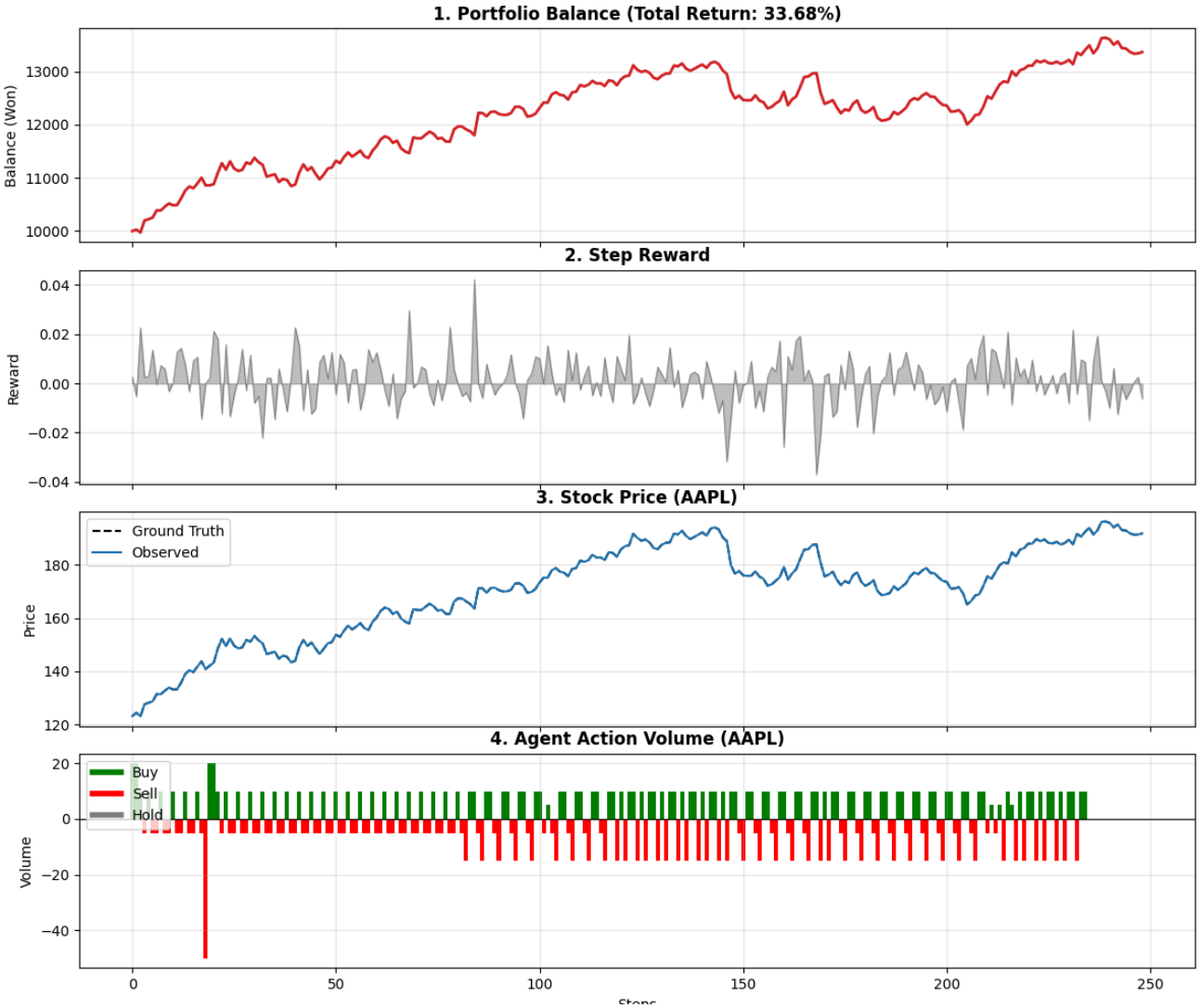

1. Portfolio Balance (Total Return: 54.46%)
Balance (Won)
2. Step Reward
Reward
3. Stock Price (AAPL)
Ground Truth
Observed
Price
4. Agent Action Volume (AAPL)
Buy
Sell
Volume
Steps

DQN Results


charts/episodic_return
Alt + Scroll to Zoom
3
2
1
0
0
500k
1M
1.5M
2M
2.5M
Run
Value
Step
Time
Relative
20251218-173548_DQN_1
0.7013
394,607
12/18/25, 5:50 PM
14.48 min
Value
1.8137
Step
2,723,093
Relative
1.872 hr


1. Portfolio Balance (Total Return: 25.75%)
Balance (Won)
13000
12000
11000
10000
2. Step Reward
Reward
0.04
0.02
0.00
−0.02
−0.04
3. Stock Price (AAPL)
Price
Ground Truth
Observed
260
240
220
200
180
160
4. Agent Action Volume (AAPL)
Volume
Buy
Sell
Hold
20
0
−20
−40
0
50
100
150
200
250


1. Portfolio Balance (Total Return: 40.23%)
Balance (Won)
14000
13000
12000
11000
10000
2. Step Reward
Reward
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
3. Stock Price (AMZN)
Ground Truth
Observed
Price
220
200
180
160
4. Agent Action Volume (AMZN)
Buy
Sell
Hold
Volume
20
10
0
−10
0
50
100
150
200
250



Discussion

- To compare the learning graph tendency, as you can see in the PPO learning graph, it looks like it has similarly high variance from DQN's but the lower bound is slightly getting higher, as the learing proceeds. As PPO is using clipping loss, some radical updates can be prevent, so the varince plot of PPO can be seen as slightly more stable than DQN. However, for DQN's learning graph, it shows high picks even in the later phase of the learning.
- As we are using very limited data(at most around 1000days), the performance of PPO and DQN actions are interesting. PPO just use the data once and then get rid of it. (Still there is a loop for reusing data multiple times; in my code, it's 4times) However, DQN keeps all the data and reusing thoughout the learning process. In addition to that, PPO's action space is continuous, while DQN's actions space is discretized. Here, it's 21 options. Generally, using continuous actions will be more helpful more sensetive investment, but given that there is very little data, it might not have enough data to explore. The action space might be to large.
- So the actions of PPO wasn't able to learning differnet actions for different observation states, but just take action of full buying or full selling(exploitatoin). However, DQN has small action space, with efficient data usage, which makes DQN to explore the aciton space enough, so it can output more diverse actions based on the observation state.

Future Works

- As I mentioned before, the main problem of all the trained policies is the dataset. It is too small and biased. It doesn't have enough data to make the policy explore enough. So I will try to get more unbiased datasets.
- Apart from dataset, I would like to take 2 differnet approach to improve the performance.
- First, company value. Specify all the informations related to the value of the company. By gathering those information and preprocessing and then using as a state, would help the agent to more accurately predict the value of the company.
- Second, tendency of the market. Not directly predicting the value of stock itself, I would like to take an approach interpreting the people's tendency. It may be possible by using analyst's informations as a state.