Report on PPO agent for Finance

November 4, 2025

Mathys VINATIER - GitHub Project page
Supervisor:
Pr Kim Tae-Wan

Mathys VINATIER

Contents

Chapter 21

Week 21

Contents

21.1 The PPO Agent Training

This week, we built and trained the model based on the previous recommendations. Several experiments were conducted to fine-tune and identify the optimal configuration of the PPO agent. Different parameter settings were tested across multiple trials, which will be discussed in this report.

As an initial step, we performed small adjustments to intentionally create an overfitting model. This helps verify that the agent can effectively learn from historical data before introducing regularization or exploration strategies. For this purpose, some parameters were fixed as follows:

- Discount factor $\gamma = 0.99$
- Learning rate $\alpha = 3 \times 10^{-4}$
- Generalized Advantage Estimation parameter $\lambda = 0.95$
- Policy clipping coefficient $\epsilon = 0.2$

These parameters play a key role in shaping the agent's learning dynamics. The discount factor γ determines how strongly future rewards influence current decisions—values close to 1 encourage long-term planning, while smaller values emphasize immediate rewards. The learning rate α controls the step size of parameter updates ; higher values speed up learning but risk instability, whereas lower values make learning slower and more stable. The GAE parameter λ balances bias and variance in advantage estimation : larger λ values produce smoother, more consistent updates but can increase variance. Finally, the clipping coefficient ϵ stabilizes policy updates by preventing the new policy from deviating too far from the old one, ensuring more reliable convergence during training.

21.1.1 First training analysis

For the initial training phase, we selected a set of baseline parameters to establish a reference point for the PPO agent's learning behavior. The chosen configuration is summarized below :

- Episodes : 300
- Epochs per update : 10
- Batch size : 128

The goal of this setup was to allow the agent to experience a relatively large number of episodes, providing diverse trajectories for training, while keeping the number of epochs moderate to prevent overfitting on limited data. This configuration encourages the critic network to refine its value estimation across a broader range of states, which in turn supports more stable policy updates by the actor.

The following section presents and discusses the main results obtained from this first round of training :

Figure 21.1: Training 1 - Training of trial 1

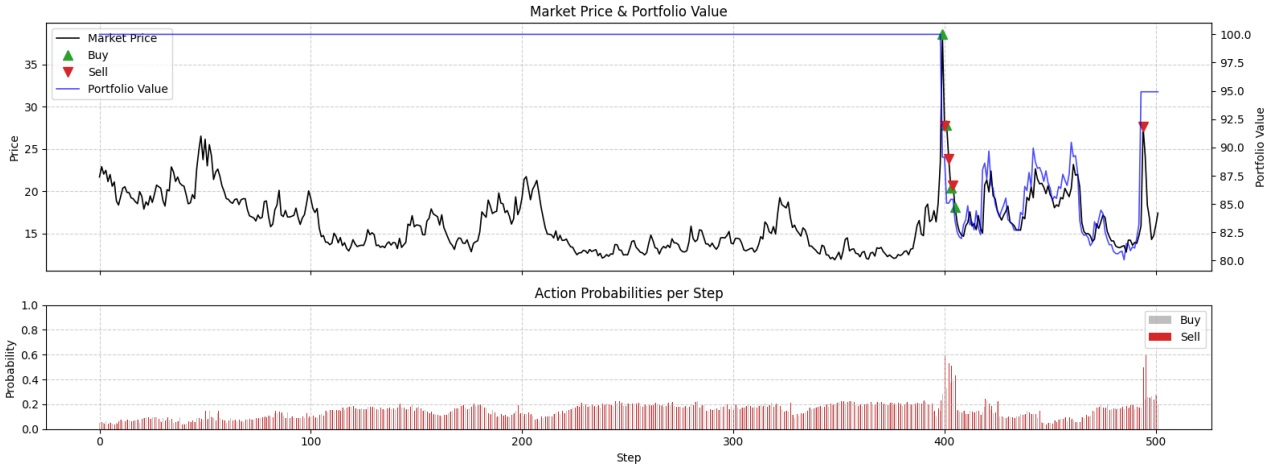


Figure 21.2: Training 1 - Test of trial 1

We can see that the training did not learn well since we have only few trades made, also the test has made very poor actions. We can consider that this first trial is really underfiting.

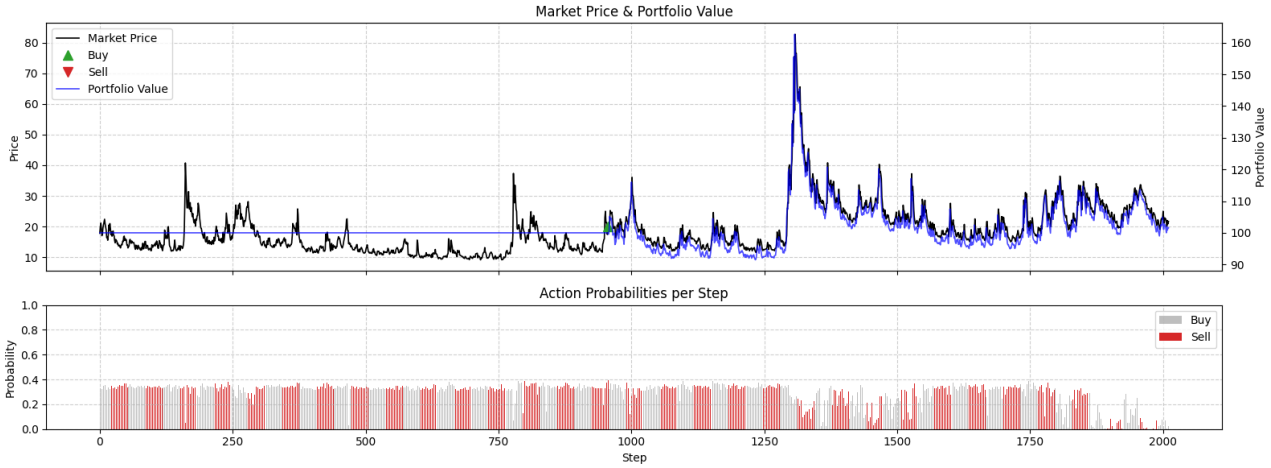


Figure 21.3: Training 1 - Training of trial 2

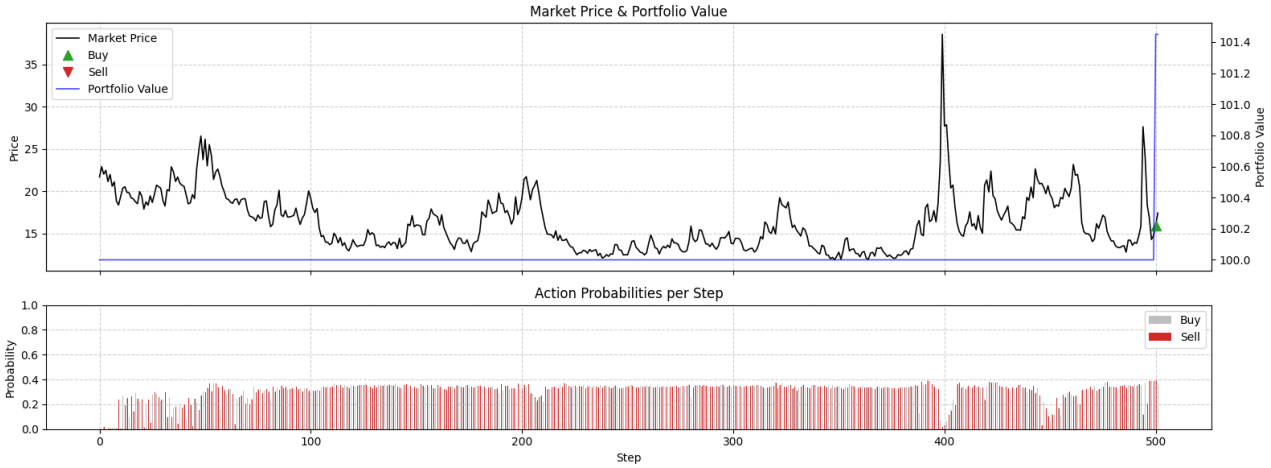


Figure 21.4: Training 1 - Test of trial 2

As previously, we can see that the model is clearly underfiting since it has made only one trade during testing and training which is representing an underfitting model.

Figure 21.5: Training 1 - Training of trial 3

Figure 21.6: Training 1 - Test of trial 3

The model did not take any trade neither on training or testing. We cannot evaluate this model.

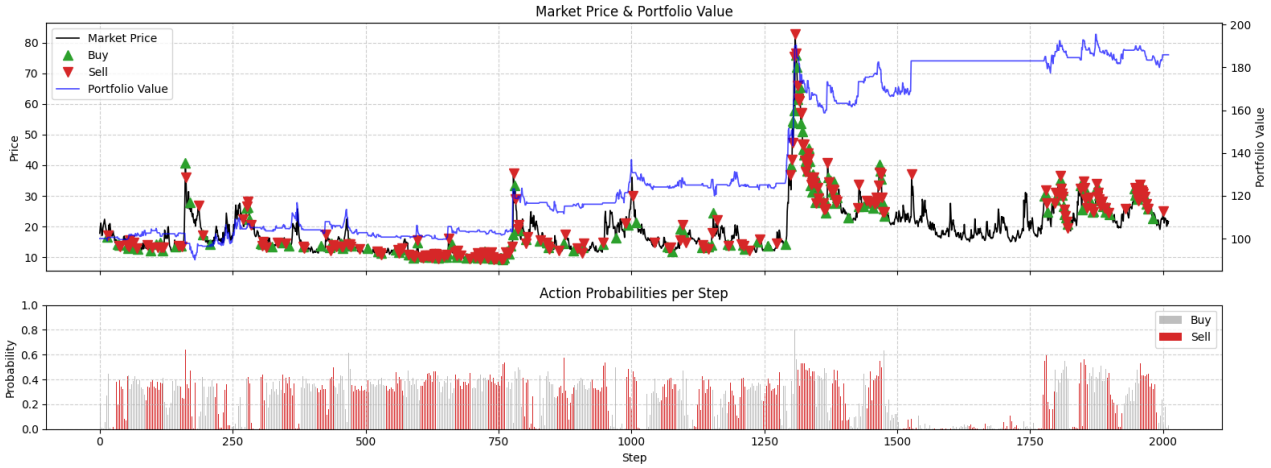


Figure 21.7: Training 1 - Training of trial 4

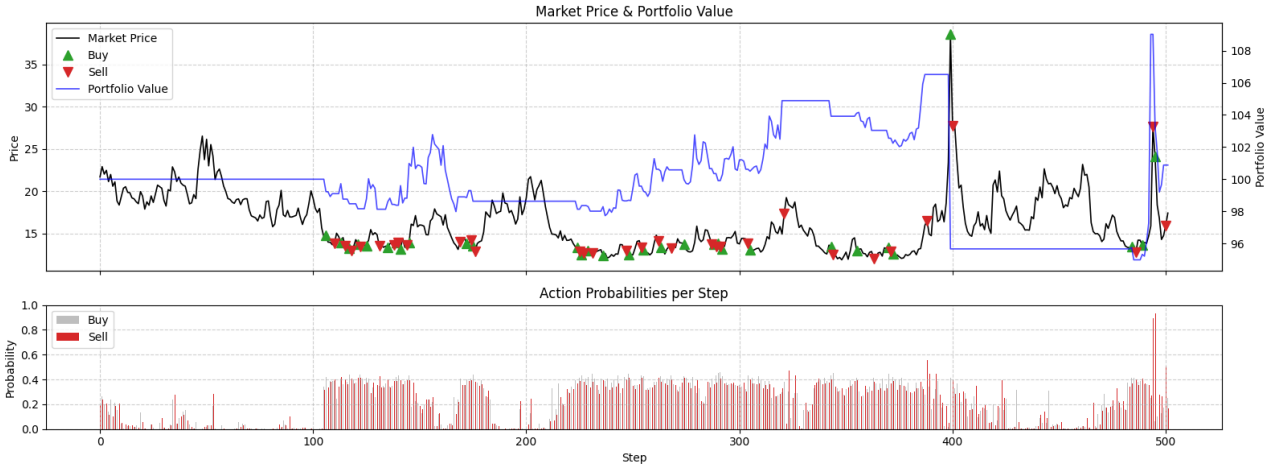


Figure 21.8: Training 1 - Test of trial 4

This model clearly learnt in training. However in testing the model is overfitting since it is not making wise trade.

Figure 21.9: Training 1 - Training of trial 5

Figure 21.10: Training 1 - Test of trial 5

Like the trial 3, we cannot analyse this model since it has made no trade.

Trial	Training Profit	Testing Profit	Annual Return	Annual Volatility	Sharpe Ratio	Max Drawdown
1	−822.5	74	0.17$/year	0.17	−0.07	−0.20
2	4.5	3	0.01$/year	0.01	0.71	0.00
3	0	0	0.00$/year	0.00	0	0.00
4	3930.5	0	0.13$/year	0.13	0.10	−0.11
5	0	0	0.00$/year	0.00	0	0.00

Figure 21.11: Summary of the trials performances

The results from these trials indicate clear signs of underfitting. Most configurations exhibited weak overall performance, and the single trial that achieved relatively good results during training failed to generalize well in the testing phase, suggesting that the model did not effectively capture the underlying dynamics of the environment.

21.1.2 Second training analysis

We would like to get a less overfitting model, for that we will work with more epoch and less episode to get a better trained thru the episodes, we will use the following training parameters :

- Episodes : 200
- Epochs per update : 50
- Batch size : 128

For this training, we are going to get a better look at each episode to better analyse the learning behavior of our model :

Figure 21.12: Training 2 - Training thru the epochs

As observed, the model demonstrates progressively wiser trading behavior across episodes, suggesting that it successfully captures and adapts to the underlying market dynamics during training. The learning curve reveals two distinct phases: initially, the agent learns to execute more strategic and selective trades, improving the efficiency of its decisions; subsequently, the model stabilizes the number of trades while gradually increasing overall profit.

Compared to the previous configuration, this version exhibits stronger learning performance, primarily due to the increased number of training epochs. However, the risk of overfitting remains and further evaluation on unseen data is necessary to assess the model's generalization capability and ensure robust performance in different market conditions :

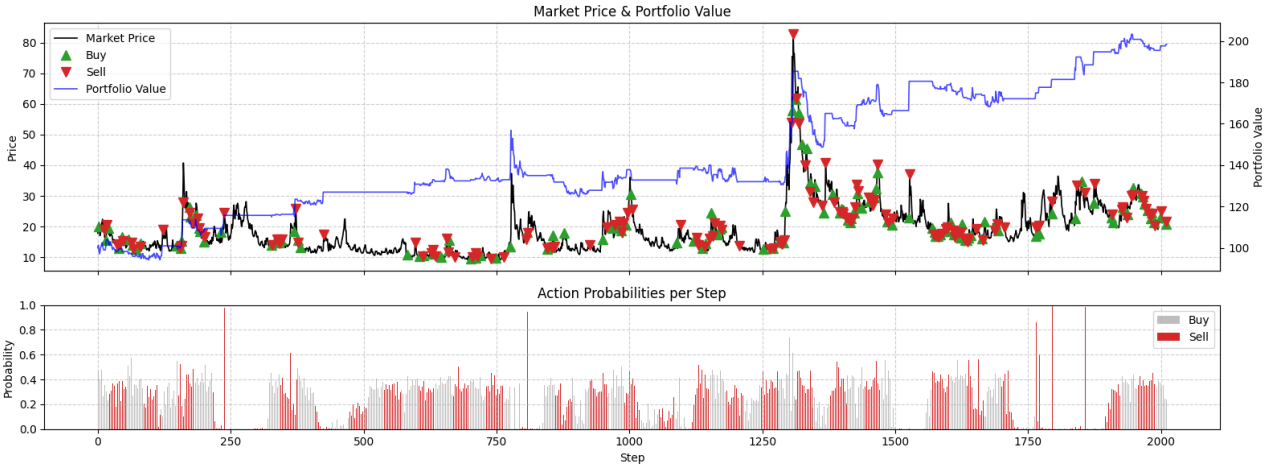


Figure 21.13: Training 2 - Training of trial 1

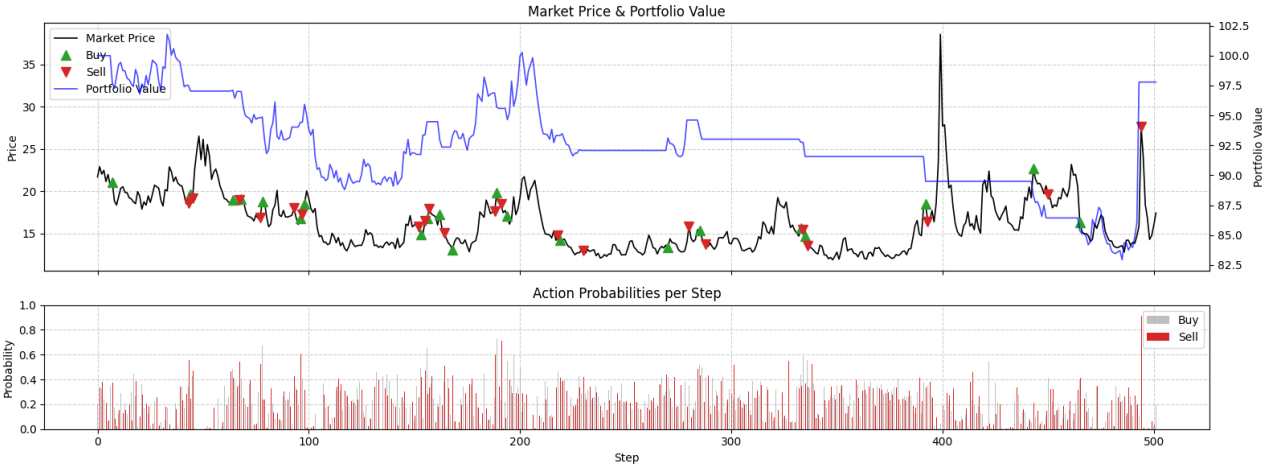


Figure 21.14: Training 2 - Test of trial 1

As we can see, the model has been overfitting. However since it is a stochastic model, we should compute many time our model to determine his usual behavior. Adding other features has input from other related market could be also a solution to our problem (VVIX / S&P500).

Chapter 23

Week 23

Contents

23.1 The PPO Training API

In this section, we present the key components of the API we developed to automate the instantiation of PPO training for different agents. This API is crucial for fine-tuning agents and understanding the structure of the reward function.

The API consists of both frontend and backend components. The frontend, built using HTML and CSS, provides a dynamic dashboard to visualize the training process in real time. The `logs` directory contains the raw output of the training scripts executed via the command line, which is essential for debugging and monitoring the progress of each training episode.

The `static` and `templates` directories support the frontend interface, managing visual elements and the real-time display of training metrics. These components are critical for handling errors gracefully without interrupting training or saving models, which are the core pieces of information.

Finally, the `utils` directory contains helper functions that complement `main.py`, the central script responsible for running the FastAPI application with Uvicorn. This setup enables a dynamic connection between the frontend and the backend databases, where all training data and model information are stored securely and efficiently.

```
/PPO_training_monitor/
├── logs/
│   ├── training_task_0.log
│   ├── training_task_1.log
│   └── ...
├── static/
│   ├── script.js
│   └── style.css
├── templates/
│   └── dashboard.html
├── utils/
│   ├── __init__.py
│   ├── config.py
│   ├── getter.py
│   ├── info_system.py
│   ├── listing.py
│   ├── plot.py
│   └── task_manager.py
└── main.py
```

Figure 23.1: API Directory Architecture

The Dashboard of the front end has 3 main parts :

- **Training Monitor** : Show a graph of the overview of each episode (Loss / number of trade / reward) with a tabular that display the detailed information of each databases of the selected task
- **Testing Monitor** : Display 3 graphics : The analysis report that is giving the results of each saved models on both the training and the testing dataset. Two plots that gievs for the training and testing dataset the detailed of the chosen action of the latest saved model.
- **System Stats** : Provide the computer statistics, the current active tasks, the launcher of training and a logs that is giving live information about the selected task.

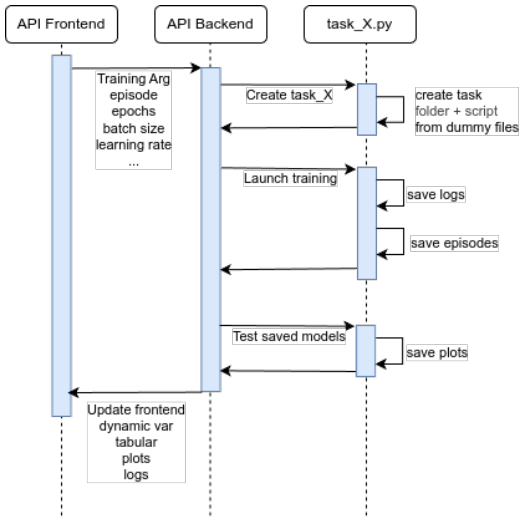
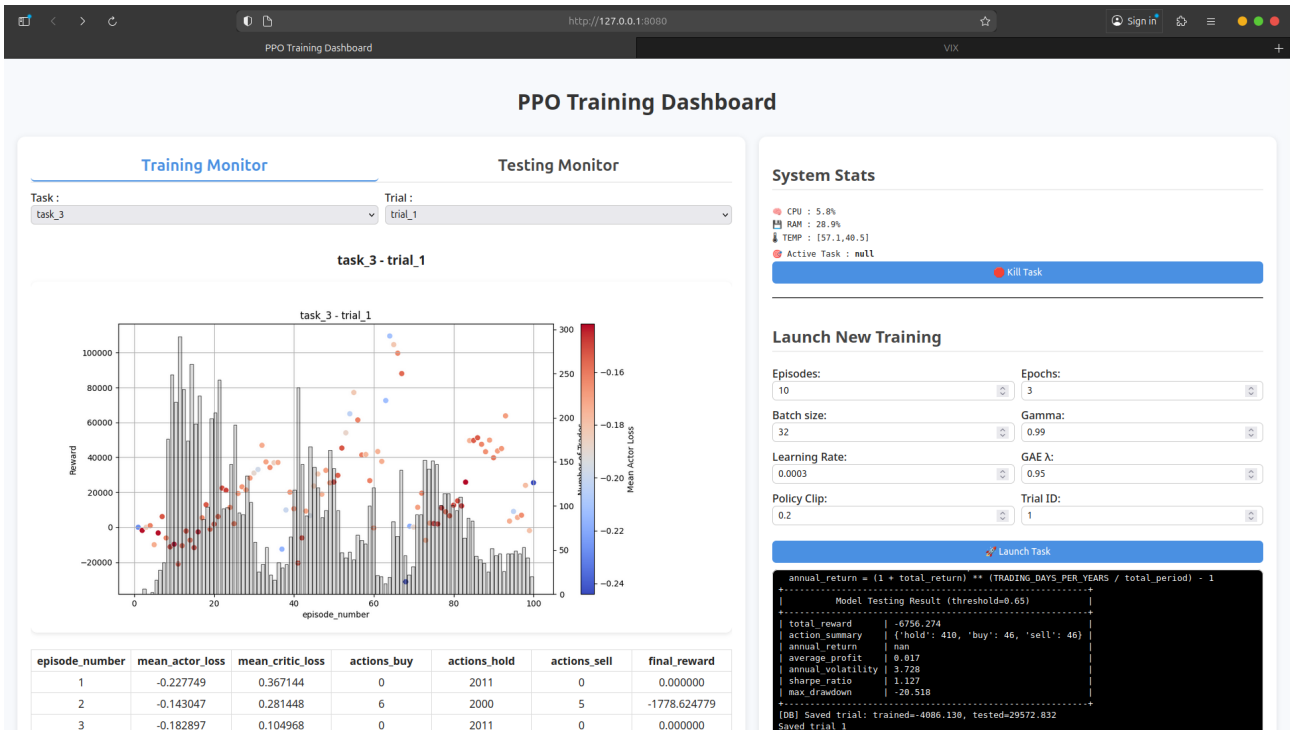


Figure 23.2: API - Sequence Diagram



episode_number	mean_actor_loss	mean_critic_loss	actions_buy	actions_hold	actions_sell	final_reward
1	-0.227749	0.367144	0	2011	0	0.000000
2	-0.143047	0.281448	6	2000	5	-1778.624779
3	-0.182897	0.104968	0	2011	0	0.000000

Figure 23.3: Dashboard - Training Monitor

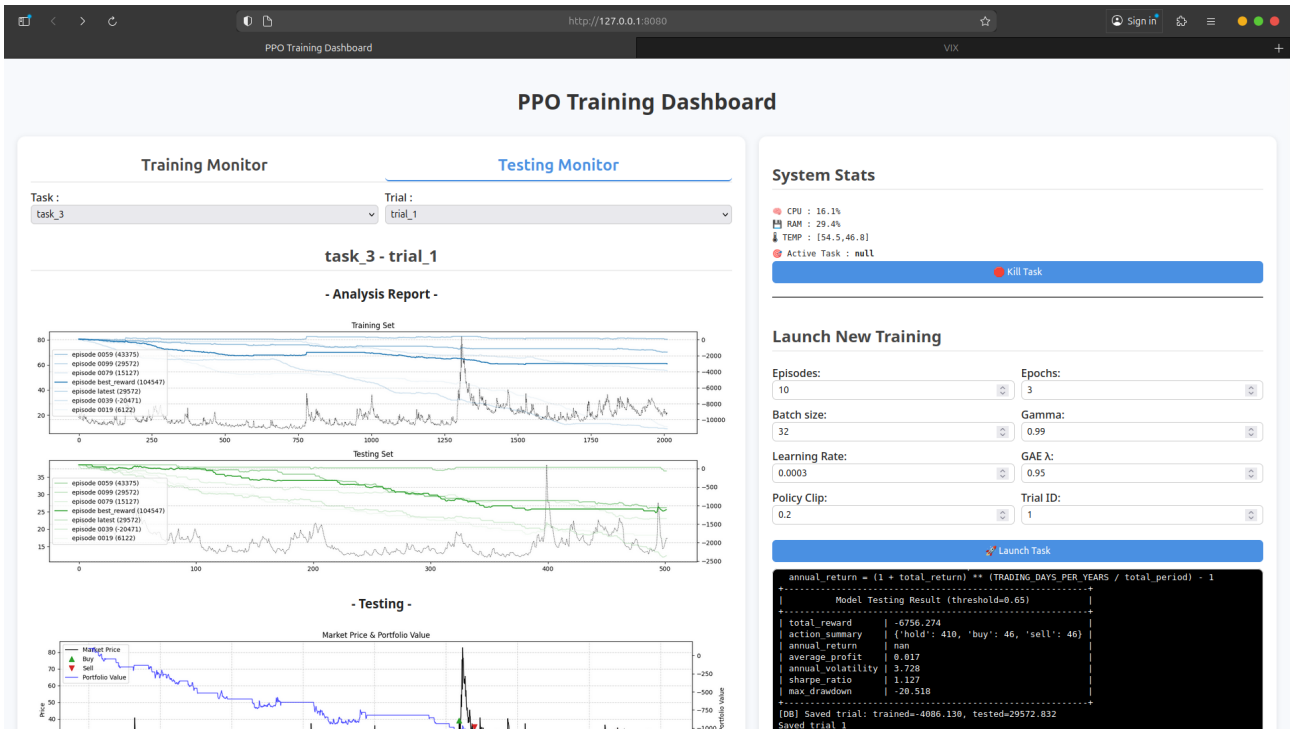


Figure 23.4: Dashboard - Testing Monitor

23.2 Training without borker fees

23.2.1 Task 0 (no broker) - batch_size = 128, epochs = 50, episode = 200

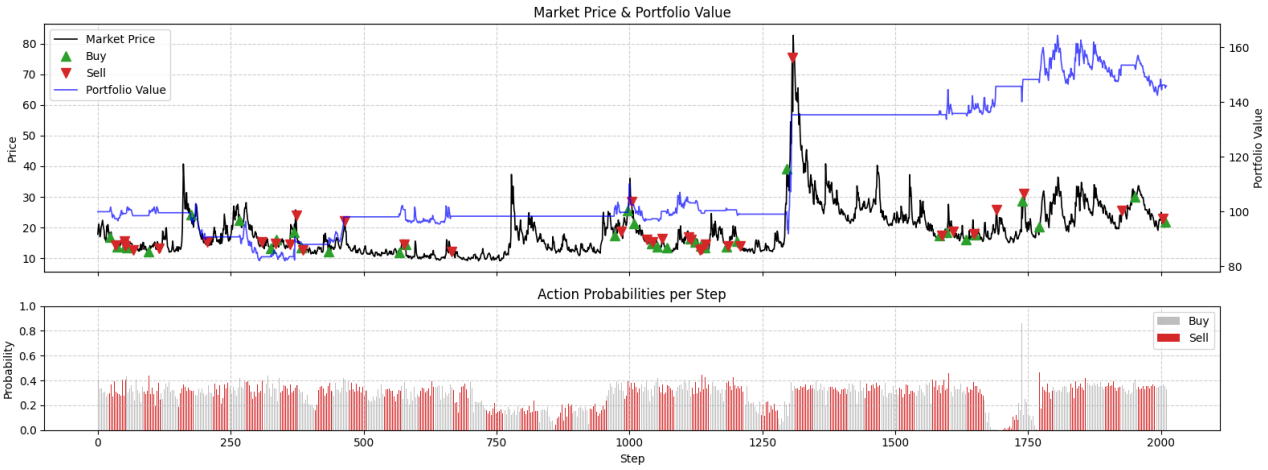


Figure 23.5: Task 0 - Training plot (no broker)

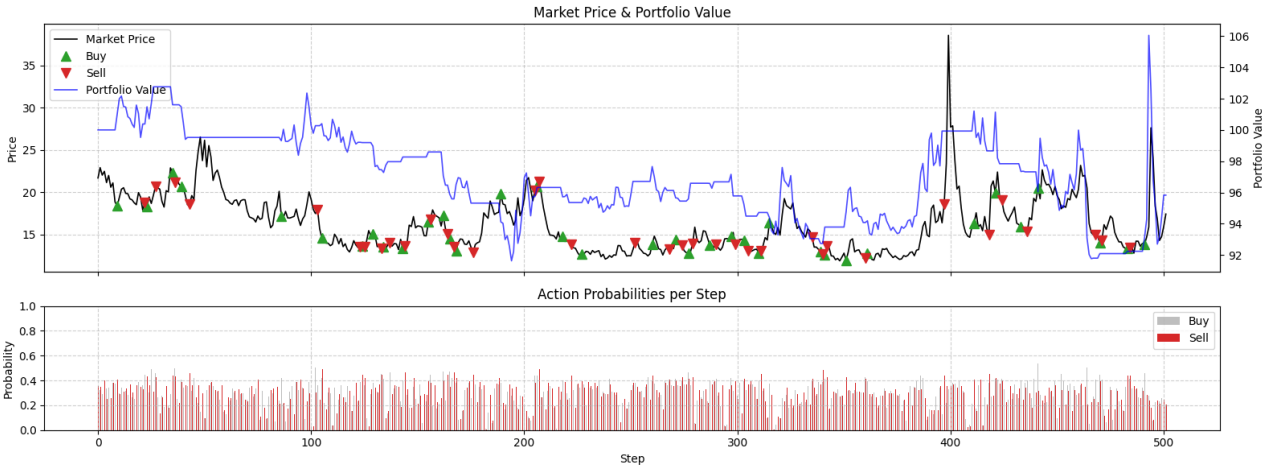


Figure 23.6: Task 0 - Testing plot (no broker)

23.2.2 Task 1 (no broker) - batch_size = 128, epochs = 10, episode = 100

Figure 23.7: Task 1 - Training plot (no broker)

Figure 23.8: Task 1 - Testing plot (no broker)

23.2.3 Task 2 (no broker) - batch_size = 128, epochs = 100, episode = 100

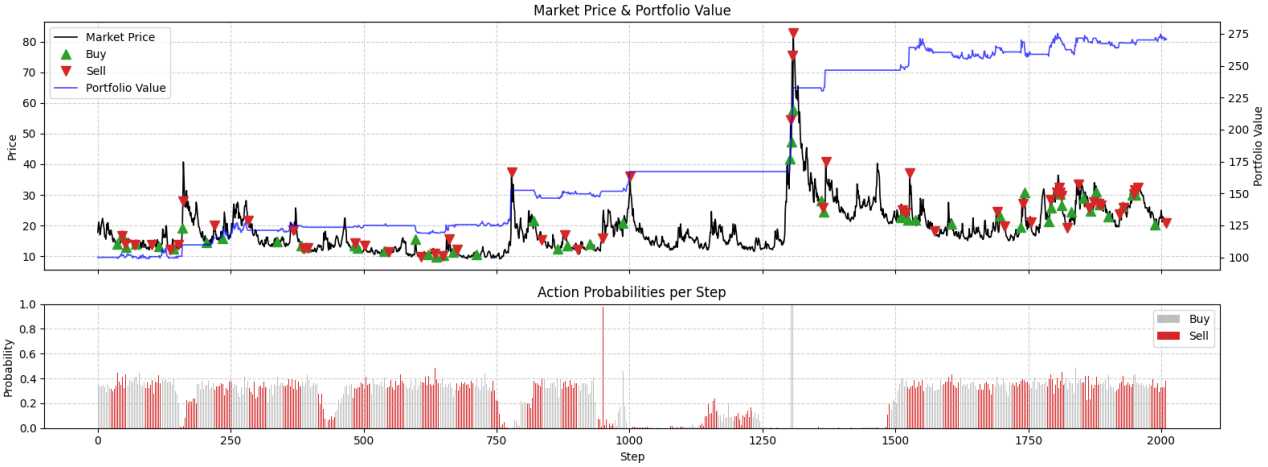


Figure 23.9: Task 2 - Training plot (no broker)

Figure 23.10: Task 2 - Testing plot (no broker)

23.2.4 Task 3 (no broker) - batch_size = 64, epochs = 112, episode = 10

Figure 23.11: Task 3 - Training plot (no broker)

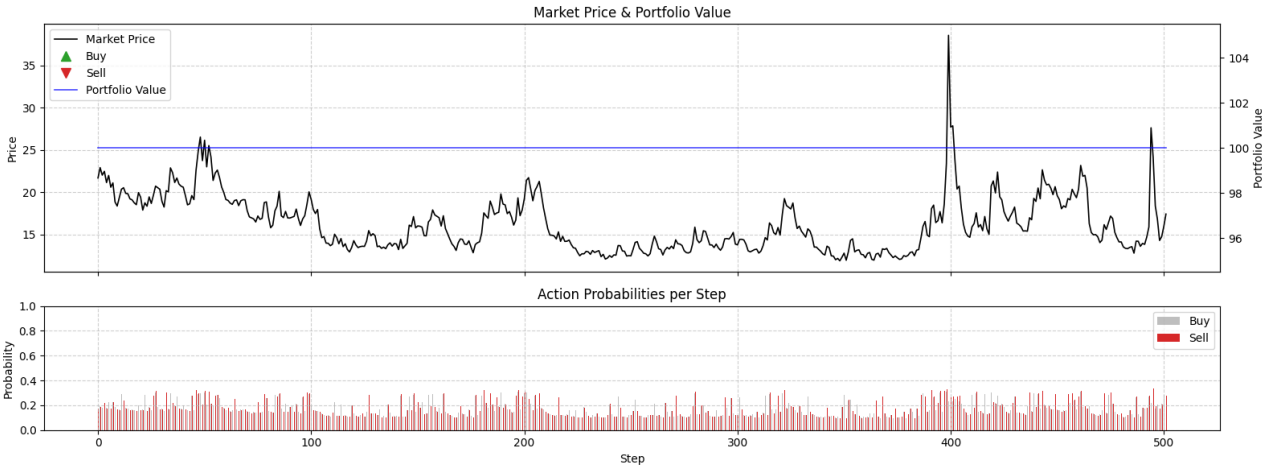


Figure 23.12: Task 3 - Testing plot (no broker)

23.2.5 Task 4 (no broker) - batch_size = 32, epochs = 100, episode = 100

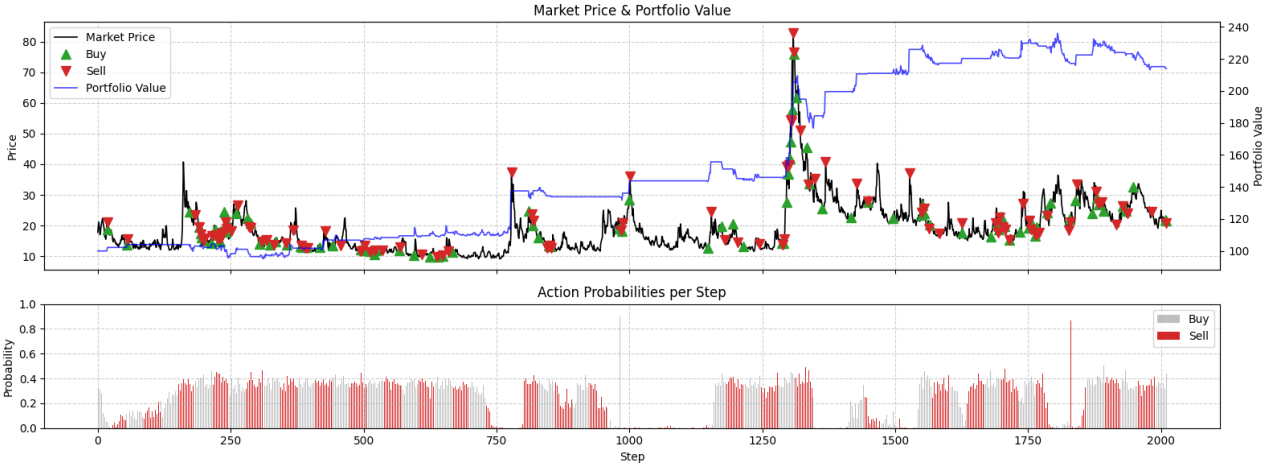


Figure 23.13: Task 4 - Training plot (no broker)

Figure 23.14: Task 4 - Testing plot (no broker)

23.2.6 Task 5 (no broker) - batch_size = 64, epochs = 100, episode = 100

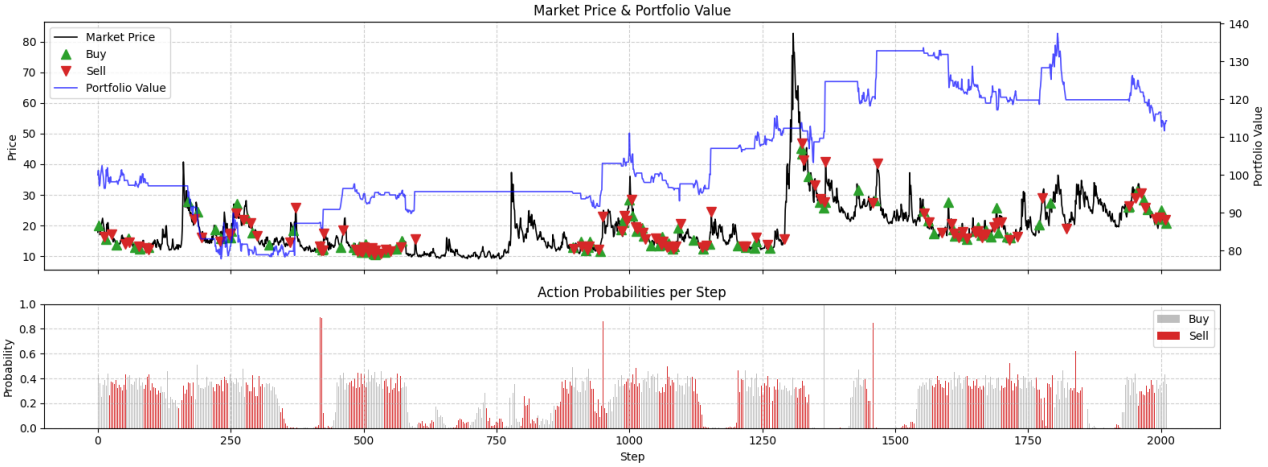


Figure 23.15: Task 5 - Training plot (no broker)

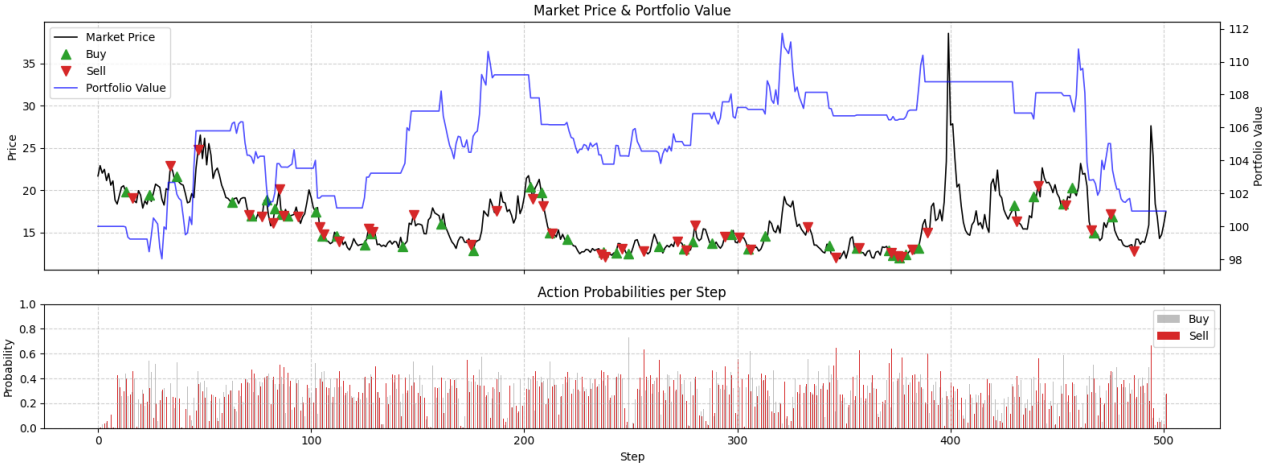


Figure 23.16: Task 5 - Testing plot (no broker)

23.2.7 Conslusion on tasks (no broker)

We can see than half of the models did not took a trade, it seems that with few episode or epochs (~ 10), the model is making to trades, which make sense since it has not learn enough. Overall, the models seems to understand the dynamic of the market during trading and the best model on test has been for the task 4 :

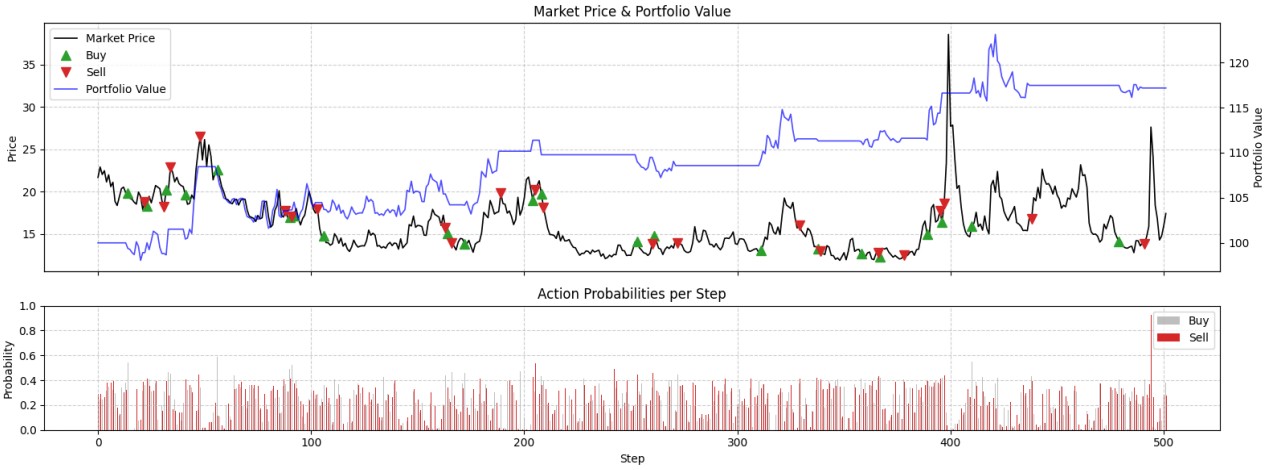


Figure 23.17: recap of Task 4 - Testing plot (no broker)

annual return	avg profit	annual vol.	max drawdown
0.082	0.0003	0.11	6%

Figure 23.18: Report Analysis of task 4

This model has a good drawdown and seems reliable and able to understand the market dynamic, so far the best model has been :

Episodes	Epochs	Tick	Batch Size	Learning Rate	Final Profit on test
100	100	Daily	32	0.0003	17%

Figure 23.19: Detailed of the best trial

Let's remember that there is no borker's fee during these tests. The next part is dedicated to the broker's fee experiment. That way we can see if our model and training are adapted to the real world market.

23.3 Training with borker fees

23.3.1 Task 0 - batch_size = 32, epochs = 50

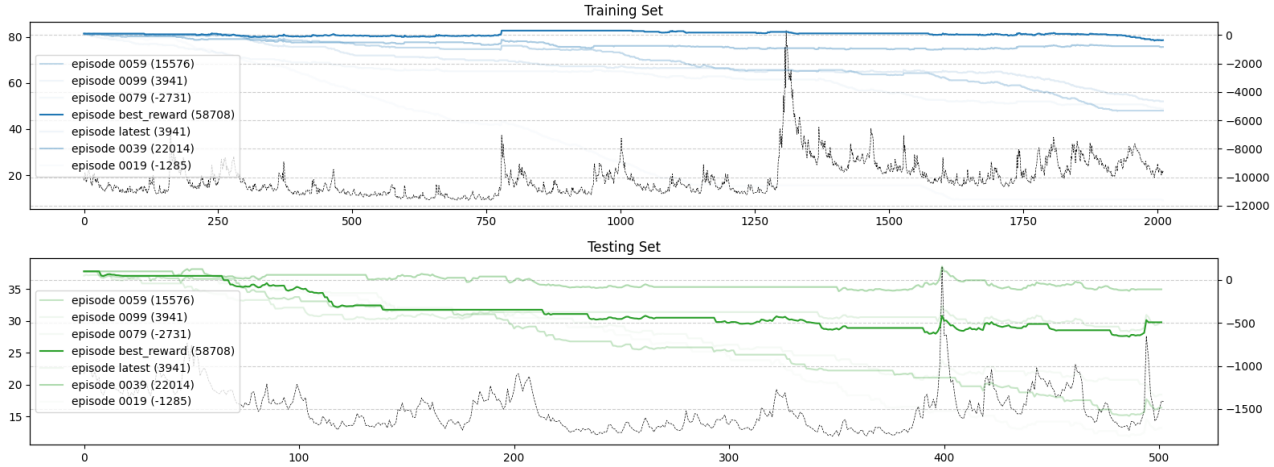


Figure 23.20: Task 0 - Analysis plot

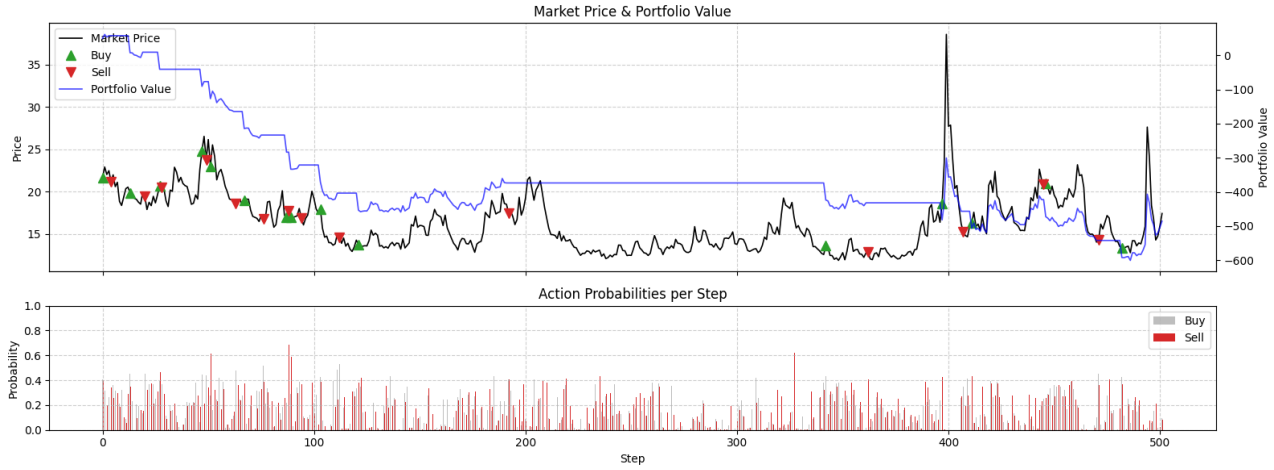


Figure 23.21: Task 0 - Training plot

Figure 23.22: Task 0 - Testing plot

23.3.2 Task 1 - batch_size = 32, epochs = 100

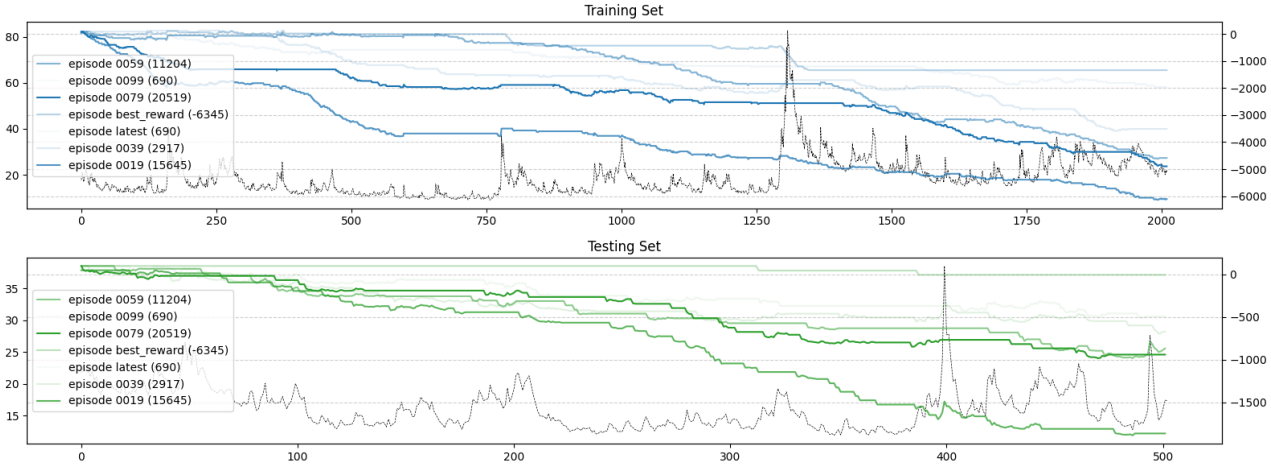


Figure 23.23: Task 1 - Analysis plot

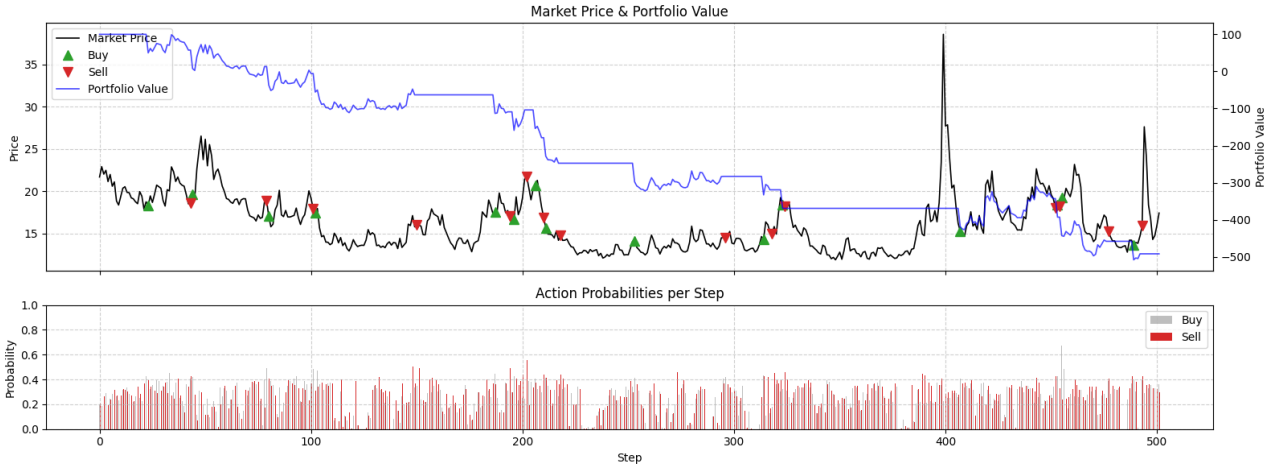


Figure 23.24: Task 1 - Training plot

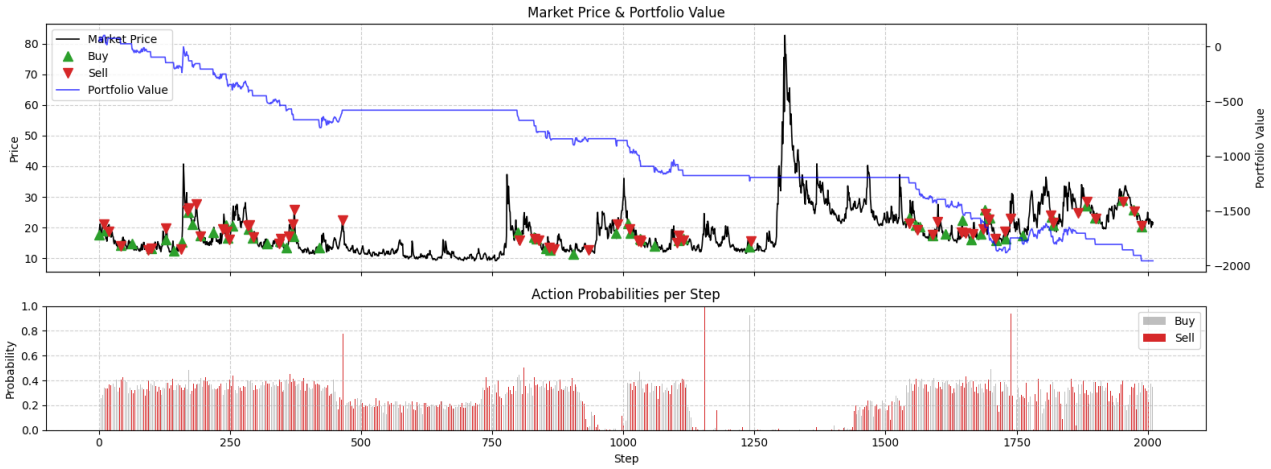


Figure 23.25: Task 1 - Testing plot

23.3.3 Task 2 - batch_size = 64, epochs = 50

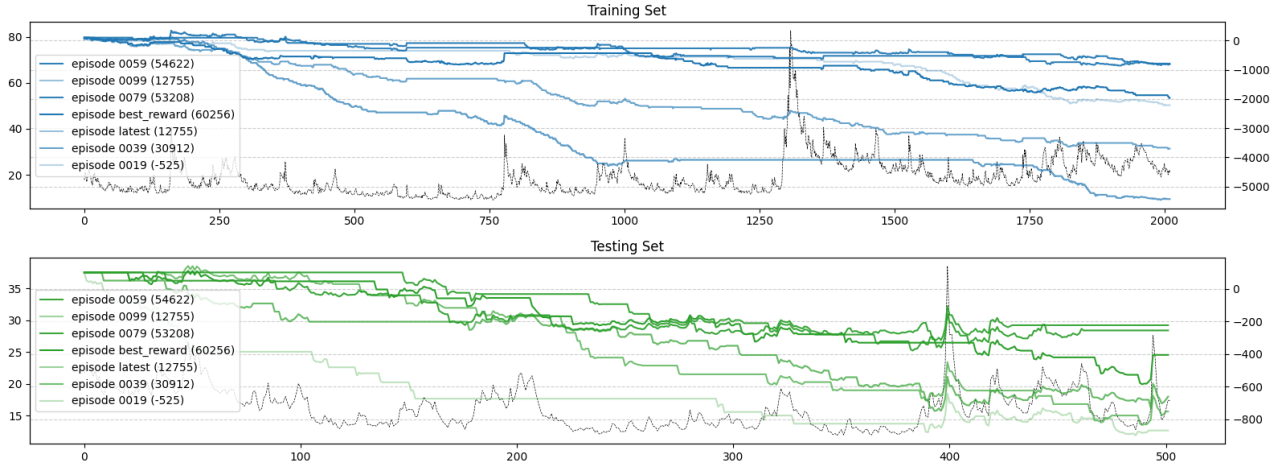


Figure 23.26: Task 2 - Analysis plot

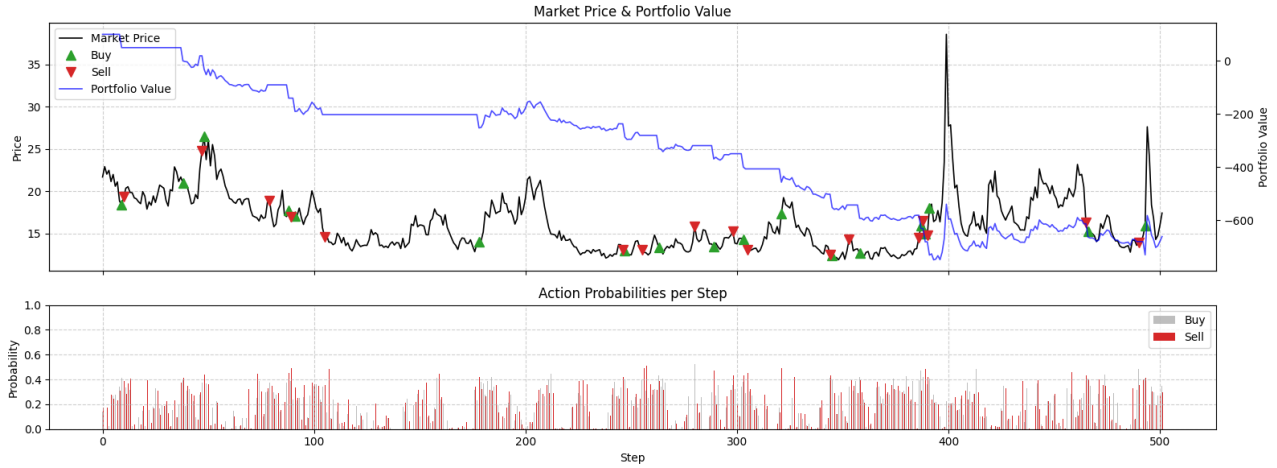


Figure 23.27: Task 2 - Training plot

Figure 23.28: Task 2 - Testing plot

23.3.4 Task 3 - batch_size = 64, epochs = 100

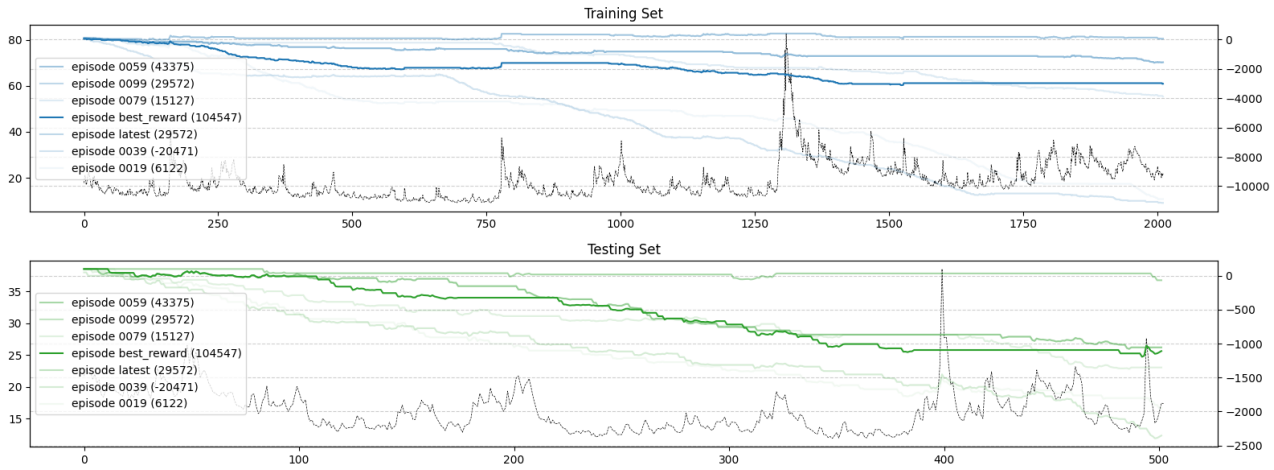


Figure 23.29: Task 3 - Analysis plot

Figure 23.30: Task 3 - Training plot

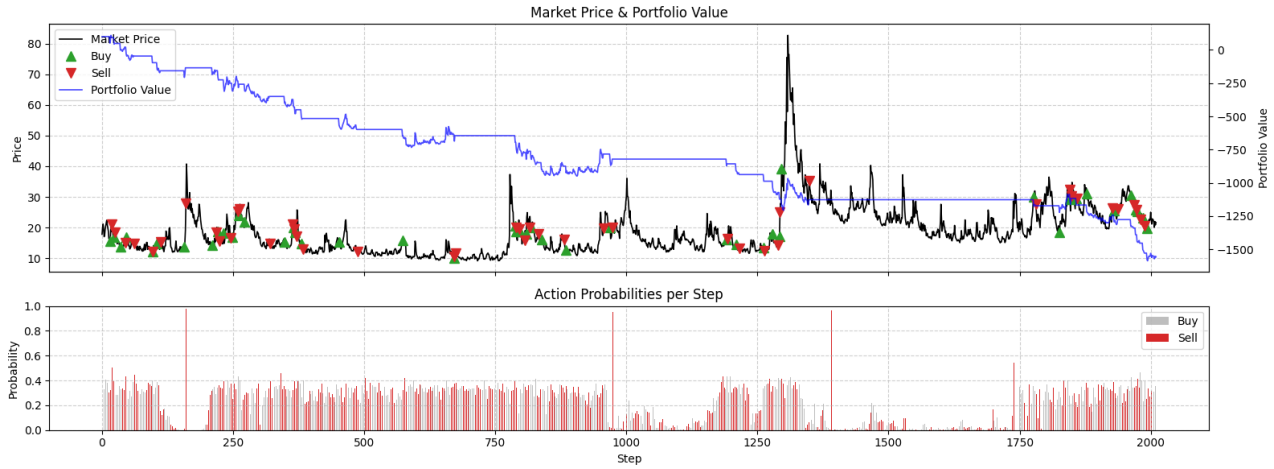


Figure 23.31: Task 3 - Testing plot

23.3.5 Task 4 - batch_size = 128, epochs = 50

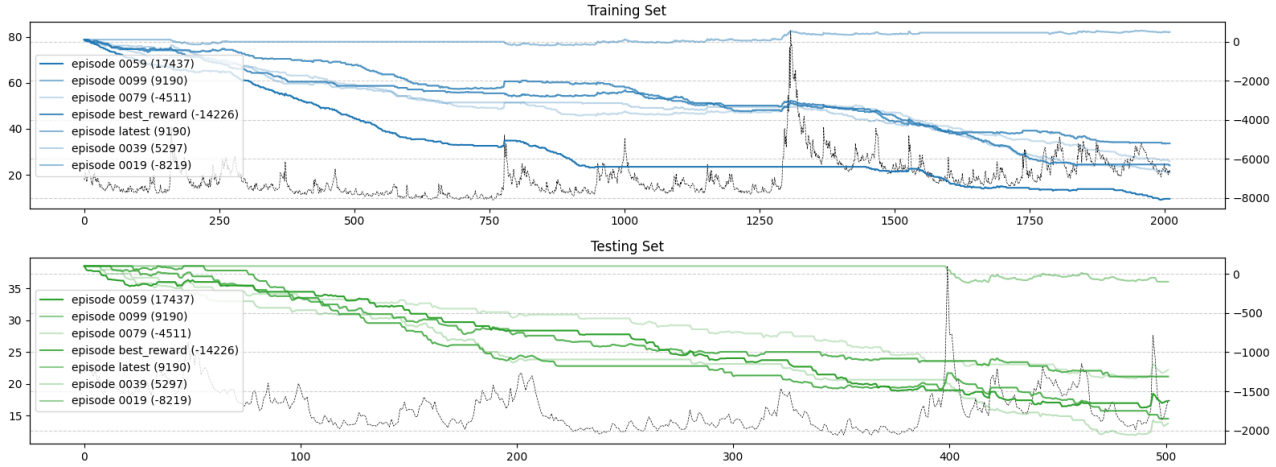


Figure 23.32: Task 4 - Analysis plot

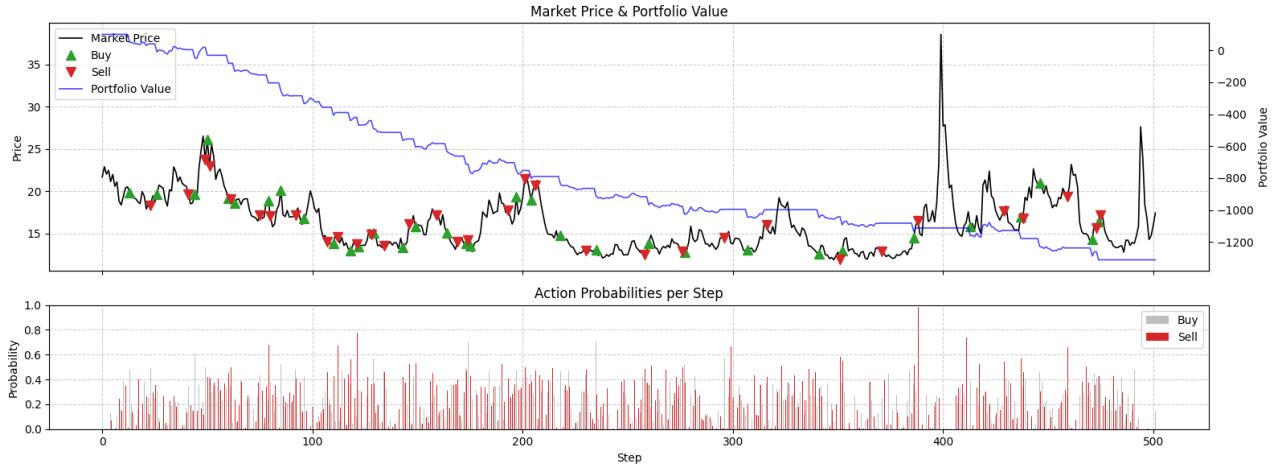


Figure 23.33: Task 4 - Training plot

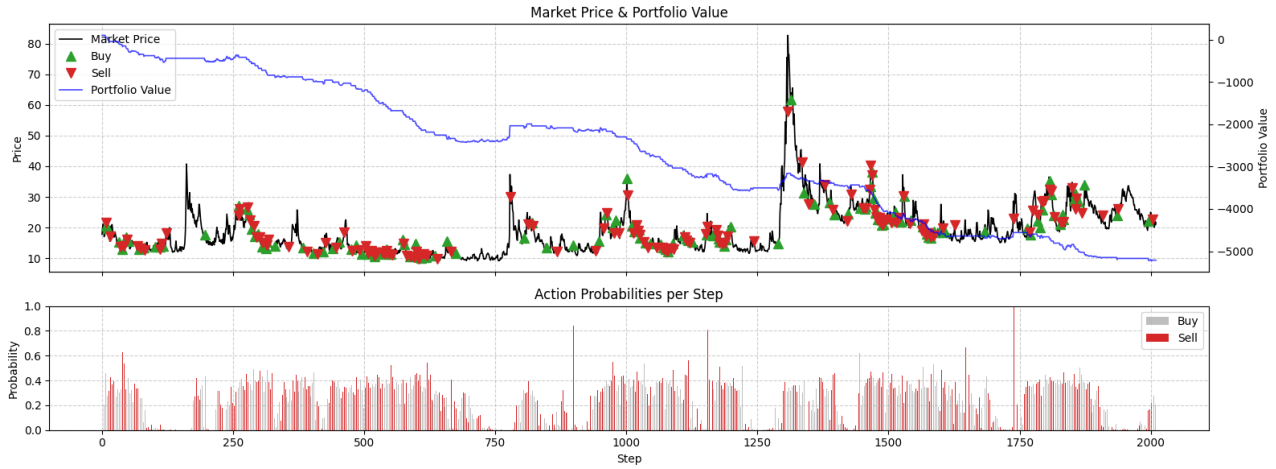


Figure 23.34: Task 4 - Testing plot

23.3.6 Task 5 - batch_size = 128, epochs = 100

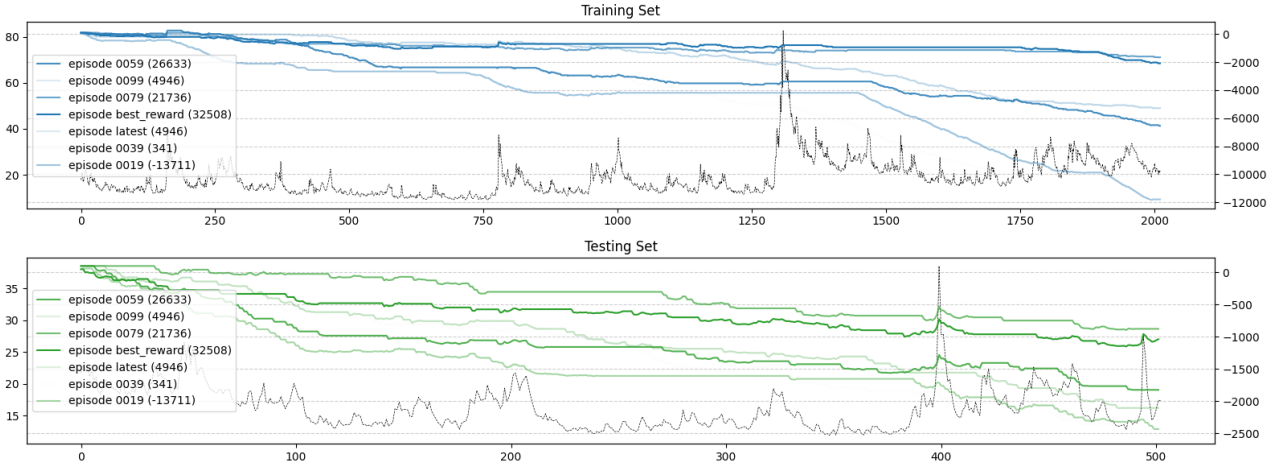


Figure 23.35: Task 5 - Analysis plot

Figure 23.36: Task 5 - Training plot

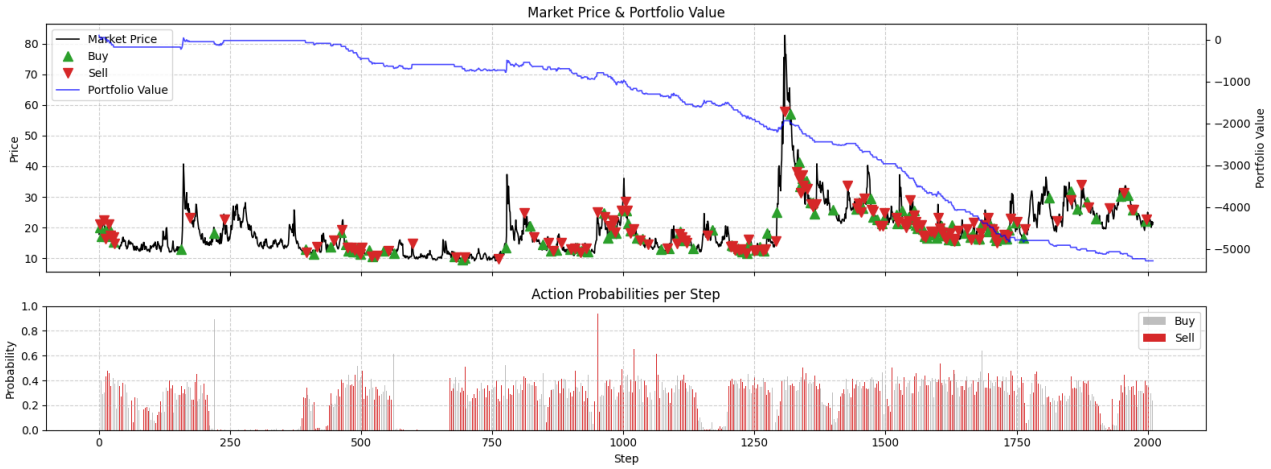


Figure 23.37: Task 5 - Testing plot

23.3.7 Task 6 - batch_size = 256, epochs = 50

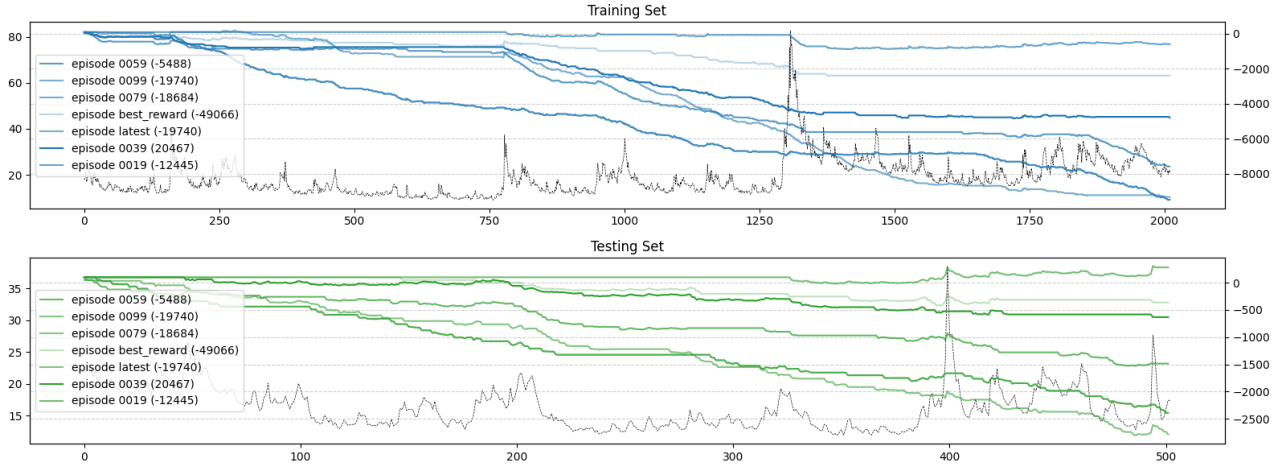


Figure 23.38: Task 6 - Analysis plot

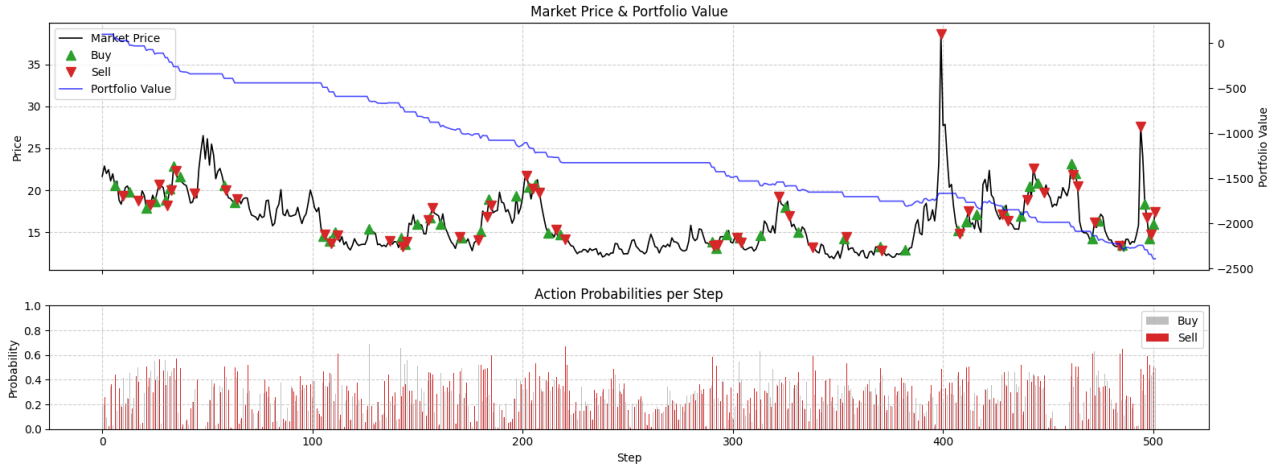


Figure 23.39: Task 6 - Training plot

Figure 23.40: Task 6 - Testing plot

23.3.8 Task 7 - batch_size = 256, epochs = 100

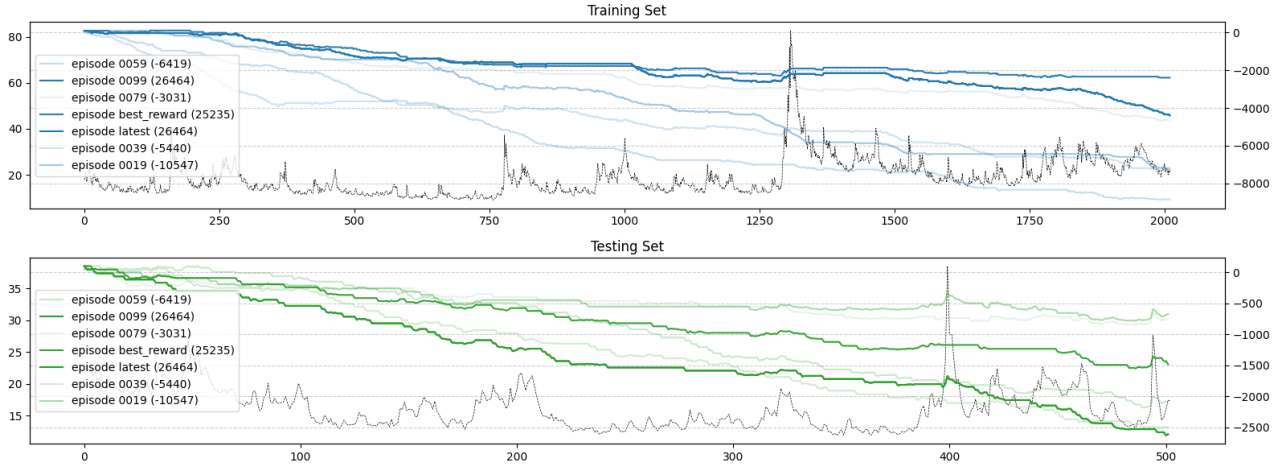


Figure 23.41: Task 7 - Analysis plot

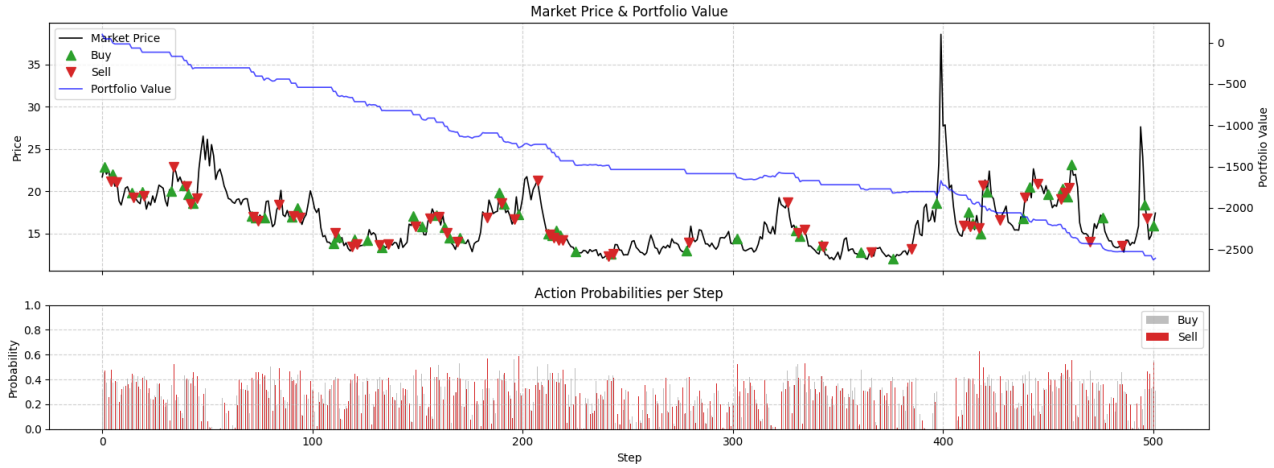


Figure 23.42: Task 7 - Training plot

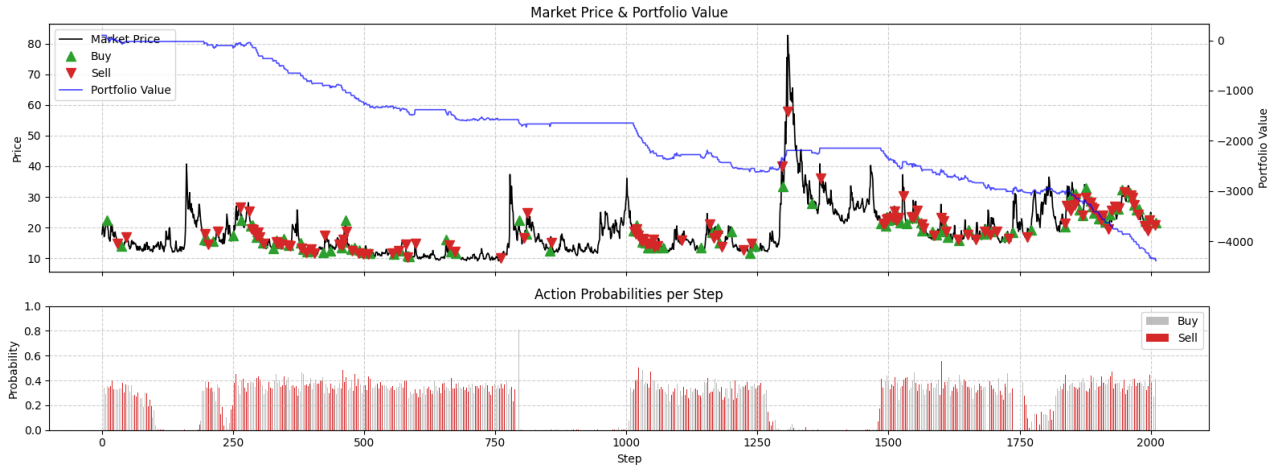


Figure 23.43: Task 7 - Testing plot

23.3.9 Conclusion on tasks

As we saw, the results of this first week of training had been poor. The model has a real issue handling the broker's fees, except for one task. The task number 6 had poor perforance during the training, however, the episode 19 showed a good final results by well handling a peak at step 400 :

Figure 23.44: Task 6 - Episode 19 Train

Figure 23.45: Task 6 - Episode 19 Test

As we can see, the test dataset has good performance but is making only one trade during two years. Also the training is terrible. We think that there is an issue with the reward function since it is always bad, further investigation needs to be done. However the conclusion on that first week training is that the best results has been on these parameters :

Episodes	Epochs	Tick	Batch Size	Learning Rate	Final Profit on test
19	50	Daily	256	0.0003	17%

Figure 23.46: Detailed of the best trial

23.4 Workflow - What's left ?

So far our main core of the workflow has been built, trhough the API, it is now possible to train any robot thanks to a test bench. However, some tasks needs to be explored :

- Changing the dataset tick to a 5-minutes baselineski
- Add the S&P500 market and VVIX index
- It has been suggested to use the following signal :

  (a)

$$\begin{aligned} a_{\text{signal}} &= VVIX_{\text{index}} \times VIX_{\text{index}} \times a_{\text{signal}} \\ &= \frac{VVIX_{\text{index}}}{VIX_{\text{index}}} \times a_{\text{signal}} \\ &= VIX_{\text{index}} \times VVIX_{\text{index}} \,(\text{ratio}) \end{aligned}$$

  (b)

$$b_{\text{signal}} = \ln\left(VVIX_{\text{index}}\right) - \ln\left(VIX_{\text{index}}\right)$$

- Fine-tune another time with the new training features