Report – Reinforcement Learning for Stock

Jakub Kawka, Zofia Stateczna, Jakub Zajac, Alicja Kałuża

January 7th 2026

1 Introduction

The goal of this project was to create a reinforcement learning agent to invest in stock for the user. To aid the model in decision-making an additional LSTM model was trained to predict future stock prices. Based on these predictions, the reinforcement learning agent was trained to make investment decisions that maximize investment profit.

2 Architecture Description

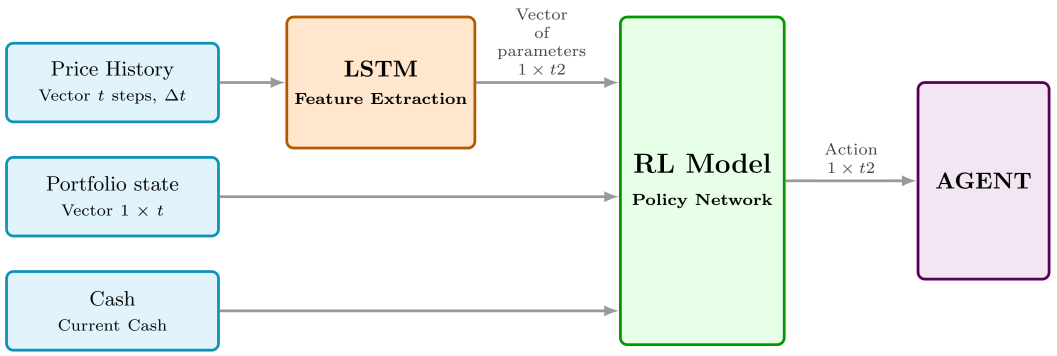


Figure 1: Diagram of architecture of Model

The project uses an observation space of account balance, portfolio state and price history. The price history is also processed to predict the future prices based on previous market trends. Price predictions are achieved using a stacked LSTM model and the RL Model internal architecture depends on the experiment conducted. Based on these predictions the agent makes decisions to hold, sell or buy stock.

3 LSTM Market Predictor

The LSTM Predictor model was prepared using the PyTorch *LSTM* layer. The input is closing price data of 60 previous days. The model's output is a single value representing the predicted next day closing stock price. An acceptable result would be an accurate prediction of the market trends in the 30 days following the end of training data.

The model was trained on the stock prices of Apple.

Initially mean square error was chosen as the loss and accuracy metric for the model.

To ensure performance of the model its parameters were chosen experimentally. The first considered parameter was the amount of stacked LSTM layers. This parameter sets how many separate LSTM layers are stacked one after another by PyTorch during inference. Experiments shown on figures 2 and 3 showed that more stacked LSTM layers resulted in closer representation of test output.

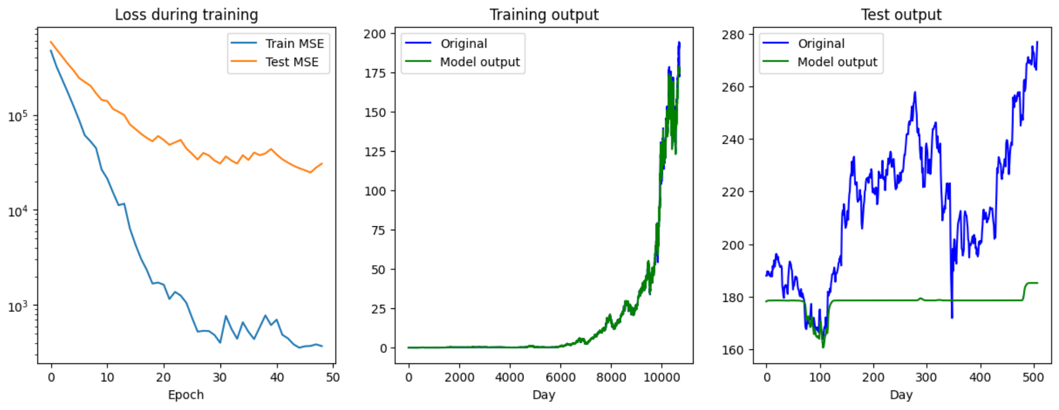


Figure 2: Training results with a single LSTM layer

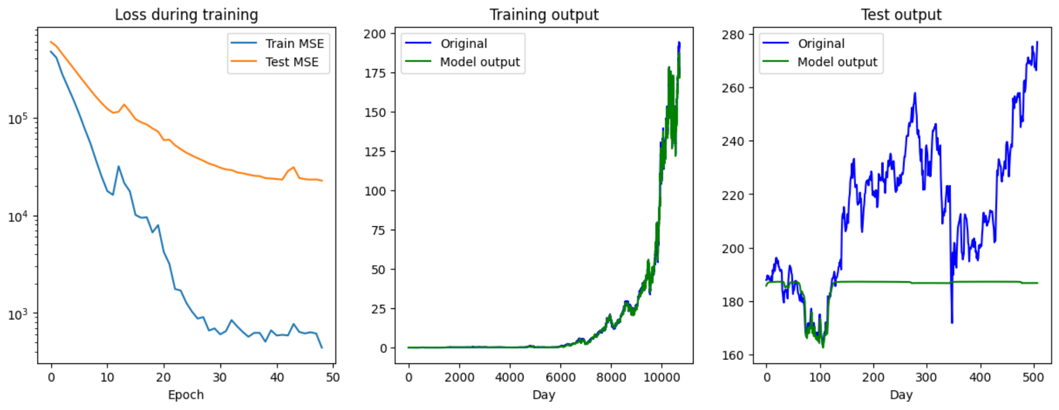


Figure 3: Training results with 5 stacked LSTM layers

Next the amount of hidden values inside the LSTM layers was considered, again increasing the amount of hidden values increased the quality of model outputs, however at larger values the increased training time of the model did not produce results which would warrant longer execution times. The results are shown on figures 4, 5 and 6.

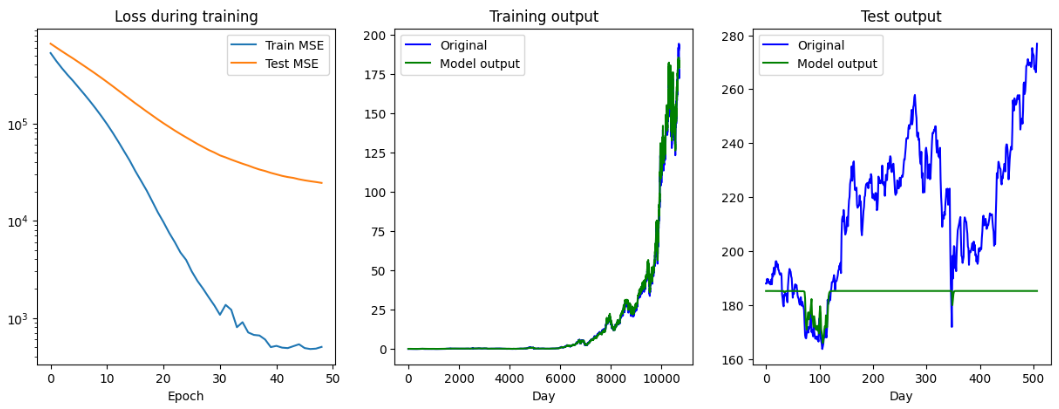


Figure 4: Training results with 32 hidden values

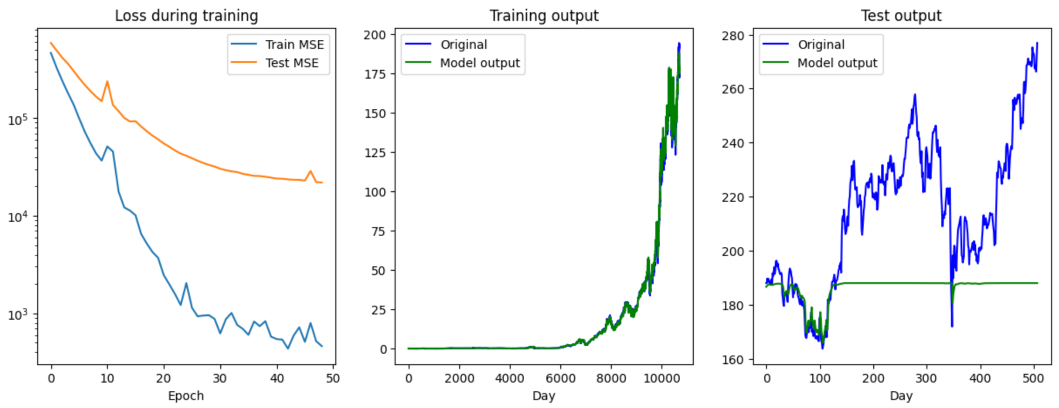


Figure 5: Training results with 64 hidden values

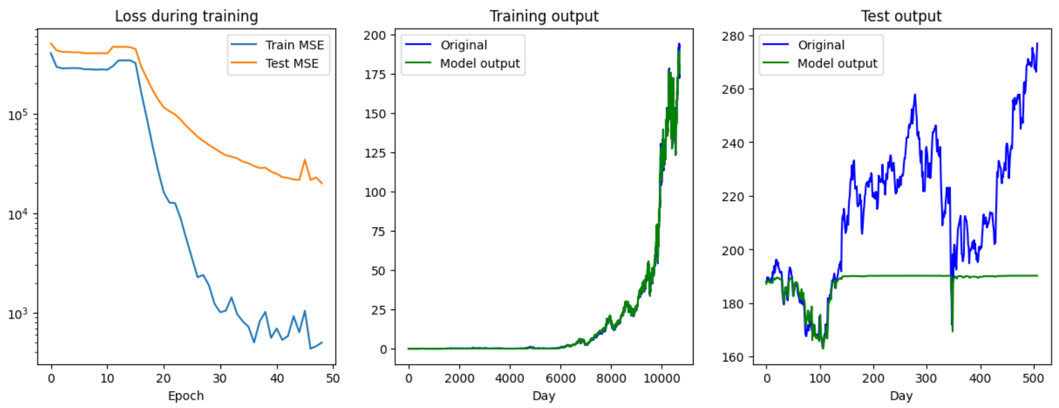


Figure 6: Training results with 128 hidden values

3.1 Combining the results into a working model

Considering the influence of the model's structure on the output quality the final model was created with the following parameters:

- 3 LSTM layers
- 96 hidden values in each layer
- 60 days of data before prediction
- output of 1 predicted day

The model was trained on MSE and RMSE losses. The results are shown on figures 7 and 8.

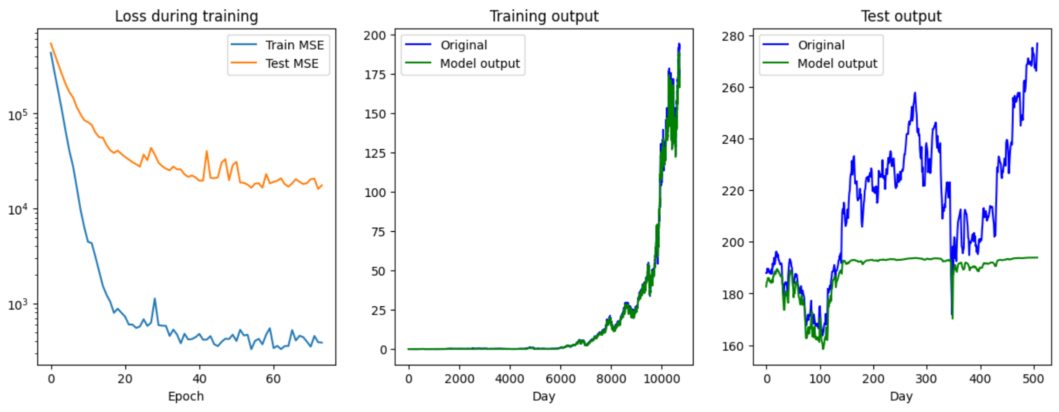


Figure 7: Final chosen structure training with MSE

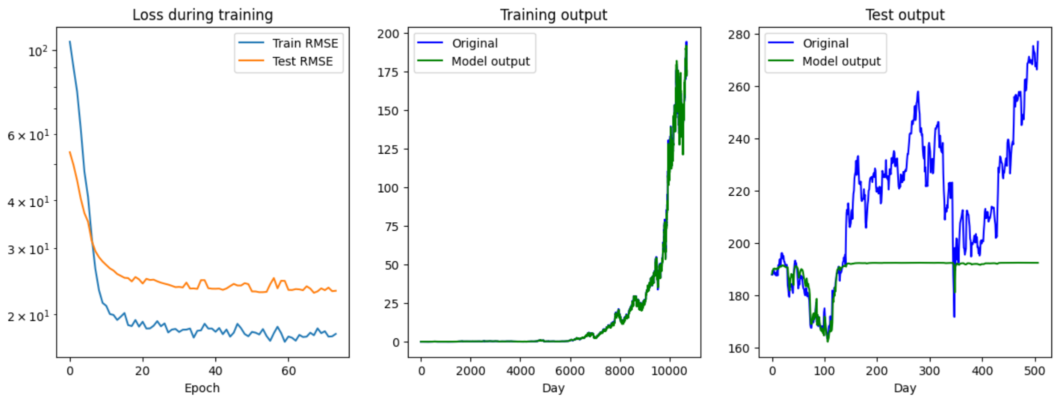


Figure 8: Final chosen structure training with RMSE

As RMSE provided better results with the chosen model structure, the RMSE LSTM model was chosen as the one to use in training of the final RL agent.

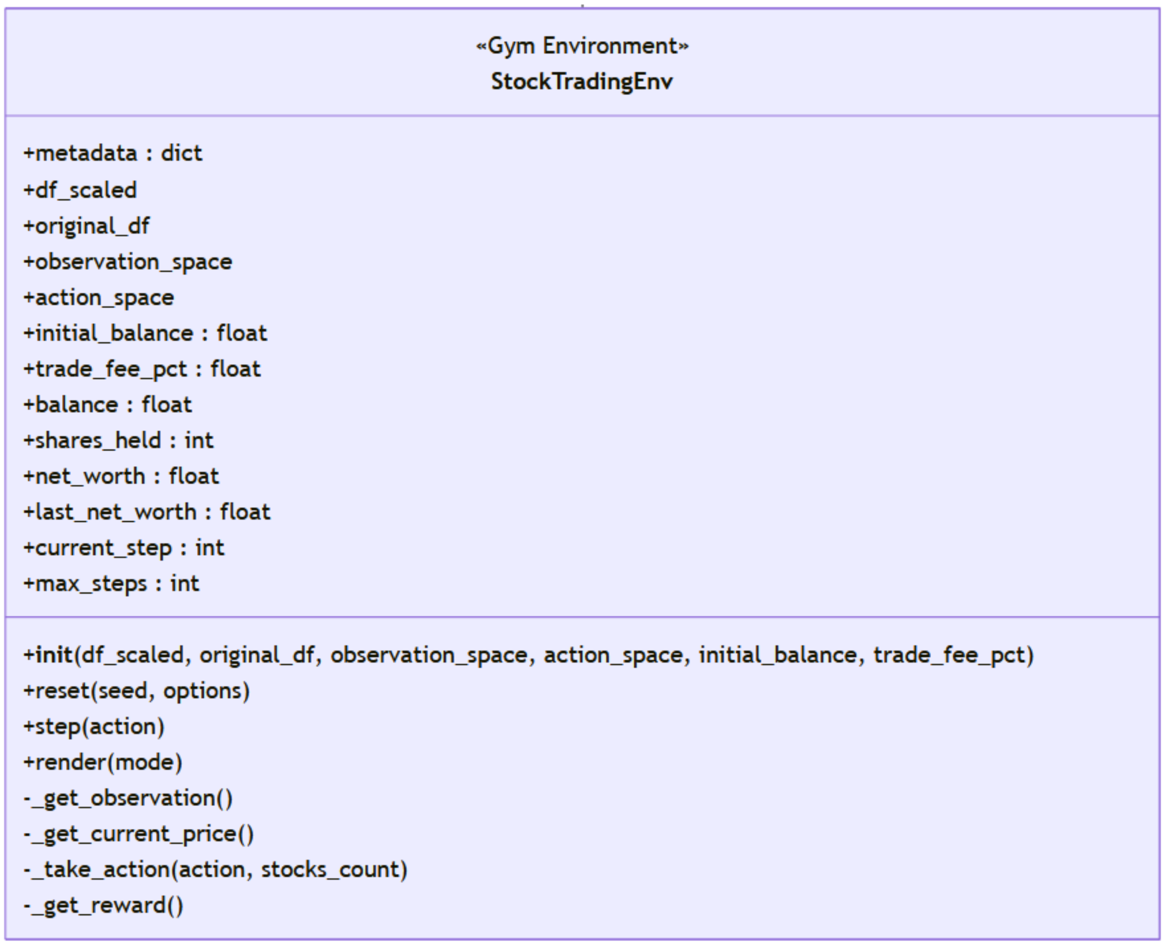
To allow for development of the environment and initial assessment an LSTM model based on a single layer and 64 hidden values was also provided. The finalized version is referred to as the upgraded LSTM model.

4 Reinforcement Learning Environment

The RL agent environment was built iteratively using small steps and frequent testing of the results. The following sections describe this process.

4.1 Empty environment

Initially, a minimal environment was implemented using the Gymnasium framework with randomly generated data and a discrete action space consisting of three basic actions. This initial setup allowed for validation of the environment dynamics independently of any learning algorithm. The agent's observation space was defined as the current time window and the action space was defined as: sell, buy, hold.



```
«Gym Environment»
StockTradingEnv

+metadata : dict
+df_scaled
+original_df
+observation_space
+action_space
+initial_balance : float
+trade_fee_pct : float
+balance : float
+shares_held : int
+net_worth : float
+last_net_worth : float
+current_step : int
+max_steps : int

+init(df_scaled, original_df, observation_space, action_space, initial_balance, trade_fee_pct)
+reset(seed, options)
+step(action)
+render(mode)
-_get_observation()
-_get_current_price()
-_take_action(action, stocks_count)
-_get_reward()
```

Figure 9: StockTradingEnv

4.2 Reinforcement Learning Environment with sinus

Subsequently, the environment was modified to operate on a deterministic sinusoidal price signal,

$$p(t) = 1 + \sin(0.1t), \tag{1}$$

which enabled visual inspection of the agent's behavior and reward dynamics through plotted trajectories.

The agent achieved satisfactory results when evaluated on the sinusoidal price curve, particularly considering the highly constrained observation space. The agent did not have access to information regarding the proportion of the portfolio invested in stocks, the proportion held as cash, nor whether any position was currently open. Furthermore, the agent was only able to modify the portfolio balance by a single unit at each time step, which significantly limited the range of possible actions.

Despite these constraints, the agent successfully increased the portfolio balance to a final value of 177.15. As illustrated in the figure below, the learned policy exhibits correct qualitative behavior, with the agent selling near price peaks and buying near price troughs.

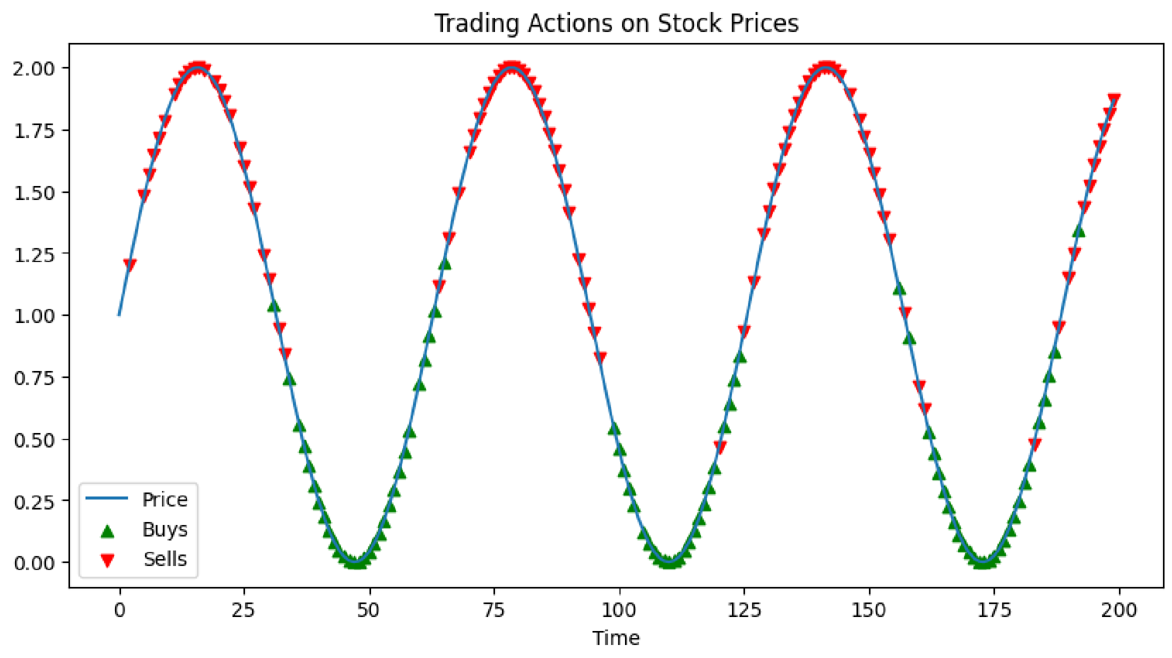


Figure 10: Trading Actions on Stock Prices for beginning

After verifying correct interaction between the agent and the environment, the action space was expanded to include six distinct actions, enabling more fine-grained control over trading decisions. The action mapping used in the environment is shown below:

```
self.action_map = {
    0: ("HOLD", 0),
    1: ("BUY", 1),
    2: ("BUY", 5),
    3: ("BUY", 10),
    4: ("SELL", 1),
    5: ("SELL", 5),
    6: ("SELL", 10),
}
```

Under this configuration, the model achieved a final portfolio balance of 264.73, indicating that the increased action space had a positive impact on overall performance.

However, this modification also introduced greater variability and instability in the agent's decision-making process. Clear separation between buying and selling phases was no longer observed. This behavior suggests that, despite improved profitability, the agent lacked sufficient information about its own portfolio state when operating under a more complex action space. In particular, the absence of explicit portfolio-related features limited the agent's ability to make consistent and structured trading decisions.

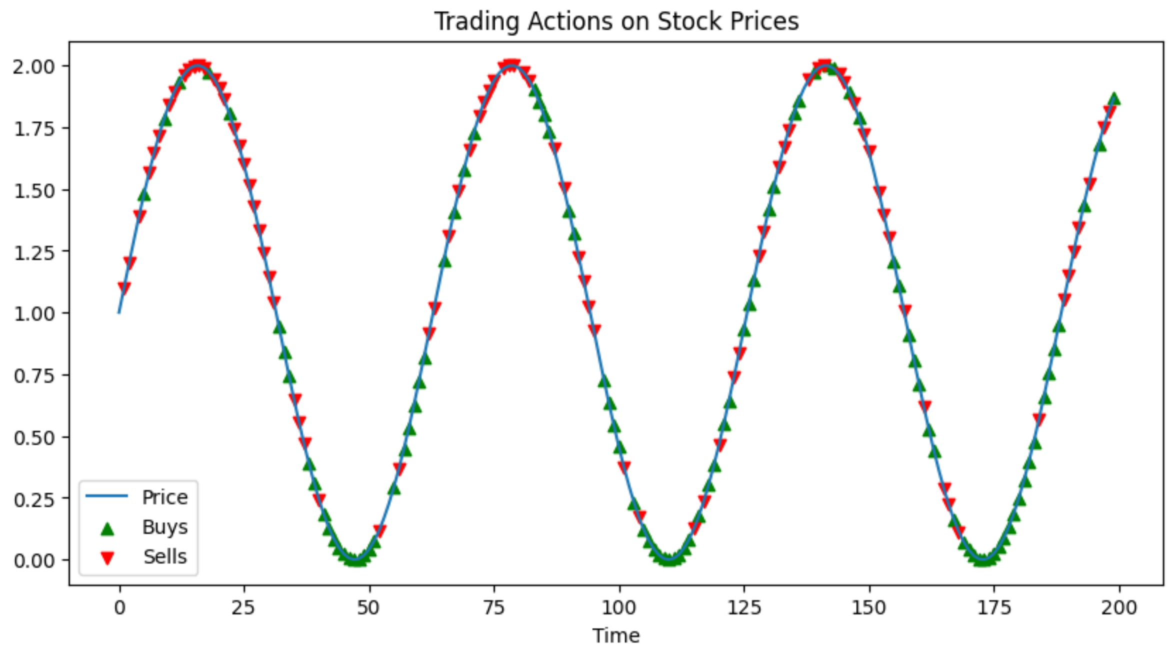


Figure 11: Trading Actions on Stock Prices for bigger action space

To improve the agent's decision-making, the observation space was expanded to include previously missing information related to the portfolio's state. The observation vector was augmented with the following elements:

$$[\text{market\_data}, \text{position\_flag}, \text{position\_ratio}, \text{cash\_ratio}],$$

where *market_data* represents information about the stock price at the current time step, *position_flag* is a binary indicator of whether the agent currently holds a position in the stock, *position_ratio* is the percentage of the portfolio invested in stocks, and *cash_ratio* is the percentage of the portfolio held in cash.

This modification achieved the desired effect, resulting in a total portfolio profit of 609.14. Furthermore, the agent's behavior, illustrated in the figure below, demonstrates the expected and repeatable pattern, with buying occurring near price troughs and selling near price peaks.

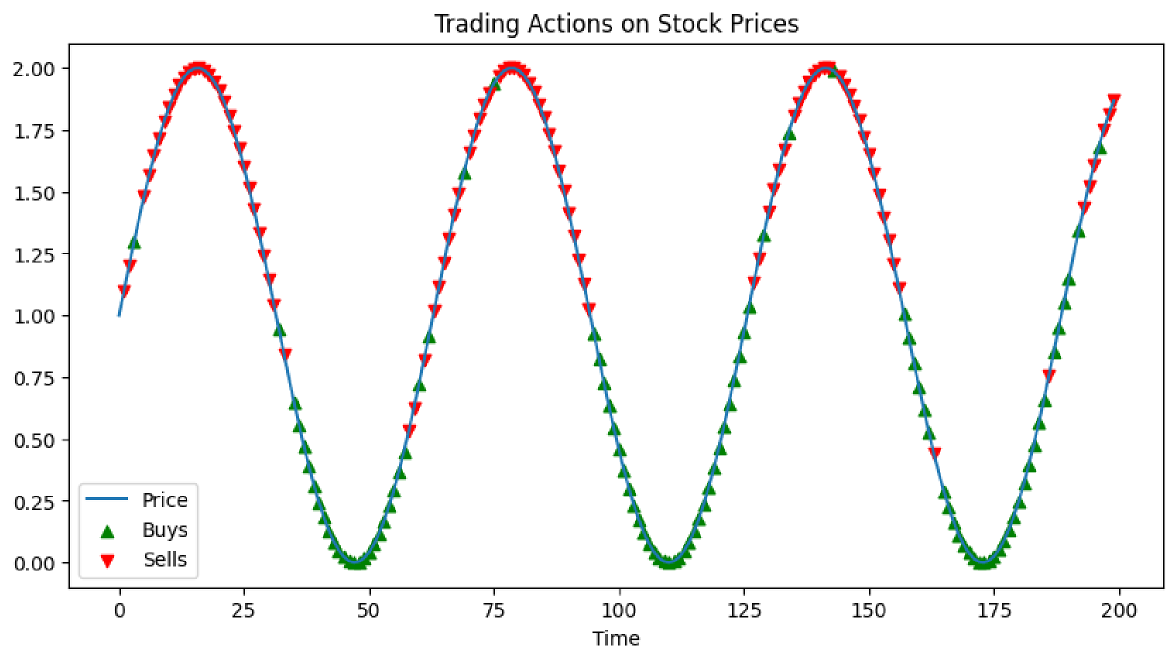


Figure 12: Trading Actions on Stock Prices for sinus

4.3 Reinforcement Learning Environment with real data

The environment and model were tested on real stock market data. Initially, the LSTM predictions were not included; instead, the agent received only normalized stock prices in the observation space. The goal of this preliminary experiment was to determine whether the agent could identify any patterns and generate profit. Using this setup, the agent achieved a total profit of 45.88.

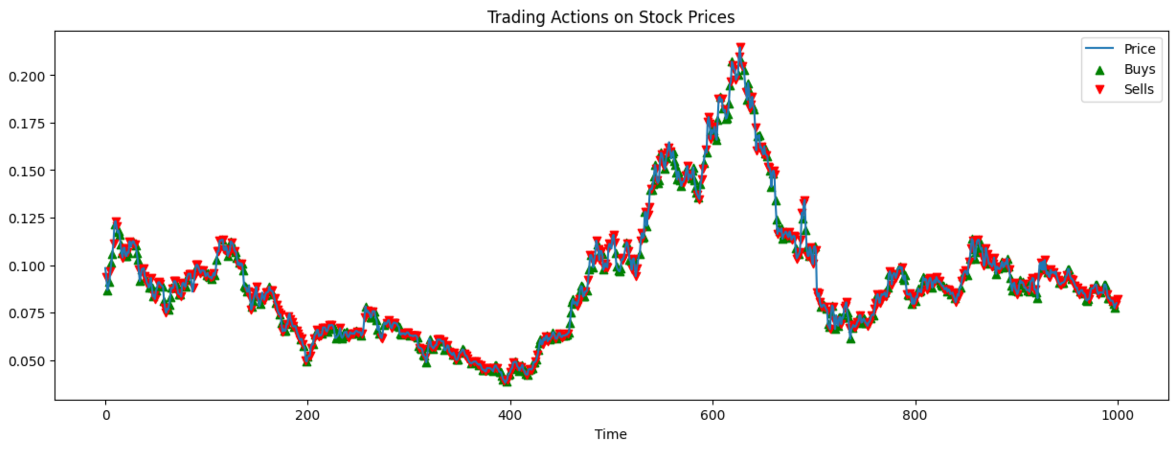


Figure 13: Agent's decisions on normalized stock prices without LSTM.

A potential improvement was expected after incorporating LSTM-based predictions. To verify the framework, a mock fake LSTM was initially introduced, providing the agent with a single time-step ahead in the observation space. With this setup, the agent achieved a total profit of 13.76, which is lower than without the LSTM. This decrease is likely due to the limited time horizon: a single day ahead is insufficient for detecting meaningful trends, whereas a longer horizon, such as one week, would allow the agent to anticipate stock price increases or decreases. These assumptions were later confirmed, as the agent earned a total profit of 96.14 when provided with an extended window of predictions.

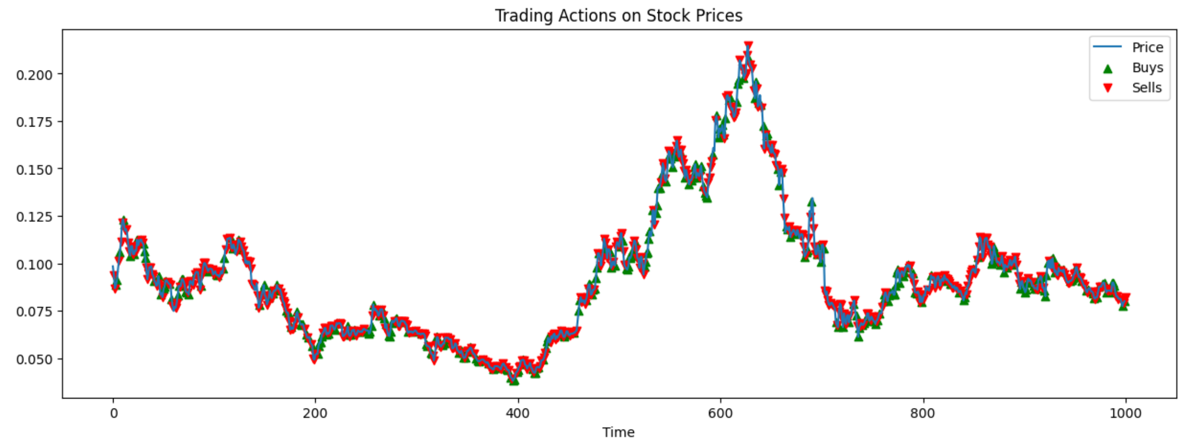


Figure 14: Agent's decisions with a mock LSTM providing one-step-ahead prediction.

Building on these results, the agent was then provided with a real LSTM prediction based on a 60-day time window, forecasting the stock price one day ahead. With this real LSTM, the agent achieved a total profit of 22.01, showing a clear improvement compared to the mock LSTM. An additional experiment was conducted using a seven-day prediction horizon, but the total profit dropped to 0.03. This outcome indicates that the used LSTM model was not sufficiently accurate for longer-term forecasts, as the predictions diverge significantly from the actual stock prices.

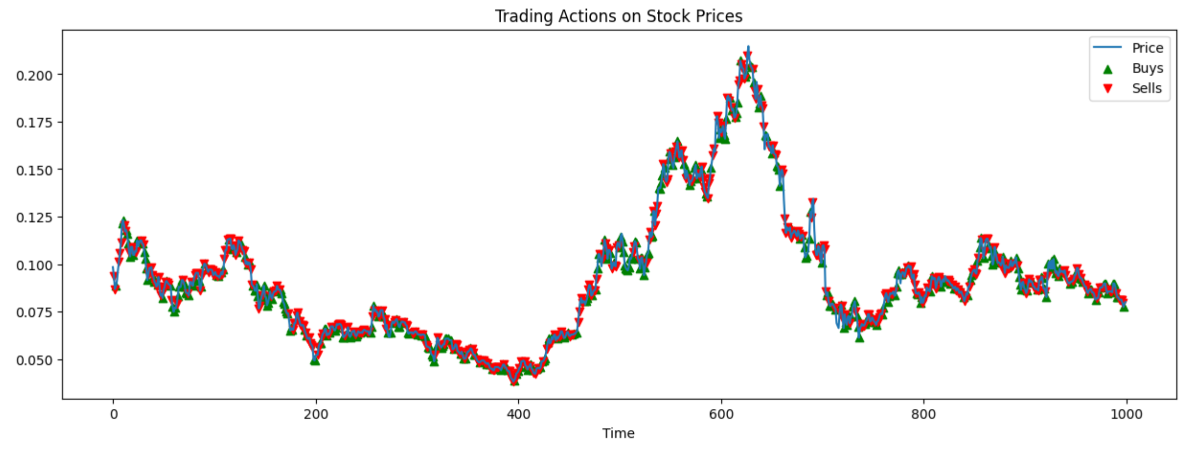


Figure 15: Agent's decisions with real LSTM predictions: one-day.

4.4 RL Environment with upgraded LSTM

The previous experiments were conducted using the initial LSTM model, the following pass of experiments was conducted using an improved version which was created through experimentation with model structure.

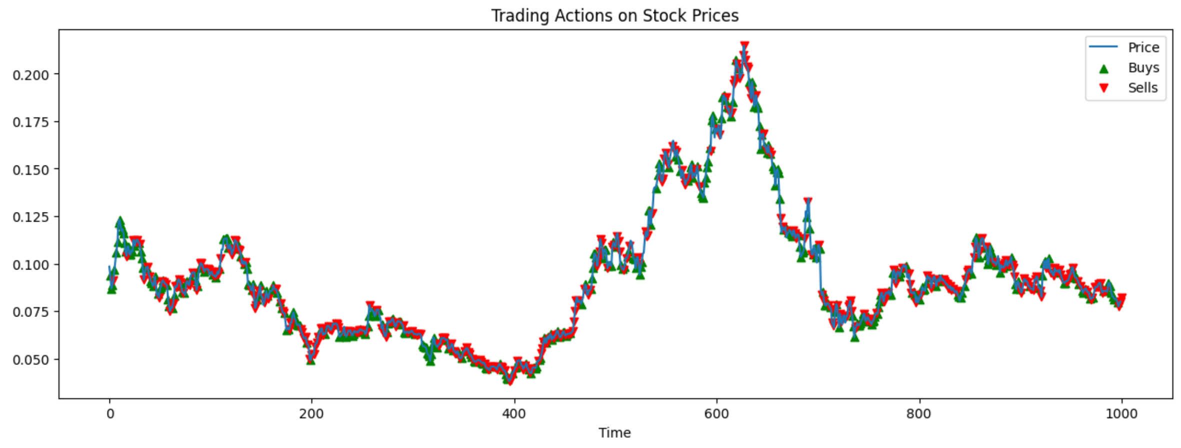


Figure 16: Agent decisions with an improved LSTM model

Using the new model with 7 days ahead prediction the agent was able to score results of 77.31 total profit.

4.5 Discussion of Results

All experiments were initialized with a starting portfolio balance of 100. The results presented in Table 4 summarize the total profit achieved by the agent under different configurations of the environment and observation space.

Table 1: Summary of total profits for different experiments

Experiment	**Description**	**Total Profit**
1	Sinusoidal prices, minimal observation space	177.15
2	Sinusoidal prices, expanded action space (6 actions)	264.73
3	Observation space augmented with portfolio info	609.14
4	Stock prices only (no LSTM)	45.88
5	Mock LSTM, 1-day ahead prediction	13.76
6	Mock LSTM, 7-day prediction window	96.14
7	Real LSTM, 60-day window, 1-day ahead prediction	22.01
8	Real LSTM, 60-day window, 7-day ahead prediction	0.03
9	Improved LSTM, 60-day window, 7-day ahead prediction	77.31

The highest profit (609.14) was obtained when the observation space was augmented with portfolio-related features, including market data, a position flag, the proportion of the portfolio invested in stocks (position ratio), and the proportion held in cash (cash ratio). This indicates that providing the agent with explicit information about its portfolio state significantly improves decision-making and overall profitability.

The mock LSTM experiments confirmed that even limited predictive information can help, but the size of the prediction window must be sufficient to capture meaningful trends.

After improving the LSTM predictor the possible total profit increased significantly. It shows that the agent is capable of using the model's predictions to its advantage and the quality of predictions has an impact on the overall performance.

5 Reinforcement Learning Agent

We experimented with Reinforcement Learning model algorithms and parameters to check if we can improve score of model investment.

Table 2: Summary of total profits for different Experiments with RL for LSTM 7-day ahead prediction

Experiment	**Description**	**Total Profit**
1	PPO, two layers(128 neurons), LR=0.0003	21.95
2	PPO, two layers(128 neurons), LR=0.0001	26.09
3	A2C, LR = 0.0007 (default parameters)	0.12
4	A2C, two layers(128 neurons),LR=0.0001	22.12

Table 3: Summary of total profits for different Experiments with RL for LSTM 1-day ahead prediction

Experiment	**Description**	**Total Profit**
1	PPO, two layers(128 neurons), LR=0.0003	26.66
2	PPO, two layers(128 neurons), LR=0.0001	32.39
3	A2C, LR = 0.0007 (default parameters)	7.50
4	A2C, two layers(128 neurons),LR=0.0001	20.60

Table 4: Summary of total profits for different Experiments with RL with Stock prices only(no LSTM)

Experiment	**Description**	**Total Profit**
1	PPO, two layers(128 neurons), LR=0.0003	53.45
2	PPO, two layers(128 neurons), LR=0.0001	94.64
3	A2C, LR = 0.0007 (default parameters)	3.12
4	A2C, two layers(128 neurons),LR=0.0001	43.74

The results demonstrate that PPO is better algorithm than A2C across both prediction horizons. Lower learning rates lead to improved performance for both algorithms. PPO has a clipped objective function that constrains policy updates, significantly improving training stability compared to standard policy gradient methods such as A2C. As for number of neurons in layer, when using 64 the results are very low. The network capacity was too small to capture the underlying market structure. The 1-day ahead prediction setup yields higher total profits compared to 7-day horizon, indicating that shorter-term forecasts provide more reliable trading signals . PPO remains the more stable and robust approach.

6 Bibliography

- Link to repository with project: AGH-mla
- Link to Gymnasium Documentation: gym-lib
- Link to Proximal Policy Optimization: PPO