BUILDING A REINFORCEMENT LEARNING MODEL IN STOCK TRADING ACTIVITY

Project - Report

Group 5

I. Member:

Hoàng Thanh Lâm - QE170013

Trương Phước Trung - QE170090

Trương Quyết Thắng - QE170008

Trần Tiến Đạt - QE170053

Dương Thành Duy - QE170105

II. Introduction:

1.1. Problem

The Vietnamese stock market is experiencing rapid growth, attracting significant interest from investors, with over 7.46 million trading accounts as of July 2023 and notable growth in newly opened accounts. Predicting market trends is a major challenge, especially given the existence of longstanding investment theories such as Value Investing and the Efficient Market Hypothesis (EMH). Value Investing, originating from “Security Analysis” (1934), and Fama’s EMH (1965) laid the groundwork for many modern financial models.

With advancements in technology and artificial intelligence (AI), online trading and algorithmic trading are becoming increasingly common, and over 53% of investors believe AI and machine learning are the most promising technologies for financial investment. Modern approaches, particularly deep learning models such as Recurrent Neural Networks (RNN), Generative Adversarial Networks (GAN), semi-supervised learning, transfer learning, and Reinforcement Learning (RL), have been applied to stock prediction.

Research on Reinforcement Learning (RL) is a promising direction for the Vietnamese stock market, with the goal of developing automated trading tools for investors.

1.2. Research Objectives and Scope

The objective of this research is to build a model to predict trends in Vietnamese stock prices, using financial and computer science knowledge. The model relies on historical stock price data to predict whether the price will increase or decrease the next day.

Due to limited resources, the research will:

- Exclude the impact of economic factors and social media information.
- Ignore transaction costs in the model.
- Assume that trading decisions do not affect stock prices.

The research aims to contribute to algorithmic trading with machine learning, using reinforcement learning instead of traditional supervised learning. The expectation is for the model to perform well in training and testing, while also laying the groundwork for a fully automated trading bot deployable in real-world markets in the future.

1.3. Tasks

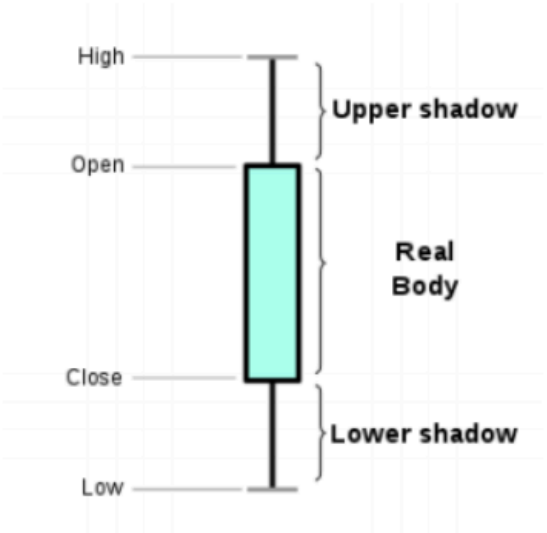
In this report, the student sets out four key tasks to achieve upon completing the research:

- Deeply study foundational knowledge in finance and reinforcement learning methods for application in stock price trend prediction,
- Conduct a comprehensive overview and detailed review of previous research on this topic,
- Propose an integrated model to address the problem,
- Implement the model.

1.4. Candlestick Chart

A candlestick chart displays the highest price (high), lowest price (low), opening price (open), and closing price (close) of a stock at a specific point in time. The

area between the opening and closing prices is called the "real body." If the body is green, it indicates that the closing price is higher than the opening price, known as a bullish (upward) candle. Conversely, if the body is red, it indicates a bearish (downward) candle, where the closing price is lower than the opening price.



The area between the highest price and the closing price (for an upward candle) or the opening price (for a downward candle) is called the "upper shadow." The area between the lowest price and the opening price (for an upward candle) or the closing price (for a downward candle) is called the "lower shadow**".**

III. Proposed model

3.1. Analysis requirement:

The stock market is characterised by uncertainty and constant change. Prices on the stock market fluctuate greatly, influenced by many different factors: from the financial results of enterprises, national economic policies, the impact of major shareholders or the comments of financial experts on the public and many other factors.

In this mini project, we explore the application of Q-learning, a reinforcement learning algorithm, for predicting stock price movements. Using historical closing prices, we train a Q-learning agent to make buy, sell, or hold

decisions, with the objective of maximising profits within a simulated trading environment.

We will define input and output as follows:

- Input: Stock price at closing for the last 10 days.
- Output: Agent will act to sell, buy or hold for maximising profits within a simulated trading environment.

3.2. Model architecture

3.2.1. Q-Learning Model

Q-learning is a popular model-free reinforcement learning algorithm used to train agents to make optimal decisions in a given environment. It's designed to maximise the total reward an agent can receive by learning the best actions to take in different situations over time.

We choose to use Neuron Network for Agent:

```
#Q-Learning Model
print(agent.model.summary())
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
dense_4 (Dense)	(None, 64)	704
dense_5 (Dense)	(None, 32)	2,080
dense_6 (Dense)	(None, 8)	264
dense_7 (Dense)	(None, 3)	27

```
Total params: 3,075 (12.01 KB)
Trainable params: 3,075 (12.01 KB)
Non-trainable params: 0 (0.00 B)
None
```

Input Layer: A dense layer with 64 neurons, taking input based on self.state_size and using the ReLU activation function.

Hidden Layers:

- Two additional dense layers: The first hidden layer has 32 neurons.
- The second hidden layer has 8 neurons.
- Both layers also use ReLU activation.

Output Layer: A dense layer with a number of neurons equal to self.action_size, using a linear activation function, suitable for regression tasks.

Compilation: The model is compiled with mean squared error (MSE) as the loss function and the Adam optimizer with a learning rate of 0.001.

MSE Formula

The formula for MSE is the following.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Where:

- n = number of data points (samples)
- y_i = actual value (ground truth)
- $\hat{y}_i$ = *predicted value by the model*

This architecture is suitable for regression tasks, such as in reinforcement learning scenarios, where it predicts values for various actions based on given states.

Agent Definition:

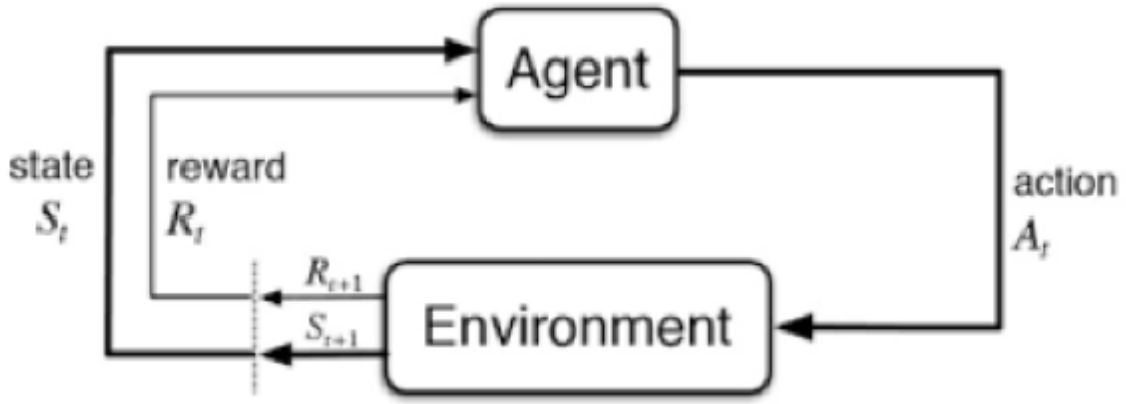
- The agent uses a neural network to approximate Q-values, with layers optimised for effective decision-making.
- Key functions include:

- Act: Chooses actions based on Q-values and exploration.
- Memory: Stores experiences for replay during training.
- Reward Calculation: Updates Q-values based on rewards.
- Policy Update: Balances exploration and exploitation

Environment for stock prediction:

MDP (Markov Decision Process) for Stock Price Prediction:

- Agent - An Agent A that works in Environment E
- Action - Buy/Sell/Hold
- States: Data values
- Rewards - Profit / Loss



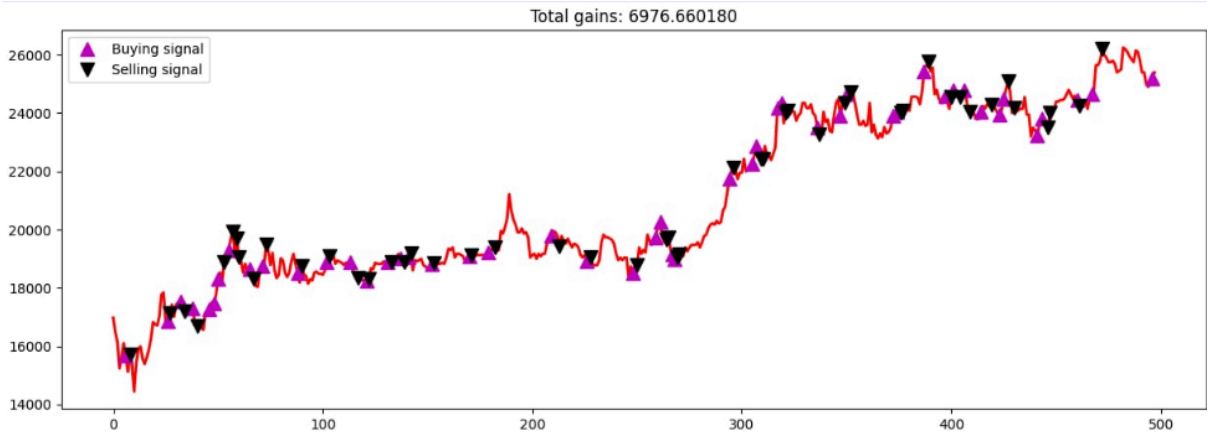
4.2.2. Model parameter

Parameter	Description	Purpose
State size	Represents the size of the input state vector (number of features used to describe the state)	Defines the input dimension for the neural network, helping the agent understand the current environment state.
Action size	The number of possible actions the agent can take, set to 3 (buy, sell,	Determines the output dimension of the neural network, with each

	hold).	output representing the Q-value of an action.
Memory	A deque with a maximum length of 1000 for storing past experiences (state, action, reward, next state).	Enables experience replay by allowing the agent to learn from past experiences, enhancing learning stability and efficiency.
Inventory	A list to track stocks the agent has bought.	Simulates the agent's inventory in a trading environment, helping manage and decide when to sell purchased stocks.
Model name	A string specifying the filename of a pre-trained model.	Allows loading a model for evaluation if is_eval is True, so the agent can perform without needing retraining.
Is eval	A boolean flag to determine whether the agent is in evaluation mode (True) or training mode (False).	If True, the agent loads a pre-trained model; if False, the agent trains a new model from scratch.
Gamma	The discount factor, set to 0.95.	Balances immediate versus future rewards, with values closer to 1.0 emphasising long-term gains.
Epsilon	The exploration rate, initialised at 1.0.	Controls the agent's balance between exploration (random actions) and exploitation (choosing the best-known action).

3.2.3. Helper function

This function plots stock price data along with buy/sell signals and total profit. It will have to overall the result of the Agent.



3.3. Dataset

Collect data strategy:

- According to Dang Quan's article (Dang, 2024), there will be 30 data sets, but we decided to get external data including 3 companies: ACB, FPT, VCB
- Data will be taken from https://finance.yahoo.com/
- Each data set will start from November 1, 2014, ending on October 31, 2024 (Excluding Saturdays, Sundays and holidays)

IV. Implementation and evaluation

4.1. Data preparation

The data used is data of 3 companies ACB, FPT, VCB. Each company will have about 2500 days starting from 11/01/2024, ending on 10/31/2024. Each data set has 5 columns: Date, Open, Low, High, Volume. For simplicity, we focus on closing price.

4.2. Data Processing

Our data does not have any NaN rows. So we decided to add only Label column. This Label column has 3 values (Hold, Buy, Sell) which will be calculated by the formula:

$$\text{label} = \frac{\text{today\_close} - \text{yesterday\_close}}{\text{yesterday\_close}}$$

In which:

Today_close: Is the closing price of today

Yesterday_close: Is the closing price of yesterday

When “label > 0.01” -> Buy

When “label < - 0.01” -> Sell

We don't want to buy/sell when the price has just increased/decreased by about 0.01. If we buy/sell like that, after deducting discounts such as withdrawing money to the account, we will lose. Therefore, we want to take action when it increases or decreases by more than 1%.

4.3. Experimental Setup with Q-learning Model

- **Model:** Q-learning agent with actions (buy, sell, hold) to maximize profit on ACB, FPT, VCB stock data.
- **Training Setup:** Trained on 80% of data with 1000 episodes, using a 10-day price window as input state.

Dataset Used

DATASET	TIME PERIOD	LENGTH	DESCRIBE
ACB, FPT, VCB	03/11/2014 - 31/10/2022	1991 DAYS	TRAIN
ACB, FPT, VCB	01/11/2022 - 31/10/2024	498 DAYS	TEST

- **Hyperparameters**:
  - Episode = 1000
  - batch_size = 32
  - Discount factor = 0.95
  - Exploration rate = 1.0
- **Performance metrics used for the evaluation:**
  - **Total Profit:** Total profit is the cumulative financial gain or loss generated by a trading strategy over a specified period. It is calculated by summing the profits from successful trades (selling at a higher price than buying) and losses from unsuccessful trades (selling at a lower price than buying).
  - **Sharpe Ratio:** The Sharpe ratio quantifies the risk-adjusted return of an investment or trading strategy. It is calculated by dividing the average return of the strategy by its standard deviation. The formula for Sharpe ratio is:

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p}$$

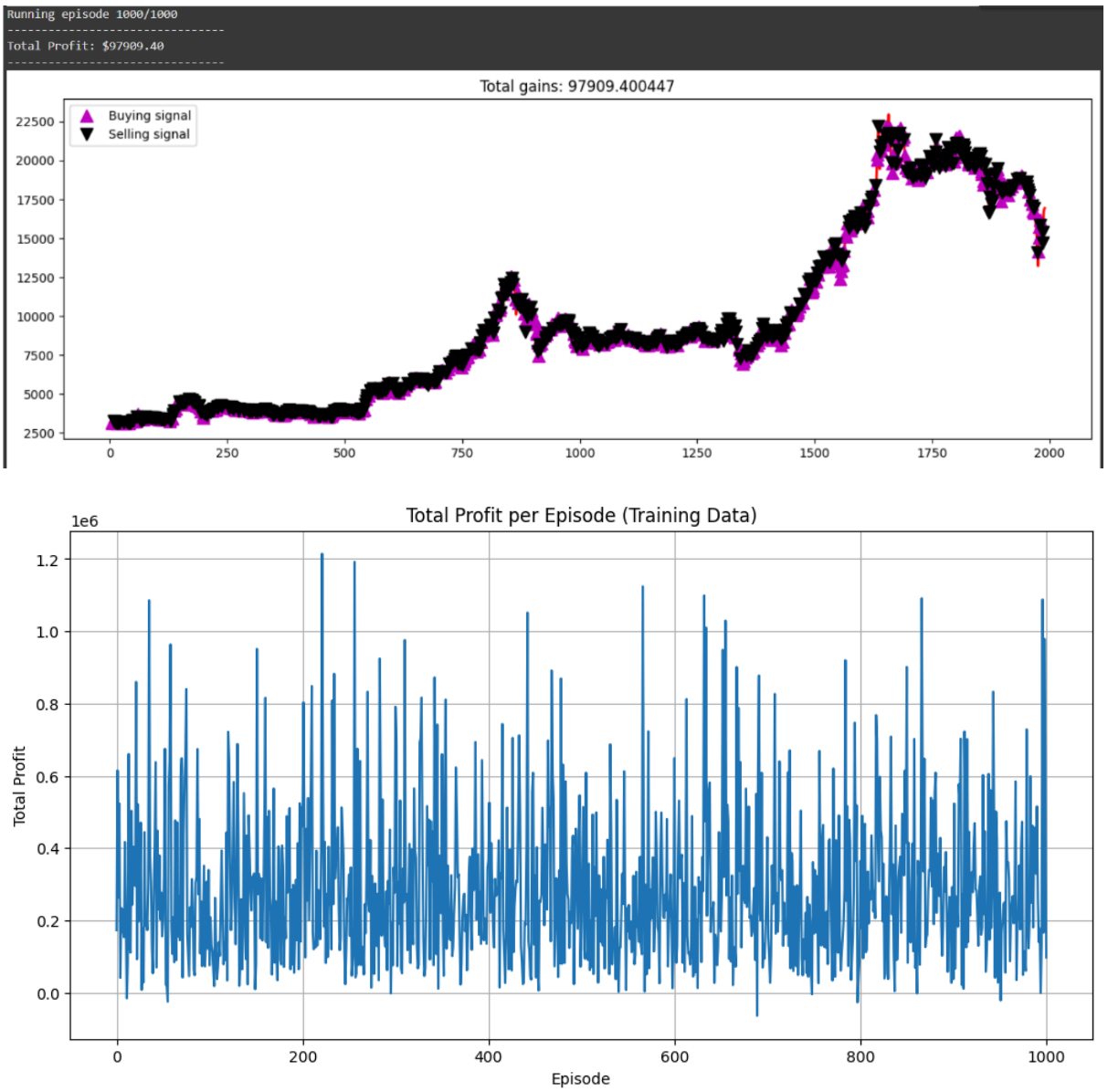
where *Rp* is the average return of the strategy, *RfRf* is the risk-free rate of return, and *σp* is the standard deviation of the strategy's returns.

- **Win Rate:** The win rate is the percentage of profitable trades out of the total number of trades executed by the model. It measures the model's accuracy in making successful predictions and generating profits consistently. The win rate is calculated as the number of profitable trades divided by the total number of trades, multiplied by 100 to express it as a percentage.
- **Baseline metrics:** The baseline metrics offer a comparison to a passive investment approach, like buy and hold. ***Baseline Profit*** calculates the profit from buying and selling without active trading. ***Baseline Sharpe Ratio*** assesses risk-adjusted return, providing a benchmark for the trading strategy's performance. ***Baseline Win Rate*** indicates the percentage of profitable trades, serving as a consistency reference relative to passive investment.

4.4. Evaluation:

4.4.1. Train ACB:

Below picture depicts the model that is the state-action pair graph after 1000 episodes of training and the total profit is also calculated after maximizing the rewards through the learning process.

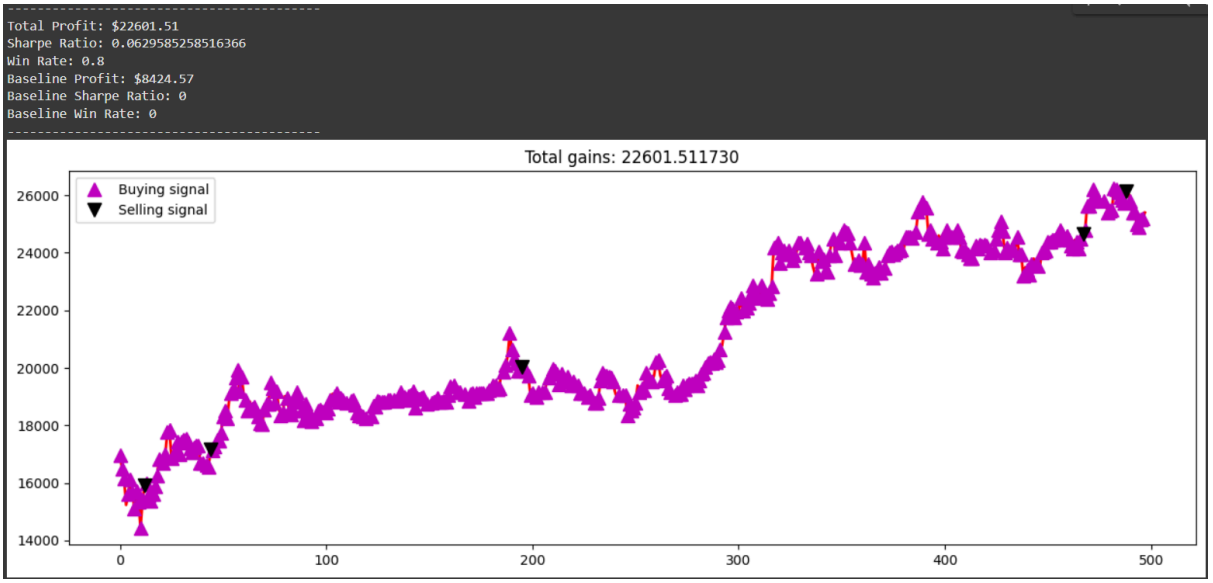


→ Accuracy (percentage of episodes with positive profit): 99.10% - 100%

4.4.2. Test ACB:

The results indicate that the model achieved a total profit of $22601.51, suggesting that it was able to generate significant returns during the trading period. However, the Sharpe ratio of 0.0629585 indicates that the risk-adjusted return may be relatively low compared to the overall volatility of the market. Additionally, the win rate of 0.8 suggests that the model made correct trading decisions approximately 80% of the time.

Comparing these results to a baseline strategy, which resulted in a profit of $8424.57, the model outperformed the baseline significantly in terms of total profit. However, further analysis is needed to assess whether the model's performance is satisfactory considering the associated risks and market conditions.



4.4.3. Result:

Experimental results on Testset

STOCK CODE	ACCURACY	F1-SCORE
ACB	52.50	55.90
FPT	49.06	54.16
VCB	55.65	59.48
AVERAGE	52.40	56.51

V. Conclusion and Discussion

5.1. The pros and cons:

Pros:

1. **Adaptability**: Q-learning allows for adaptive learning, which means it can potentially adjust to changing market conditions over time, making it useful for environments like stock markets that are constantly evolving.
2. **No Need for a Model of the Environment**: Q-learning is a model-free algorithm, so it doesn't require a pre-defined model of stock market behavior. Instead, it learns through interactions with data, which can be beneficial for stock price prediction where true dynamics are complex.
3. **Decision-Making Capability**: As a reinforcement learning method, Q-learning is well-suited for making sequential decisions. It could be applied to develop trading strategies by learning optimal buy/sell actions to maximize reward (profit).

Cons:

1. **High Complexity and Instability**: Stock markets are notoriously difficult to predict due to their noisy, non-stationary nature. Q-learning, especially in its traditional form, may struggle with stability and convergence issues in such an environment.
2. **Data and Computation Intensive**: Q-learning can require extensive data and computation, especially in high-dimensional spaces typical for stock market predictions (e.g., including many technical indicators, multiple stocks, or macroeconomic factors).
3. **Slow Convergence**: Q-learning can converge slowly, especially with a large state-action space, which can make it impractical for real-time stock predictions. It may take a significant amount of time to reach an effective policy.

5.2. Learned through the process of doing this assignment:

Reflecting on a stock price prediction assignment using Q-learning, here's what could stand out as key learnings and potential improvements:

Complexity of Financial Market Modeling: One major takeaway is the inherent complexity and unpredictability of financial markets. Unlike standard Q-learning environments, stock prices are influenced by countless factors, including investor sentiment, global events, and economic indicators, which are

challenging to capture in a single model. This project may reveal the limits of traditional Q-learning in handling such complex, real-world data.

Importance of Data Quality and Diversity: Another important lesson could be the importance of high-quality and diverse training data. Unlike other Q-learning applications where data is often consistent or has controlled noise, financial data is highly variable and prone to anomalies. This makes it essential to have a diverse and large dataset to cover a wide range of scenarios, from bull to bear markets and even crises.

5.3. Potential Improvements if Given More Time:

1. **Integrate More Advanced Models**: If time allowed, transitioning from traditional Q-learning to more advanced reinforcement learning models, like Deep Q Networks (DQNs) or Policy Gradient methods, could improve accuracy. These models tend to perform better in complex environments by leveraging deep learning to approximate the Q-function.
2. **Optimize Hyperparameters Systematically**: Reinforcement learning algorithms, particularly Q-learning, are sensitive to hyperparameters (e.g., learning rate, exploration rate, discount factor). Additional time would allow for more thorough hyperparameter tuning, potentially using automated methods like grid search or Bayesian optimization.

VI. Contribution:

Topic	Team Effort	Hoàng Thanh Lâm	Trương Phước Trung	Trương Quyết Thắng	Trần Tiến Đạt	Dương Thành Duy
Backgrounds of Stock Trading	**100%**	**20%**	**20%**	**20%**	**20%**	**20%**
Backgrounds and related works of RL techniques you used	**100%**	**20%**	**20%**	**20%**	**20%**	**20%**
Your proposed model	**100%**	**20%**	**30%**	**15%**	**15%**	**20%**
Implementation and evaluation	**100%**	**30%**	**20%**	**15%**	**20%**	**15%**
Documentation (technical report, slides)	**100%**	**20%**	**20%**	**20%**	**20%**	**20%**

References

1. Dang, Q. (2024). *Data collection strategies in financial research*. Retrieved from https://finance.yahoo.com/
2. Fama, E. F. (1965). *The Behavior of Stock-Market Prices*. Journal of Business, 38(1), 34-105.
3. Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529-533. doi:10.1038/nature14236
4. Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). Cambridge, MA: MIT Press.

5. Williams, R. J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*. Machine Learning, 8, 229-256. doi:10.1007/BF00992696
6. Zhang, Y., Zohren, S., & Roberts, S. J. (2020). *Deep Reinforcement Learning for Trading*. The Journal of Financial Data Science, 2(2), 6-20.