Final Project COMP 579

Félix Jean
COMP 579
McGill University
Montreal, Canada
felix.jean@mail.mcgill.ca

Abstract

This project explores the application of deep reinforcement learning (DRL) algorithms to optimize trading strategies in the financial markets. Specifically, it examines the effectiveness of Deep Q-Network (DQN), Double DQN (DDQN), and Dueling Double DQN (Dueling DDQN) in navigating complex and volatile market environments. Utilizing a comprehensive dataset of stock prices, trading volume, and economic indicators. The projects reproduce in its own way the paper from by Li et al. [1] by investigating these models' performance against traditional trading strategies like Buy & Hold and benchmarks such as the S&P 500 index. Results indicate that while DRL models can occasionally outperform traditional methods, their success varies significantly across different stocks and market conditions. Notably, simpler models such as DQN often performed comparably or better than more complex architectures, suggesting potential overfitting issues with advanced models. These findings highlight the critical importance of model selection and data diversity in developing robust trading strategies.

1 Introduction

Over the past decades, significant advancements in mathematics and computer science have transformed various industries, yet many traders in the financial markets continue to rely on traditional techniques. This has drawn scientists and mathematicians into the finance sector, sparking a wave of innovation aimed at developing sophisticated trading models. Initially, these efforts focused on mathematical formulas for pricing diverse financial products. More recently, machine learning has been used in their models. Some firms that were able to introduce those techniques had an edge and were able to outperform market averages, achieving exceptional returns.

The last few years have seen remarkable progress in the field of reinforcement learning. This branch of artificial intelligence that learns from its environment by interacting with it through trials and errors have been able in many cases to develop/optimize techniques that no human had thought about. Given its success in other domains, this project explores the potential of deep reinforcement learning algorithms to revolutionize trading strategies in the financial markets. By leveraging these advanced techniques, we aim to determine whether they can similarly detect profitable opportunities in complex market environments.

2 Background

2.1 Data Utilization

For this project, various datasets beyond the standard course material are employed to provide a comprehensive view of stock market behavior. These include daily stock prices, logarithmic prices, trading volume, and volatility (variance). Additionally, longer-term economic indicators such as moving averages (MA), Price Rate of Change (ROC), interest rates, inflation rates, GDP growth, and unemployment rates are considered. These data are can be easily found online and are crucial for analyzing and predicting stock market trends.

2.2 Algorithmic Framework

The project explores advanced deep reinforcement learning algorithms, specifically focusing on Deep Q-Network (DQN), Double DQN (DDQN), and Dueling Double Q-Network (Dueling DDQN). Among these, Dueling DDQN extends beyond the course content and merits a detailed description:

Double Q-Network (DDQN): This modification over the standard DQN aims to mitigate the overestimation of action values. By employing two separate networks to select and evaluate the best action, DDQN provides a more stable and reliable estimation process.

Dueling Network Architecture: This architecture differentiates the evaluation of state values and the advantages of possible actions within those states. It introduces two specific functions:

State Value Function (V): Estimates the value of being in a given state. Advantage Function (A): Calculates the additional value of choosing a particular action from the current state. The Q-value for each action is derived from these two functions. This separation allows for more precise estimation of state-independent action advantages and enhances the policy's decision-making accuracy.

Integration of Both Models: The Dueling DDQN combines the stability of DDQN with the precision of the dueling architecture. By integrating these approaches, the model effectively handles environments where discerning the precise value of actions is critical yet challenging. This integration proves especially potent in complex market conditions, typical of financial trading environments, where small advantages can translate into significant outcomes.

3 Related work

This section reviews significant contributions in the field of financial trading systems that employ machine learning techniques, particularly focusing on reinforcement learning approaches.

The introduction of Double Q-Learning by Van Hasselt et al. (2016)[2] and Dueling Network Architectures by Wang et al. (2016)[3] marked a significant advancement in deep reinforcement learning. These methods significantly reduce the overestimation bias inherent in traditional Q-learning, making them suitable for the volatile environments seen in stock trading.

Recent implementations of these advanced RL techniques in finance have shown promising results. For instance, Zhang et al. (2019)[4] employed a Dueling Double DQN to manage a portfolio of stocks and achieved superior results compared to classical strategies and simpler RL methods. This study underscores the potential of sophisticated RL models to navigate the complex and stochastic nature of financial markets.

This project extends the current body of knowledge by comparing a Dueling DDQN, DDQN and DQN to optimize trading decisions in real-time.

4 Methodology

For this project, three high-volume stocks were selected: Apple (AAPL), Amazon (AMZN), and Tesla (TSLA). The data concerning these stocks were retrieved from Yahoo Finance, which provides historical trading data such as Open, High, Low, Close, Volume, Dividends, and Stock Splits. Additionally, external economic indicators such as Interest Rates, Inflation Rates (Consumer Price Index, CPI), GDP Growth, and Unemployment Rates were obtained using the pandas-datareader library. To further analyze market dynamics, we derived several technical indicators, including Variance, Logarithmic Prices, Logarithmic Returns, Volatility over multiple time frames (10-day and 30-day), Rate of Change (ROC) for 3, 5, 10, and 30 days, and Moving Averages (MA) for 10, 50, 100, and 200 days.

The core of the project involves the implementation of three reinforcement learning algorithms: Deep Q-Network (DQN), Double DQN (DDQN), and Dueling Double DQN (Dueling DDQN). To do this, six Python classes were developed:

QNetwork: A standard neural network class serving as the base model for DQN and DDQN.
DuelingQNetwork: Extends QNetwork to include separate pathways for estimating state values and advantages for actions, specific to Dueling DDQN.
ReplayMemory: Manages memory storage for past experiences, enabling the network to learn from a diverse sample of old states and actions.
DQN, DDQN, DuelingDDQN: These classes implement the respective reinforcement learning algorithms using the models defined above. All neural network models were built using the PyTorch library, while numerical computations were handled by NumPy.

The trading simulations were conducted within a virtual environment created using gymnasium and gym-anytrading. gym-anytrading is a specialized library that adapts OpenAI's Gym interface to the specifics of a trading environment, allowing for a realistic and controlled setting to evaluate the performance of each RL algorithm. This setup enables the models to interact with a simulated market environment where they can execute buy, hold and sell decisions based on the learned strategies. Note, that it can only hold one stock at a time.

Each algorithm was trained on historical data from 2022-06-06 to 2023-11-15, an entire year (365 trading day) worth of data and then tested on the next 90 trading days used for out-of-sample testing to assess performance. The models' effectiveness was measured by the overall profitability. The evaluation metric the total return percentage.

During the initial training phase, the models exhibited issues such as failing to converge and taking random actions. These challenges were primarily addressed by modifying the window size of each training step within an episode. The window size is critical as it determines the amount of historical data considered in each data point, influencing the agent's ability to understand and react to market conditions. For example, a window size of 5 means that the model uses data from today and the previous four days to make trading decisions. Initially, the window size was set to 5 days. However, this smaller window seemed insufficient for the models to capture and learn more complex market patterns effectively. Therefore, the window size was increased to 10 days. This adjustment allowed the models to incorporate a broader range of historical data into their decision-making processes, which improved the training outcomes significantly. The larger window provided a deeper context, leading to better convergence and more strategically sound actions during simulated trading sessions.

5 Experiment

5.1 Hyperparameters

Table 1 details the hyperparameters used for training each of the models across different stock tickers. These parameters are crucial for ensuring that the models learn effectively.

Models	**DQN**			**DDQN**			**Dueling DDQN**		
Ticker	**AAPL**	**AMZN**	**TSLA**	**AAPL**	**AMZN**	**TSLA**	**AAPL**	**AMZN**	**TSLA**
Learning Rate	0.005	0.01	0.01	0.0075	0.0075	0.0075	0.005	0.001	0.0075
Batch Size	8	64	8	16	8	8	32	12	16
Hidden Layers	64x64	64x64	64x64	64x32	64x32	64x32	64x64	64x64	64x64
Discount Factor	0.9	0.9	0.9	0.6	0.6	0.6	0.8	0.8	0.8
Epsilon Decay	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9	0.9
Replay Memory Size	10000	10000	10000	10000	10000	10000	10000	10000	10000
Target Update	10	10	10	10	10	10	10	10	10
# Episodes	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 1: Hyperparameters for each model and stock ticker used in the experiment.

5.2 Benchmark

To assess the performance of the trading models, two benchmarks will be used:

1. Buy-and-Hold Strategy: This strategy involves purchasing a stock at the beginning of the testing period and holding it until the end. It represents a common baseline for evaluating investment strategies.

2. S&P 500 Index Comparison: The models' returns will also be compared against the performance of the S&P 500 index over the same period. This comparison helps to understand the models' effectiveness relative to broader market movements, providing insight into their ability to generate return in varying market conditions.

5.3 Results

Models	**DQN**			**DDQN**			**Dueling DDQN**			**Buy & Hold**			**S&P 500**
Ticker	**AAPL**	**AMZN**	**TSLA**	**AAPL**	**AMZN**	**TSLA**	**AAPL**	**AMZN**	**TSLA**	**AAPL**	**AMZN**	**TSLA**	**Overall**
Returns	-3.64%	21.69%	-3.32%	-2.23%	-3.71%	-26.5%	-11.66%	21.59%	-26.5%	-9.49%	26.29%	-24.74%	16.42%

Table 2: Comparison of returns from 2023-11-16 to 2024-03-28 for each model against the "Buy & Hold" strategy and the S&P 500 benchmark for each ticker.

The results outlined in Table 2 reveal varied performances across different models and stocks. Notably when looking at Figure 1, the DQN model outperformed the Buy & Hold strategy for AAPL and TSLA, indicating selective effectiveness in certain market conditions. Similarly, the DDQN model showed improved performance for AAPL but struggled significantly with TSLA, reflecting a -26.5% return.

These inconsistencies highlight the models' varying ability to generalize across different data patterns. During periods where models surpassed the Buy & Hold strategy, such as with AAPL for DQN, they appeared to capitalize on optimal buying moments rather than adhering strictly to a Buy & Hold approach. This suggests that the models can identify and act on favorable market conditions but may require more comprehensive training data to improve consistency and generalization. To note, that the results after the training process were as expected (Figure 2) as it

Figure 1: Results of the different algorithm when testing on the following 90 trading days

did buy and sell at specific moment where it took advantage of sudden dips, bought while the stock was about to increase, etc.

A potential limitation in data utilization, such as the selection of moving averages, ROC, and macroeconomic indicators, may not have been ideally suited for the chosen tickers or the specific timeframe of the experiment. To enhance future model performance, incorporating a wider array of historical data or diversifying the types of financial indicators could be beneficial. Additionally, expanding the dataset to include more stocks might help the models better learn and adapt to varied market dynamics.

Overall, while the models occasionally outperformed traditional strategies, their performance was not uniformly successful, suggesting room for significant model refinement and data strategy optimization.

Additionally, an intriguing observation from Table 2 is that the DQN model outperformed both the DDQN and Dueling DDQN. This result suggests that the simpler DQN model may be more suited to this specific trading environment. Similarly, DDQN exhibited better performance compared to Dueling DDQN. A plausible explanation for this phenomenon could be that the increased complexity of DDQN and Dueling DDQN leads to greater overfitting on the training data. These more complex models might not adapt as effectively to new, non-stationary data, unlike the simpler DQN model. This underscores the importance of model robustness and generalization, especially in financial markets where data conditions can change unpredictably.

As observed in the study on the application of Deep Reinforcement Learning for Stock Trading Strategies and Stock Forecasting [1], a similar result was noted where the simpler DQN model outperformed more complex variants. This aligns with their observation that "improvements based on the original method are not always better than the original and that each method has its own specific field of application." This underscores the importance of context and suitability when selecting algorithms for practical applications in stock trading, emphasizing that more complex algorithms do not necessarily yield better results in every scenario.

6 Conclusion

This project investigated the application of deep reinforcement learning algorithms, specifically DQN, DDQN, and Dueling DDQN, to optimize trading strategies in the stock market. The findings reveal that while advanced models like DDQN and Dueling DDQN offer theoretical advantages in learning dynamics, simpler models like DQN sometimes provided superior performance in this specific setting. This suggests a potential overfitting issue with more complex models, or it might indicate that simpler models are better suited to adapting to the non-stationary nature of financial data. Despite some successes, the models did not consistently outperform the traditional Buy & Hold strategy, underscoring the challenge of trading in highly volatile and non-stationary market environments. The results also suggest that the choice of data and the timeframe of training significantly influence the effectiveness of the trading strategies derived from these models.

For future research, several avenues appear promising such as Data Expansion and Diversity which consist of incorporating a broader range of data, including more diverse financial indicators and a wider array of stocks, might help improve the models' ability to generalize and perform in different market conditions. Another avenue could be to investigate other RL approaches, such as policy gradient methods or actor-critic models, could offer new perspectives and potentially more robust strategies for financial trading.

7 Figures

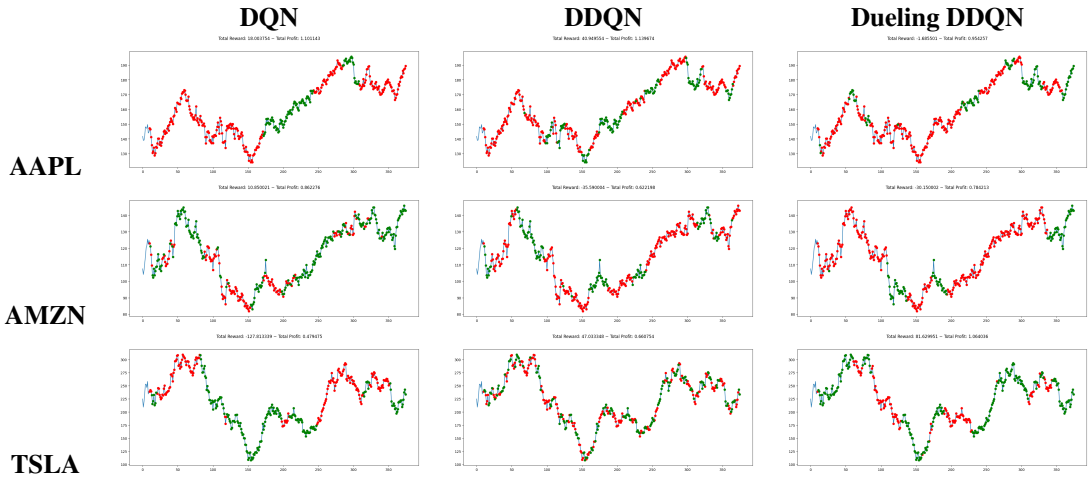

Figure 2: Results after training

8 References

[1] Li, Y., Ni, P. & Chang, V. Application of deep reinforcement learning in stock trading strategies and stock forecasting. Computing 102, 1305–1322 (2020). https://doi.org/10.1007/s00607-019-00773-w

[2] Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 2094–2100.

[3] Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016). Dueling Network Architectures for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 1995–2003.

[4] Zhang, H., Dong, Y., Wang, X., & Lai, K.K. (2019). A deep learning framework for optimizing stock trading strategies. *Journal of Financial Data Science*, 1(4), pp. 86–101.