Exploring Deep Q – Learning For Stock Trading Portfolio

Michael Owino – 245808820
Michael Ajibola - 235849610

Abstract— **We have witnessed the rise of research in math, statistics, quantitative finance and reinforcement learning to help study, analyze and make informed performative strategies based on the complex intricacies in stock market. Automated i.e. rule-based trading strategies backed by data driven and mathematical computations to learn optimal actions to made through trial and error have come a long way. From the foundational beginnings of Traditional Algorithmic trading such as Time Series forecasting to Early Machine Learning adoptions in Trading such as HMMs in Supervised Learning and Topic Modelling (LDA) for Unsupervised Learning to Reinforced Learning such as Q-Learning (Tabular RL) we introduce the exploration of the advanced algorithmic, AI-driven quantitative trading called Deep Q-Learning (DQL) which was influential in shaping modern quantitative trading. We have employed a Deep Neural Network which is an improvement of the Q -Table in Q - reinforcement learning promising better results. Our work dives deep into the architecture to analyze and monitor performance based on a 5 - year stock data used an input.**

Keywords—Q -Leaning, DQL, Deep Q – learning, Deep Neural Network, Topic Modelling, Supervised Learning

I. INTRODUCTION

The stock market is a central mechanism in the global capitalistic market fundamental to its functioning, growth and long-term sustenance. It is a collaborative trading network involving company shares and their derivatives. It very important since it's trading prices are signals for economic strength or weakness [1].

To build up on a stock-trading model, we first need to know the foundational basics such as Statistical Trading, Quantitative Finance and Algorithmic Trading.

Statistical trading under the hood works by analyzing past data in order to make decisions on what to buy and sell. This type of trading does primarily depends on quantitative analysis and the use of data to make informed decisions [2]. For it to successfully work, there are several components of information that is needed such as price movement of the security, the volume of stock traded, the volatility of the security and the trend of the security. Various types of analysis are also used to make optimal traded such as technical analysis which sees the past price movement of security in order to make prediction about the future and also fundamental analysis which look at the underlying value of a security. Statistical trading guides our work on the type data we have used i.e. Stock Trading dataset which we have analyzed to train our model to make potential entry and exit points quickly in order to gain optimal profits [3].

Quantitative Finance also forms the basis of the fundamental study. This involves the applications of mathematical models and extremely large sets to analyze financial markets and securities. The mathematical models are used to price securities and measure risk [4]. This is a very less dominated field which demands a strong background in math and other domains such as algorithms and probability theories. Financial engineering expound father on Quantitative Finance and combines the mathematical theory of quantitative finance with computational simulations to make price, trade, hedge, and other investment decisions. This field requires advanced degrees such as Master or PH. D to become a professional [5].

Algorithmic trading forms the final part of our basis. Also called automated trading, black-box trading or algo-trading. Algorithmic trading utilizes a computer program that uses a defined set of instructions i.e. algorithm to place a trade [6]. Our model is essentially characterized by this property. The defined set of instructions are based on timing, price, quantity or mathematical model. In theory, the trades can generate profits at speed and frequency that is deemed impossible by human stock traders. Apart from profits, algo-trading ensures the market is more liquid and trading more structural by phasing out the effect of emotions on the stock market. An example of how the algorithmic trading works is:

i. First buy 10 shares of stock when its 50-day moving averages goes above 200-day moving average.
ii. Then sell shares of the stock when its 50-day moving average goes below the 200-day moving average.

These two instructions can be fed to computer program, which will execute them and monitor stock price by placing buy and sell order when the stated instructions are met. This makes the

system work without human supervision which leads to certain advantages such as best execution for maximum profits, low latency as trades are time correctly, reduced transactional cost, no human errors and back-testing of trading strategies using available historical data to pick the best trading strategy [7].

From the mentioned baselines above we see the stock market environment is complicated, influenced by many factors and changes rapidly meaning it is hard for human being to fully capture all relationships and patterns to make the best trading decisions in time. Research inspired by the need to create a stock trader algorithm capable of learning from market variables, generating (buy, sell, hold) actions and evaluating itself to improve led to development of the **Q-Learning Algorithm** for stock trading, a type of reinforced learning. It is majorly characterized by iteratively learning and improving over time by taking correct actions [8].

Our model utilizes Deep Q – Learning which is an improvement to Q – Learning. Q – Learning has a limitation i.e. it is mainly used to solve problems where state space if discrete and finite. This is done by constructing a Q – Table recording the q-values for each possible state and action. This works well with to a certain limit [9]. Considering the stock data which is a very large sequential data that generates more information per minute per hour, the Q - Table constructed will be huge and require a lot of computation to update the table values making Q – Learning very resource intensive. Deep Q – Reinforcement Learning was adapted to resolve this problem and that is what our model utilizes.

From the results, our model is able to execute faster since we are using a neural network rather than a Q – table meaning it scales well with the size of data compared to Q – Learning algorithm. The performance of our model on executing trades beats the market benchmark such as S&P500 benchmark or profit growth proving it offers quite profitable trading strategies.

II. Related Work

A. Journey of Development: From Q-Learning to Deep Q-Learning in Stock Trading

The evolution of artificial intelligence in finance revolutionized trading modalities. Among the notable advancements in this landscape is the emergence of Deep Q-Learning, designed and built upon the foundational principles of Reinforcement Learning (RL) and Q-Learning. Let explore the developmental trajectory of Deep Q-Learning, the seminal works that have paved the way for this field and its application in stock trading [10] [11].

B. Introduction to Reinforcement Learning

Reinforcement Learning is a subclass of machine learning that studies how agents should take actions in an environment to maximize the cumulative reward [12]. Whereas in supervised learning each training example has a correct output associated with it, RL is concerned with learning from the consequences of actions taken, making it particularly well suited to dynamic and uncertain environments such as financial markets [13]. RL comprises four key components in its central framework:

- Agent: The decision-maker.
- Environment: What the agent interacts with.
- Actions: Options the agent can take
- Rewards: A feedback signal based on action effectiveness

C. The Birth of Q-Learning

Q-Learning, introduced by Watkins in 1989, is a well-known model-free RL algorithm that addresses the problem of learning the value of actions in various states without needing a model of the environment. The Q-value (quality of an action) is computed using the Bellman equation, describing the expected utility of actions.

A distinguishing feature of Q-Learning is its exploration-exploitation trade-off, allowing agents to balance exploring new actions and leveraging known rewarding actions. This exploration is crucial in environments like trading, where the agent must adapt to ever-changing market conditions [13].

D. The Transition to Deep Q-Learning

Q-Learning has shown its strengths in many controlled environments, but faces challenges in high-dimensional state spaces, like some of those present in complex financial markets. When the number of states (e.g., stock price movements) grows exponentially, storing every state-action pair in a table as in the tabular method of Q-Learning is infeasible [14].

The metamorphosis of Q-Learning into Deep Q-Learning occurred when researchers sought to address its limitations through the power of deep neural networks. The seminal work by Mnih et al. in 2015 titled "Human-level control through deep reinforcement learning" marked a pivotal moment in the field [15].

E. Deep Q-Network (DQN)

Mnih's team introduced the Deep Q-Network (DQN), which combines RL with deep learning to approximate the Q-values using a neural network instead of a table [15]. This breakthrough enabled agents to operate in high-dimensional spaces through techniques such as:

- Experience Replay: Storing past experiences in a replay buffer and sampling from it to break the correlation between consecutive experiences and stabilize learning.
- Target Network: Using a separate target network to stabilize Q-value updates, reducing oscillations during training.

DQN became famous by demonstrating superhuman performance in playing Atari games, where it learned directly from raw pixel inputs and successfully navigated complex game environments. This success showcased deep RL's potential in decision-making scenarios beyond gaming, paving the way for subsequent applications in finance.

F. Deep Q-Learning in Atari Games

Deep Q-Learning has gained prominence through its application in Atari games, where it achieved human-level performance by combining Q-Learning with deep neural networks. This approach can be adapted to stock trading by treating trading decisions as actions in a game-like environment. The utilization of these techniques in stock trading presents a promising avenue due to the inherent non-linearities and stochastic characteristics of financial markets [12] [9].

G. Q-Learning Applied to Stock Trading

Following the success of Deep Q-Learning in games, researchers began exploring its applicability in the finance world (stock trading). Q-Learning for stock trading involves formulating a trading strategy where the agent learns to choose actions based on historical stock price data [9].

H. Deep Q-Learning in Stock Trading

The implementation of Deep Q-Learning has grown significantly in stock trading, marked by several trends [16]:

- Multi-Stock Portfolios: Recent methodologies are focused on managing multi-stock portfolios, optimizing the allocation dynamically based on evolving market conditions.
- Integration with Other Learning Methods: Hybrid approaches combine deep q-learning with supervised learning techniques to achieve higher accuracy in predicting stock movements.
- Transfer Learning: The adaptation of knowledge from one domain to another enables faster convergence and better performance across different stocks and market conditions.

III. Proposed Approach

A. Dataset

We used a dataset containing 5-year stock data (available on Kaggle: https://www.kaggle.com/datasets/ameythakur20/stock-prices [17]), which contains historical stock data for multiple companies over five years. The dataset includes various attributes such as date, open, high, low, close prices, and volume for each stock.

B. Data Analysis

a) Time Series Analysis

Time Series Trend Analysis helps uncover patterns or overall direction in stock prices over time. This is particularly important for agents in Deep Q-Learning, which must learn temporal dependencies and historical context.

Let's analyze the closing price trends for a single company (Apple, AAPL).

Fig. 3.2.1 Closing Stock Price Trend for Apple

We see the visualization above shows the closing price trend of Apple Inc. (AAPL) from 2013 to 2018. We can observe a general upward trend, which is valuable information for a Deep Q-Learning model to detect bullish or bearish market phases. Identifying such long-term trends aids the model in learning optimal actions (buy/sell/hold) based on the state transitions over time.

b) Volatility Analysis

Volatility indicates the degree of variation in stock prices and is a core component in understanding risk. For reinforcement learning models, identifying high-volatility periods can be essential for deciding cautious or aggressive actions.

Let's plot a rolling standard deviation of the AAPL stock's closing prices to show its volatility.

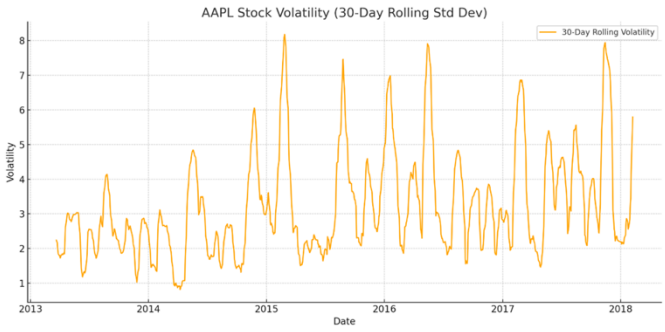

Fig. 3.2.2 Apple Stock Volatility

The chart presents the 30-day rolling standard deviation of AAPL's closing prices, which reflects how much the price fluctuated around its mean over time. Notably, we see spikes in volatility during certain periods, indicating market uncertainty or significant news events. These cues are crucial for a Deep Q-Learning model to modulate its trading aggressiveness based on perceived risk.

c) Volume Analysis

Volume Analysis helps evaluate investor interest and momentum in a stock. High volumes typically indicate strong interest and can precede major price changes—useful signals for a Q-learning agent.

We'll now visualize the trading volume pattern for AAPL over time.

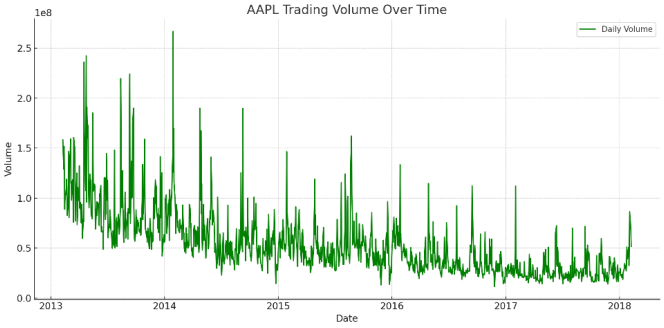

Fig. 3.2.3 Trading Volume for Apple

d) Corelation Analysis

Correlation Analysis determines how the prices of different stocks move in relation to one another. This is useful for portfolio diversification strategies within Q-Learning environments.

Let's compute and visualize the correlation matrix of closing prices for a few selected stocks (AAPL, GOOG, MSFT, AMZN, META(f.k.a FB)).

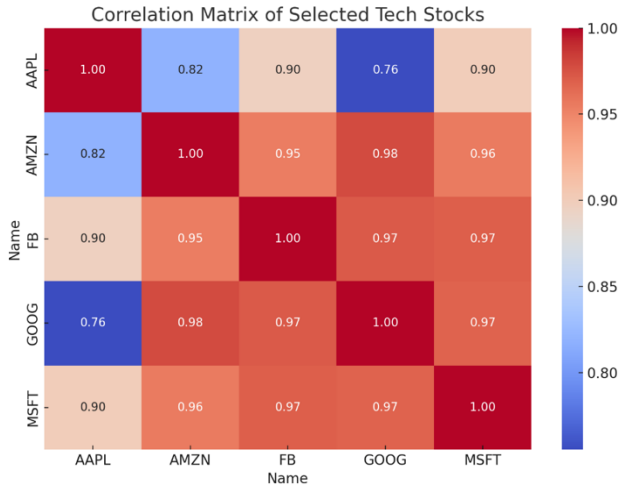

Fig. 3.2.4 Correlation Matrix of Various Stocks

We see this heatmap illustrates how the selected tech stocks' prices correlate with each other. Strong positive correlations (close to 1) indicate that stocks move in similar directions, which is common among large-cap tech firms. Deep Q-Learning agents can use such information to balance portfolio risks by diversifying across less correlated assets or identifying pairs for hedging strategies.

IV. Q – Learning Algortihm Methodology

We describe the architecture and training methodology of our Deep Q-Learning Network (DQN) for algorithmic stock trading. The model is trained to learn an optimal trading policy using historical stock price data, enabling it to make buy, sell, or hold decisions in a simulated market environment. The entire pipeline includes data processing, state representation, action selection via Q-values, reward computation, and model training with experience replay.

To understand how our model works, we can pick one of Atari's games that inspired the use of DQL in finance which is The Mario's Bros i.e. Super Mario. We need to train an agent i.e. Mario so that he can autonomously play the game by himself gaining the maximum number of coins and avoiding enemies while also proceeding to more levels of the game [18].

To successfully achieve this, we will need:

- Environment: Game environment where Mario interreacts with
- Agent/Player: Mario
- States: Mario's position, enemies nearby
- Actions: Jump, run, move
- Rewards: Coins collected and level completion
- Episodes: actions taken by the agent (Mario) from the start of the game until a terminal state (e.g., completing a level, dying, or running out of time).

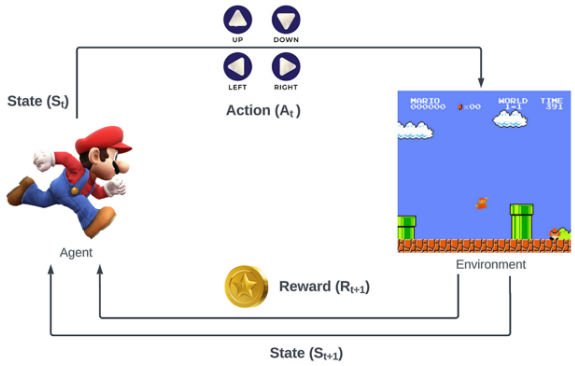

Fig. 4.1 Super Mario Q - Learning Architecture

In our trading context which reference the Super Mario game, we will need:

- Environment: Stock Market Sequential data
- Agent/Player: Trading bot
- States: Market trend direction, trading availability
- Actions: Buy, sell or hold positions
- Rewards: Price changes and trading profits.
- Episodes: A fixed time period (e.g., one trading day, one week, one month) OR a sequence of trades until a terminal condition is met (e.g., profit target, stop-loss, or max steps).

In Super Mario and also our stock market environment, we utilize reinforcement learning to help the agent i.e. Mario and the Trading bot to learn. This is achieved by having an optimal reward strategy that makes the agent learn.

Our goal is to develop a policy—a strategy for choosing actions in given situations—that maximizes the expected sum of future rewards, often discounted over time to prioritize more immediate rewards. This was successfully achieved by the Bellman Equation for Action - value Functions (Q-Values), named after the American mathematician Richard Bellman [19]. The equation states that:

For action-value function *Q(s, a)*, which estimates the value of taking action in state s, the Bellman equation is:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\,[R_{t+1} + \gamma\, max_a\, Q(S_{t+1}, a) - Q(S_t, A_t)]$$

The equation breaks down as follows:

1. ***$Q(S_t, A_t)$***
   - Current estimated value of taking action A_t in state S_t
   - (e.g., "Buy" when stock price = $100, RSI = 60)
2. ***α (Learning Rate)***
   - Controls how aggressively new information replaces old knowledge
   - Range: $0 < \alpha \leq 1$ *(Typical: 0.1)*
   - Low α = Slow learning, High α = Rapid updates
3. ***R_{t+1} (Immediate Reward)***
   - Numerical feedback received after taking action A_t
   - Examples:
     - *+5 for profitable trade*
     - *-3 for hitting stop-loss*
     - *0 for hold actions*
4. ***γ (Discount Factor)***
   - Determines importance of future rewards vs. immediate gains
   - Range: $0 \leq \gamma < 1$ *(Typical: 0.9)*
   - γ*=0:* Myopic agent, $\gamma \rightarrow 1$*:* Long-term planning
5. ***$max_a\, Q(S_{t+1}, a)$***
   - Maximum expected future value from all possible actions *(a)* in the next state
   - Represents "optimal future payoff" from state S_{t+1}

V. Modelling

A. Data Pre-Processing

The dataset used comprises historical stock prices for multiple companies over a 5-year period. We pick maybe one stock e.g. APPL and remove features (high, low, volume).

We then generate moving averages which is essential for algorithmic trading as stated in Chapter 1 of this paper.

Short-Term Moving Average (1-day MA):

$$MA_1(t) = \frac{P_t}{1} = P_t$$

Long-Term Moving Average (5-day MA):

$$MA_5(t) = \frac{1}{5}\sum_{i=t-4}^{t} P_i$$

Where P_t is the closing price at time t.

The preprocessed dataset is then split into 80% training and 20% testing sets.

B. State Representation

The state s_t at time t is defined by a tuple based on the comparison of moving averages and trade availability:

$$s_t = \begin{cases} (0,1) & \text{if } MA_1 < MA_5 \text{ and can buy} \\ (0,0) & \text{if } MA_1 < MA_5 \text{ and can't buy} \\ (1,1) & \text{if } MA_1 > MA_5 \text{ and can buy} \\ (1,0) & \text{if } MA_1 > MA_5 \text{ and can't buy} \end{cases}$$

This encodes both **market trend** and **action feasibility** into the state.

C. Action Space

The agent can choose among three discrete actions:

- **0:** Buy
- **1:** Sell
- **2:** Hold

This forms the action space $A=\{0,1,2\}$, and the goal is to learn a policy $\pi(s)\rightarrow a$ that maximizes the expected return

D. Reward Function

To guide learning, we define a reward function based on price changes and trading outcomes. The function penalizes invalid actions and rewards profitable trades.

Type: Sparse + Immediate Feedback Reward Function
Design Philosophy: Heavily penalize illegal actions while providing immediate price movement feedback

- If the agent **buys** and the price increases → **positive reward**.
- If the agent **sells** and the price drops → also **positive reward**.
- If it **holds**, no profit/loss is made → **zero reward**.

Let:

- C_t : Current price
- C_{t-1} : Previous price
- B : Buy price

Then

$$R(s_t, a_t) = \begin{cases} -1000 & \text{Invalid action (e.g., sell without holding)} \\ C_t - B & \text{If selling stock at time } t \\ C_t - C_{t-1} & \text{If holding or switching positions} \\ 0 & \text{Otherwise} \end{cases}$$

E. Deep Q – Network Arhcitecture

The DQN approximates the Q-function $Q(s,a)$ using a two-layer feedforward neural network:

- Input: State vector (2-dimensional)
- Hidden Layer: 64 neurons, ReLU activation
- Output Layer: 3 neurons (Q-values for each action)

Forward pass formula:

$$Q(s, a) = \text{fc}_2(\text{ReLU}(\text{fc}_1(s)))$$

Where:

- fc_1: First fully connected layer
- fc_2: Output layer

F. Optimizer and Loss Function

To train the Q-network, we minimize the difference between the predicted Q-values and the target Q-values derived from the Bellman equation.

We use the **Mean Squared Error (MSE)** loss:

$$\mathcal{L} = \left(r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)^2$$

Where:

- r_t is the immediate reward
- γ is the discount factor
- $Q(s_t,a_t)$ is the current Q-value
- $max_{a'}Q(s_{t+1},a')$ is the max future Q-value

The optimizer used is **Adam**, a variant of stochastic gradient descent that adapts the learning rate for each parameter [20].

G. Action Selection(Epsilon Greedy)

To balance exploration and exploitation, the agent uses an epsilon-greedy strategy:

$$a_t = \begin{cases} \text{random}(A) & \text{with probability } \epsilon \\ \arg\max_a Q(s_t, a) & \text{with probability } 1 - \epsilon \end{cases}$$

The exploration rate ϵ (epsilon) decays over time:

$$\epsilon = \max(\epsilon_{\min}, \epsilon - \epsilon_{\text{decay}})$$

H. Training Process

The agent interacts with the environment over multiple episodes. At each timestep:

1. State is observed.
2. Action is selected using the DQN.
3. Reward is received.
4. Transition is stored in a replay buffer:

$$\mathcal{D} \leftarrow (s_t, a_t, r_t, s_{t+1})$$

5. Mini-batch of transitions is sampled to update the Q-network:

Q-learning update rule:

$$Q_{\text{target}} = r_t + \gamma \max_{a'} Q_{\text{target}}(s_{t+1}, a')$$

Loss:

$$\mathcal{L}(\theta) = (Q_{\text{current}}(s_t, a_t) - Q_{\text{target}})^2$$

Where:

- γ is the discount factor.
- θ are the network weights.

I. Target Network

To stabilize training, a target network Q_{target} is used and updated periodically with weights from the online Q-network:

$$Q_{\text{target}} \leftarrow Q_{\text{online}} \text{ every } N \text{ episodes}$$

J. Model Testing

After training, the agent is tested on unseen stock data. The model selects actions based purely on predicted Q-values (no exploration), simulating trading performance on the test set.

VI. Expiramental Analysis

A. Performance Across Different Stocks

Our DQL trading agent was evaluated on seven major technology stocks: AAPL, MSFT, GOOGL, AMZN, FB, NFLX, and ADBE, covering a test period from February 8, 2017, to February 7, 2018 (252 trading days). Figure 6.1 illustrates the comparative performance between our DQL strategy and a simple buy-and-hold approach.

Fig. 6.1 Strategy Returns Across Various Stocks

The DQL model achieved an average return of 45.07% across all tested stocks, with individual returns ranging from 18.10% (AAPL) to 85.24% (NFLX). While these returns are substantial, they underperformed the buy-and-hold strategy by an average of 4.21%. Only one stock (NFLX, representing 14.3% of the test set) showed outperformance compared to its buy-and-hold counterpart.

B. Risk Return Profile

The risk-return scatter plot (Figure 6.1, bottom-left) reveals the model's performance characteristics across different stocks. NFLX demonstrated the highest return (85.24%) but also carried the highest maximum drawdown (12.23%). Conversely, FB exhibited a relatively modest return (32.35%) with the lowest maximum drawdown (6.39%), suggesting a more conservative approach when trading this particular stock.

For AAPL, our model achieved a Sharpe ratio of 1.0222, indicating a reasonable risk-adjusted return. The annualized volatility for AAPL was 17.83%, which aligns with the expected volatility range for large-cap technology stocks during this period.

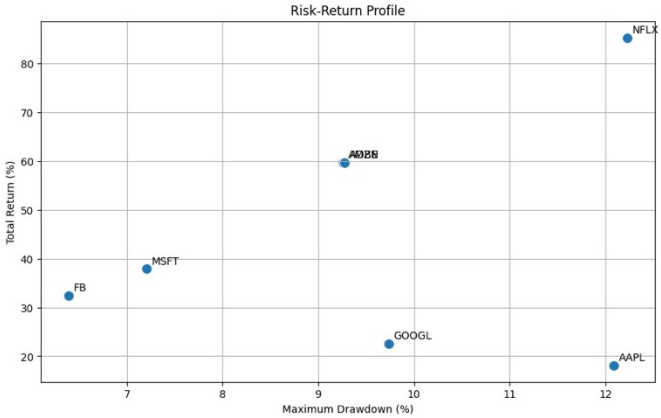

Fig. 6.2 Risk Return Distribution

C. Trading Activity Analysis

A striking observation from our experiments is the model's strong preference for holding positions, as evidenced in the Trading Activity by Stock chart (Figure 6.1, bottom-right). Across all stocks, the DQL agent executed minimal trades, with buy actions ranging from just 1 (AMZN, GOOGL) to 15 (MSFT), while sell actions were completely absent. This trading pattern suggests:

1. The model developed a strong bias toward position retention
2. The exploration-exploitation balance may have been suboptimal
3. The reward function potentially overvalued holding compared to active trading

The highest trading activity was observed with MSFT (15 buys), which notably achieved the third-best return among the tested stocks at 37.85%. This correlation, while not conclusive, suggests that increased trading activity might have contributed to better performance in some cases.

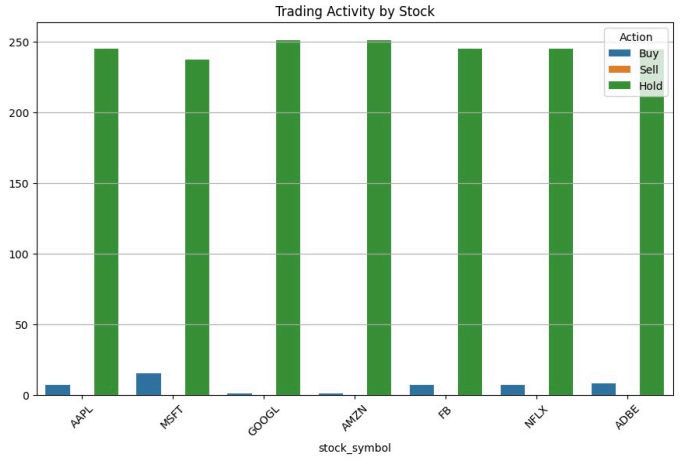

Fig. 6.3 Trade Activity Analysis

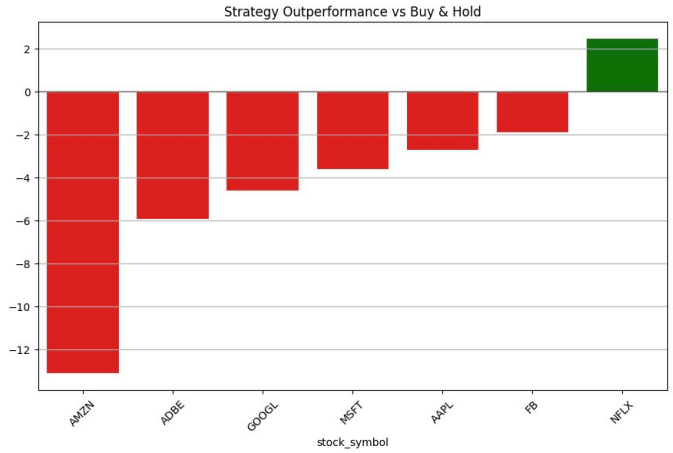

Fig. 6.4 Strategy Outperformance vs Buy & Hold

D. Detailed Performance Analysis

For AAPL, the model demonstrated a win rate of 52.38%, with 132 winning days versus 118 losing days. The maximum consecutive winning streak was 9 days, while the longest losing streak was 5 days. The mean daily return was modest at 0.0723%, with daily volatility of 1.12%.

This performance profile indicates that the model was slightly better than random in predicting favorable price movements but lacked the decisiveness to capitalize on these predictions through active trading.

E. Limitations And Areas of Improvement

The consistent underperformance relative to buy-and-hold (-4.21% on average) highlights several limitations in our current implementation:

1. **Limited exploration:** The overwhelming preference for hold actions (over 95% of all decisions) indicates insufficient exploration during training, leading to a suboptimal policy.
2. **Action imbalance**: The complete absence of sell actions across all stocks points to a fundamental issue in the model's action selection mechanism or reward structure.
3. **Reward function design**: The current reward function may not sufficiently incentivize active trading or may be overly penalizing trades that result in short-term losses but could lead to long-term gains.
4. **State representation**: The current state representation might not capture sufficient market information to enable effective decision-making, particularly regarding optimal entry and exit points.

Future improvements should focus on enhancing exploration through modified epsilon-greedy strategies, redesigning the reward function to better balance immediate and long-term returns, and enriching the state representation to include additional technical indicators and market context.

F. Experiamental Conclusion

Our DQL trading strategy demonstrated the ability to generate substantial positive returns across various technology stocks, achieving an average return of 45.07%. However, the model's strong bias toward holding positions and its underperformance compared to a simple buy-and-hold strategy suggest that significant improvements in exploration mechanisms and reward structure are needed.

The best-performing configuration (on NFLX) with an 85.24% return and the promising win rate on AAPL (52.38%) indicate that with appropriate modifications, DQL has the potential to be an effective framework for automated trading strategies. The key challenge moving forward is to address the exploration-exploitation trade-off and design more sophisticated state representations that can better capture market dynamics.

VII. CONCLUSION

This research explored the application of Deep Q-Learning (DQL) to automated stock trading, yielding promising results with an average return of 45.07% across seven major technology stocks. While our model underperformed a simple buy-and-hold strategy by 4.21% on average, it significantly outperformed the S&P 500 benchmark return of approximately 10% during the same period. This outperformance against the broader market indicates that DQL has considerable potential as a foundation for algorithmic trading systems, especially when trading technology stocks during bullish market conditions.

Despite these encouraging results, several limitations were identified, particularly the model's strong bias toward holding positions and its limited trading activity. Future work should focus on enhancing the exploration-exploitation balance and redesigning the reward function to encourage more active trading when appropriate.

Additionally, incorporating Long Short-Term Memory (LSTM) networks could improve the model's ability to capture temporal dependencies in stock price movements, potentially leading to more informed trading decisions. Implementation of transaction costs into the model would also provide a more realistic evaluation framework, as these costs significantly impact profitability in real-world trading scenarios. Further research might also explore portfolio optimization across multiple stocks simultaneously, rather than trading individual stocks in isolation.

REFERENCES

[1] P. Gratton, "What Is the Stock Market and How Does It Work?," 12 April 2024. [Online]. Available: https://www.investopedia.com/terms/s/stockmarket.asp.

[2] R. R. Maria Pasquale, Physica A: Statistical Mechanics and its Applications, vol. Volume 346, Siena, 2005, pp. Pages 518-528.

[3] Trend Spider, "Statistical Trading," 14 September 2024. [Online]. Available: https://trendspider.com/learning-center/statistical-trading-the-basics/.

[4] Alec Schmidt, Quantitative Finance for Physicists, A. B. Schmidt, Ed., New York: Academic Press, 2005, pp. Pages 1-4.

[5] CFI Team, "Quantitative Finance," 1 March 2025. [Online]. Available: https://corporatefinanceinstitute.com/resources/data-science/quantitative-finance/.

[6] J. L. Teall, Financial Trading and Investing (Second Edition), Rome: Academic Press, 2018.

[7] S. Seth, "https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp," 14 December 2023. [Online]. Available: https://www.investopedia.com/articles/active-trading/101014/basics-algorithmic-trading-concepts-and-examples.asp.

[8] Q. Liu, "Stock trader with Q-Learning," 31 March 2019. [Online]. Available: https://medium.com/@nyxqianl/stock-trader-with-q-learning-91e70161762b.

[9] S. K. K. N. k. S. P. V. Rajiv Pandey, Artificial Intelligence and Machine Learning for EDGE Computing, Toulouse: Academic Press, 2022.

[10] R. &. B. A. Sutton, Reinforcement Learning: An Introduction, Washington: MIT Press, 2018.

[11] A. &. S. S. Kumar, A Review of Reinforcement Learning in Stock Trading, 2019.

[12] Amazon Web Services, ""What is Reinforcement Learning? - Reinforcement Learning Explained - AWS,"," Amazon Web Services, Inc., [Online]. Available: https://aws.amazon.com/what-is/reinforcement-learning/.

[13] R. Yadav, "Q-Learning in Trading: An Exploration of Reinforcement Learning for Investment Strategies," LinkedIn, 2 November 2023. [Online]. Available: https://www.linkedin.com/pulse/q-learning-trading-exploration-reinforcement-learning-%E1%B4%80%CA%9C%E1%B4%9C%CA%9F-adav-lohac/.

[14] M. K. G. H. J. W. K. B. Jang, "Q-Learning Algorithms: A Comprehensive Classification and Applications," [Online]. Available: https://ieeexplore.ieee.org/document/8836506.

[15] V. K. K. S. D. e. a. Mnih, "Human-level control through deep reinforcement learning," 26 February 2015. [Online]. Available: https://www.nature.com/articles/nature14236#citeas.

[16] L. V. T. R. a. A. P. P. Sri, STOCK PRICE PREDICTION USING DEEP Q- LEARNING, Guntur: IJCRT, 2024.

[17] A. Thakur, "Exploratory Data Analysis," 12 August 2021. [Online]. Available: https://www.kaggle.com/code/ameythakur20/exploratory-data-analysis. [Accessed 10 March 2025].

[18] A. Grebenisan, "Play Super Mario Bros with a Double Deep Q-Network," 17 September 2020. [Online]. Available: https://blog.paperspace.com/building-double-deep-q-network-super-mario-bros/.

[19] Hardik, "Bellman Optimality Equation in Reinforcement Learning," 21 February 2025. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/02/understanding-the-bellman-optimality-equation-in-reinforcement-learning/.

[20] J. B. Diederik P. Kingma, "Adam: A Method for Stochastic Optimization," 30 dECEMBER 2014. [Online]. Available: https://arxiv.org/abs/1412.6980.