INVESTIGATING REINFORCEMENT LEARNING APPROACHES IN STOCK MARKET TRADING

Davies Luo
School of Mathematics, Statistics and Physics
Newcastle University
Newcastle upon Tyne, UK
`z.luo24@newcastle.ac.uk`

August 2024

ABSTRACT

The financial industry and stock market analysis have been significantly reshaped by digital advancements, fostering remarkable growth and innovation. This transformation has been accelerated by the rise of artificial intelligence (AI) and the increasing demand for sophisticated computational power across various industries. A prime example of this shift is the rapid growth of semiconductor giants like Advanced Micro Devices (AMD) and Nvidia Corporation, both of which have seen notable increases in stock prices and market share. This project investigates the application of Reinforcement Learning (RL) in automating investment strategies within financial markets. By focusing on AMD and Nvidia, the study explores the implementation of RL algorithms, including policy networks and deep Q-networks, to optimise stock trading and investment decisions. The project aims to identify robust RL approaches that enhance decision-making by maximising returns while effectively managing market complexities and volatility, through simulations of real-world scenarios and comprehensive market data.

1 Introduction

Reinforcement Learning (RL) is a type of machine learning in which an agent learns to make optimal decisions by interacting with its environment to maximise cumulative rewards. This method has achieved success in various fields. For example, Boston Dynamics utilises RL to enhance the agility and robustness of its robots. In their latest model, Spot, RL is employed to perform complex tasks while maintaining balance, highlighting how RL enables machines to learn and adapt to dynamic environments [6]. Similarly, Google DeepMind has applied RL to optimise the cooling systems in Google's data centers. By training an RL model to predict and improve cooling efficiency, they achieved a 40% reduction in energy consumption for cooling, demonstrating the practical utility of RL in industrial settings and its significant environmental benefits [21].

Financial markets are characterised by their complexity and volatility, influenced by economic indicators, geopolitical events, and investor sentiment [2, 36]. Traditional trading methods are limited by human cognitive and operational factors, making them less effective in rapidly changing market conditions. Reinforcement Learning (RL), with its ability to learn and adapt through continuous interaction with the environment, offers a promising solution for developing automated trading strategies that can respond to market fluctuations and optimise investment outcomes.

The semiconductor industry, the critical sector driving the third industrial revolution [44], underpins modern technology and plays a vital role in powering everything from consumer electronics to advanced computing systems. Therefore, applying RL to stock trading in the semiconductor industry, particularly with companies such as AMD and Nvidia, offers a compelling opportunity for analysis. As leaders in GPU technology and AI innovation, these companies experience significant market movements that are reflective of broader industry trends [4]. Their dynamic stock behaviour, driven by technological influence, provides a rich environment for testing RL's effectiveness in trading strategies [41]. The primary aim of this work is to explore how RL can automate stock trading processes for retail investors, potentially outperforming traditional methods and delivering consistent returns even in volatile markets [30].

The stock market is a dynamic and complex environment where conditions change rapidly. Reinforcement Learning (RL) is particularly well-suited for navigating this environment because:

- **Adaptability**: RL can adapt to changing market conditions by continuously learning from interactions.
- **Sequential Decision Making**: RL naturally handles the sequential nature of trading decisions, optimising long-term returns rather than immediate gains.
- **Exploration and Exploitation**: RL balances exploration (trying new strategies) and exploitation (using known strategies) to find the optimal policy [14].

One significant challenge faced is data availability. In real-world automated trading applications, algorithms often trade at very high frequencies, with such data typically accessible only through online brokers like Bloomberg [5]. With some code modifications, we can enhance the model and obtain more insightful results using this data. This project explores various Reinforcement Learning techniques, including Policy Networks, Deep Q-learning, and Long Short-Term Memory (LSTM) Networks. Additionally, we examine several forecasting methods by applying them in the real market and analyse the predictions and returns from these models.

This research makes several important contributions to the field of AI and financial market trading:

1. **Advancement of RL Techniques:** By implementing and comparing various RL algorithms, this study advances the understanding of how RL can be effectively applied to stock market trading.
2. **Integration of Comprehensive Indicators:** The incorporation of diverse financial, economic, and social indicators provides a holistic approach to trading strategy development and helps capture a wide range of market influences.
3. **Practical Implications for Investors:** The findings offer valuable insights for retail investors and financial institutions, demonstrating the potential of RL to enhance trading performance and decision-making.
4. **Foundation for Future Research:** This study lays the groundwork for future research in RL and algorithmic trading, highlighting key areas for further exploration and development.

In summary, this paper explores the application of RL to automate stock trading strategies, with a focus on the semiconductor industry. By leveraging advanced RL algorithms and integrating comprehensive market data, this research aims to develop robust trading models that can navigate the complexities of financial markets and deliver superior investment outcomes. The insights gained from this study have significant implications for the future of AI-driven trading and provide a foundation for ongoing research and innovation in this field.

This paper is structured as follows:

- **Section 2: Background:** This section offers an in-depth overview of the essential concepts needed to understand the financial investment market and modern reinforcement learning (RL) algorithms.
- **Section 3: Methodology and Analysis:** This section presents our RL methodologies and training processes.
- **Section 4: Results:** This section describes the performance of various algorithms on real stock market data.
- **Section 5: Conclusion:** This section highlights the key findings of our research and suggests possible areas for future exploration.

All code used to produce the results of this study is available on GitHub[1]. The data supporting this study can be accessed through the NCL Research Repository[2].

2 Background

In this section, we explore the fundamental concepts and methodologies that underpin the application of Reinforcement Learning (RL) in stock market analysis, particularly focusing on the semiconductor industry. We begin by providing a detailed mathematical explanation of RL and its suitability for stock market price forecasting. We then delve into the basics of financial market investing, key economic figures influencing the stock market and the impact of media and investor sentiment. Throughout, we discuss how these elements integrate with RL models to optimise investment strategies.

[1]`www.github.com/daviesluo/Investigating-Reinforcement-Learning-Approaches-in-Stock-Market-Trading`

[2]Repository: Data for Investigating Reinforcement Learning Approaches in Stock Market Trading. `data.ncl.ac.uk`.

2.1 Reinforcement Learning

The stock market is a dynamic and complex environment influenced by various factors, including financial indexes, economic indicators, and investor sentiment. Reinforcement Learning (RL) has emerged as a powerful tool for developing trading strategies by allowing agents to learn and adapt to these changing market conditions. This section provides an overview of the key concepts and methodologies used in this paper, including the fundamentals of RL, policy gradient methods, Q-learning with Proximal Policy Optimisation (PPO), and Actor-Critic models. We also discuss the integration of advanced time-series models and the impact of economic and investor sentiment data on stock market investing.

The process is modelled mathematically using Markov Decision Processes (MDPs) [46]. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$:

- $\mathcal{S}$: a set of states, called the *state space*.
- $\mathcal{A}$: a set of actions, called the *action space*.
- $P(s'|s,a)$: the transition probability from state s to state s' given action a.
- $R(s'|s,a)$: the reward received after transitioning from state s to s' when taking action a.
- γ: the discount factor, $0 < \gamma < 1$, which prioritises immediate rewards over future rewards.

The objective is to find an optimal policy π^* that maximises the expected cumulative reward:

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1} \mid s_t, a_t) \middle| s_0 = s\right].$$

Here, $V^\pi(s)$ is called the *state-value function*. The Bellman equation provides a recursive way to solve for $V^\pi(s)$:

$$V^\pi(s) = \mathbb{E}[R(s' \mid s, a) + \gamma V^\pi(s') \mid s].$$

This equation breaks down the value of a state into the immediate expected reward plus the discounted value of the subsequent state, facilitating the computation of the optimal policy by iteratively improving the value estimates [16].

2.1.1 Q-Learning

Q-Learning [49] is a value-based method where the Q-function, also known as the *action-value function*,

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1} \mid s_t, a_t) \middle| s_0 = s, a_0 = a\right]$$

is learned. The goal is to learn the optimal action-value function $Q^*(s,a)$ which satisfies the Bellman equation:

$$Q^*(s,a) = \mathbb{E}\left[R(s'|s,a) + \gamma \max_{a'} Q^*(s',a') \mid s, a\right].$$

Q-Learning is effective for discrete action spaces and can handle a wide variety of environments.

2.1.2 Policy Gradient Methods

Policy Gradient methods optimise the policy directly by adjusting the parameters of the policy π_θ. The objective is to maximise the expected return $J(\theta)$:

$$J(\theta) = \mathbb{E}_s[V^\pi(s)]. \tag{1}$$

The Policy Gradient Theorem [43] provides a way to compute the gradient of $J(\theta)$:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_{t+1}|s_t, a_t) \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

Policy Gradient methods are particularly useful for high-dimensional action spaces and continuous action spaces.

Policy Gradient with Baseline: Policy Gradient with Baseline methods [50] enhance standard policy gradient methods by introducing a baseline function $b(s_t)$, which reduces the variance of the gradient estimates. The baseline is often chosen as the value function $V^\pi(s_t)$. The modified gradient of $J(\theta)$ then becomes:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t (R(s_{t+1}|s_t, a_t) - b(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)\right].$$

By subtracting the baseline, the learning process becomes more stable and efficient, whilst still ensuring unbiased estimates of the gradient.

Actor-Critic Models: Actor-Critic methods [28] combine the benefits of both policy-based and value-based methods. The actor updates the policy parameters θ in the direction suggested by the critic, which evaluates the action taken by the actor using the value function V^π. The advantage function $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ helps to improve the stability and efficiency of the learning process:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a|s) A^\pi(s, a)\right].$$

Actor-Critic models can handle both discrete and continuous action spaces, making them flexible for various stock market scenarios. They allow for more stable training by using the critic to reduce the variance of the updates.

2.2 Financial Market and Stock Market Investing

The financial market involves buying and selling securities, and stock market investing focuses on trading shares of public companies. This section will cover several crucial metrics and indices that play significant roles in stock price analysis and reinforcement learning models: OHLCV data, Number of Shares Outstanding, and key stock indices such as S&P 500, NASDAQ-100, and PHLX Semiconductor.

- **OHLCV Data**: OHLCV stands for Open, High, Low, Close, and Volume, which are key components of stock market data [31]. Open Price (O) is the price at the start of a trading session, High Price (H) is the highest price during the session, Low Price (L) is the lowest price during the session, Close Price (C) is the price at the end of the session, and Volume (V) is the number of shares traded. OHLCV data provides crucial insights into market behavior and is fundamental for technical analysis, helping to identify price trends, volatility, and trading volume for informed trading decisions.
- **Number of Shares Outstanding (S.O.)**: This refers to the total number of a company's shares that are currently owned by all its shareholders, including large holdings by institutional investors and restricted shares held by company executives and insiders. [35].

$$\text{Number of Shares Outstanding} = \text{Issued Shares} - \text{Treasury Shares}$$

  - **Stock Splits Adjustment**: The number of shares outstanding is critical for adjusting historical stock prices to reflect stock splits. Stock splits can significantly impact the stock price by increasing the number of shares outstanding, thus lowering the price per share without changing the company's market capitalisation [32]. This adjustment ensures that the total return of a stock includes the effects of stock splits, providing a more accurate measure of performance.

$$\text{Adjusted Price} = \frac{\text{Previous Closing Price}}{\text{Stock Split Ratio}}$$

- **S&P 500**: This index tracks the stock performance of 500 large companies listed on US stock exchanges and is widely considered one of the most accurate reflections of the overall US stock market and economy. Movements in the S&P 500 can influence the stock prices of companies within it, including AMD and Nvidia, which are both key players in the technology sector [18].
- **NASDAQ-100**: This index consists of 100 of the largest non-financial companies listed on the NASDAQ stock exchange, with a strong emphasis on technology companies, making it particularly relevant for assessing the performance and market sentiment of tech stocks like AMD and Nvidia [23].
- **PHLX Semiconductor**: The PHLX Semiconductor Sector Index (SOX) is a modified market capitalisation-weighted index composed of companies primarily involved in the design, distribution, manufacture, and sale of semiconductors. Given that AMD and Nvidia are major players in the semiconductor industry, this index serves as a vital indicator of industry trends and performance[29].

Understanding these indices and metrics is essential for RL models as they provide a macroeconomic context that influences the stock prices of AMD and Nvidia. Data is sourced from financial databases Yahoo Finance and Bloomberg, then processed using Python's Pandas library for accurate calculations and adjustments.

2.3 General Economic Influence

General economic factors can significantly impact the stock market [17]. Key indicators include:

- **Inflation Rate**: The rate at which the prices of goods and services increase, impacting the value of money and economic stability. Higher inflation often results in rising costs for businesses and decreased consumer spending.
- **Federal Reserve Interest Rate**: The interest rate that banks use for overnight loans between each other, set by the Federal Reserve. This rate influences borrowing costs for businesses and consumers, thereby impacting economic activity and stock market[45].
- **Effective Federal Fund Rate**: The interest rate banks charge each other for overnight lending of excess reserves. This rate affects the overall cost of borrowing and liquidity in the financial system[8].
- **Consumer Confidence Index**: An indicator of how optimistic or pessimistic consumers feel about the economy's future. When confidence is high, consumer spending tends to increase, which can spur economic growth and lift stock prices, and vice versa.
- **Oil Prices**: The price of crude oil affects the cost of energy and transportation. Fluctuations in oil prices can impact the profitability of companies dependent on energy, as well as the overall economy[38].
- **Gold Prices**: Gold is often seen as a safe-haven asset. Changes in gold prices can reflect investor sentiment towards economic stability and inflation expectations [3].

These indicators are incorporated into RL models to forecast stock prices by adjusting for economic conditions. Data is sourced from economic reports and databases such as the Federal Reserve and World Bank.

2.4 Media Influence and Investor Sentiment

Media and investor sentiment significantly impact stock market dynamics, where positive or negative news can drive stock prices up or down. Thus, we utilise Google Trends data [20] to measure investor sentiment by analysing search volumes for 'AMD' and 'Nvidia'.

- **Google Trends Score**: A numerical representation of the search interest over time for specific keywords, serving as a proxy for investor sentiment [34].

Google Trends data offers valuable insights into public interest and sentiment towards selected companies. By integrating this data, we aim to enhance the predictive power of our models, providing a more comprehensive understanding of market behaviour.

2.5 Previous Work

Previous research in stock trading with reinforcement learning (RL) has focused on various approaches but has faced certain limitations. Dasgupta [10] used Deep Q-Learning (DQL) with Convolutional Neural Networks (CNN) to incorporate social media sentiment, yet this approach may overlook complex temporal dependencies. Xiong et al. [51] employed Deep Deterministic Policy Gradient (DDPG) for continuous action spaces in stock trading, which can suffer from instability and require extensive tuning. Kim and Kim [26] combined LSTM networks and CNNs to improve predictions but did not fully leverage economic indicators . Our work can be seen as an extension of these previous works, since, as discussed in Section 3.3.3, we combine time-series models with comprehensive financial, economic, and sentiment indicators. We demonstrate in Section 4 that this holistic approach achieves impressive predictive accuracy and trading performance.

3 Methodology and Analysis

This section outlines the methodologies employed to develop, implement, and evaluate the reinforcement learning (RL) models used for stock market trading, specifically focusing on the semiconductor industry with case studies of AMD and Nvidia. We detail the data preparation, environment setup, RL algorithms implementation, and the evaluation metrics used to analyse the performance of these models.

3.1 Data Preparation

The dataset used in this project comprises various sources of financial, economic, and social data, including OHLCV data, number of shares outstanding, and key economic indicators. The steps involved in data preparation are as follows:

- **Data Collection:** Data was sourced from reliable financial databases such as Yahoo Finance, Google Trends, and economic reports from the Federal Reserve. The data included minute intervals of stock OHLCV data, daily intervals of financial and economic indicators, weekly intervals of Google Trends data, and quarterly intervals of the number of shares outstanding.
- **Data Normalisation:** All longer-period values were resampled into minute intervals, where each minute within the original period was assigned the same value as that of the corresponding longer period. This ensures consistency in the dataset. Additionally, gaps in the stock price data caused by market closures, such as weekends and public holidays, were filled with the last available price to maintain continuity in the data. However, this method of filling gaps may limit the model's ability to detect sudden market shifts or significant events that occur during these periods.
- **Data Alignment:** All datasets were aligned based on each minute to ensure that each record corresponds to the same time period across different data sources. This involved merging datasets on the date field to create a unified dataframe.
- **Final Dataset Creation:** The final dataset was created by combining all the processed features into a single dataframe. This dataframe includes columns for OHLCV data after stock split adjustments, financial indexes, economic indicators and Google Trends scores.

3.2 Reinforcement Learning Environment

The trading environment used in this project was developed with OpenAI Gymnasium, a toolkit designed for creating and benchmarking reinforcement learning algorithms and environments[53]. This environment is customised to replicate real-world trading scenarios, providing a controlled setting where RL algorithms can be trained and evaluated under the intricate conditions of financial markets. The environment's key components include:

- **Action Space:** Continuous vectors representing decisions to buy, sell, or hold assets, which give the agent detailed control over trading actions and allow for the adjustment of asset proportions at each step.
- **Observation Space:** A multi-dimensional array that includes OHLCV data, economic indicators, and investor sentiment metrics. This comprehensive market information equips the RL agent to make informed trading decisions [24].
- **Initial Balance:** The agent begins with a starting balance of $1,000,000, simulating real-world trading conditions, and providing a realistic benchmark for evaluating trading strategies.

The methodology for implementing the trading environment encompasses several essential methods:

- **Reset Method:** Resets the environment to its initial state, establishing the initial balance, the number of shares held, and other key parameters, ensuring consistency for reproducible experiments [33].
- **Get State Method:** Gathers the current state of the environment, including all relevant financial indicators and stock market data, giving the RL agent a full overview of market conditions [42].
- **Step Method:** Executes the action selected by the agent, updates the environment's state, and calculates the reward based on changes in the agent's net worth. The reward structure encourages maximising net worth, with the reward function defined as:
$$r_t = \log\left(\frac{\text{NetWorth}_t}{\text{NetWorth}_{t-1}}\right)$$
where NetWorth_t is the agent's net worth at time t. This logarithmic return captures the multiplicative nature of returns and stabilises training by penalising large fluctuations in net worth [11].

In this study, we defined two types of trading agents to evaluate the performance of various reinforcement learning models: Simple Trader and Complex Trader. Each type of agent operates under a specific set of rules that govern their trading actions, providing a structured framework for comparison.

- **Simple Trader:** The Simple Trader agent follows basic and predefined trading rules aimed at providing a benchmark for evaluating the performance of more sophisticated reinforcement learning (RL) models. This agent can perform only three possible actions at any given time:

  - **Buying:** Purchase 5% of the current balance in shares of the stock.
  - **Selling:** Sell 5% of the shares currently held.
  - **Holding:** Take no action.

  These simple actions ensure that the agent operates within a constrained and easy-to-understand framework, making it suitable for initial testing and comparison against more complex models. The simplicity of this trader serves as a baseline to highlight the benefits and improvements brought by advanced RL techniques.

- **Complex Trader:** The Complex Trader agent employs advanced RL algorithms to dynamically adjust its trading strategy based on continuous action values, enabling more sophisticated decision-making processes. The agent's actions are determined by the output of the RL model, which can take any value within the range of -1 to 1:
  - **Positive Action Value ($a \in (0, 1]$):** Indicates a decision to buy a proportion of the balance in shares, with the proportion directly related to the action value (e.g., an action value of 0.5 means buying 50% of the balance in shares).
  - **Negative Action Value ($a \in [-1, 0)$):** Indicates a decision to sell a proportion of the currently held shares, with the proportion directly related to the action value (e.g., an action value of -0.3 means selling 30% of the shares held).
  - **Action Value of ($a = 0$):** Indicates a decision to hold, taking no action.

  This continuous action space allows the Complex Trader to fine-tune its trading strategy, responding to market conditions with greater precision and flexibility. The goal is to optimise trading performance by making informed decisions that balance risk and reward.

The RL agent can be trained by:

1. **Simulating an Episode:** At each state, the RL agent determines an action to take based on its policy. The environment then transitions to a new state according to the action taken, following the Markov Decision Process. This cycle continues for a specified number of time steps, comprising an episode [47].
2. **Updating the Parameters:** After simulating an episode, the objective function $J(\theta)$ is computed as the expected cumulative reward. For policy networks, the objective function can be seen in (1). The parameters θ of the RL agent are then updated by stochastic optimisation of $J(\theta)$ [48].

3.3 Reinforcement Learning Algorithms

Several reinforcement learning (RL) algorithms were implemented and trained to develop robust trading strategies, each with unique structures and training methodologies:

3.3.1 Deep Q-learning

The initial phase of our experiment involved training a reinforcement learning model in the Simple Trader environment, which provides a straightforward setup to test our model's capabilities. For this, we chose the Proximal Policy Optimisation (PPO) algorithm [40], a widely-used RL algorithm known for its robustness and effectiveness in continuous action spaces. PPO is particularly advantageous due to its ability to maintain a balance between exploration and exploitation, ensuring stable and efficient learning. Additionally, we utilised the `stable_baselines3` library [37], which provides a reliable and well-documented implementation of PPO, facilitating ease of experimentation and reproducibility. By leveraging PPO and `stable_baselines3`, we aimed to achieve optimal performance in training our Q-learning model.

Prior to training, we conducted a grid search to identify the optimal combination of hyperparameters, focusing specifically on the number of steps, batch size, and learning rate [39]. This systematic search was aimed at maximising the average reward across multiple training runs. We trained the model over 100 episodes, each consisting of 100 steps, and evaluated the performance based on the average reward. The grid search allowed us to identify the best-performing set of hyperparameters [13]. The best model parameters were then used to train the model for 20,000 episodes, each consisting of 100 steps. For each episode, we randomly sampled AMD and Nvidia stock data to create diverse and realistic training scenarios.

3.3.2 Policy Gradient, Baseline and Actor-Critic

The next phase of our experiment involved training reinforcement learning models using the Complex Trader environment, which provides a more sophisticated setup for testing our models' capabilities. We explored three approaches: Policy Gradient, Policy Gradient with Baseline, and Actor-Critic with Epsilon-Greedy strategy.

The rationale behind selecting these three methods lies in their unique approaches to policy optimisation. The Policy Gradient method directly optimises the policy, making it particularly suitable for environments with continuous action spaces. The Policy Gradient with Baseline method incorporates a value network to reduce variance in policy updates, potentially leading to more stable learning. The Actor-Critic with Epsilon-Greedy strategy combines value function approximation with policy optimisation while employing an exploration strategy to enhance learning.

Prior to training, we conducted hyperparameter tuning to identify the optimal combination of learning rate, discount factor, and exploration parameters for each model [27, 54]. The best model parameters were then used to train each model for 20,000 episodes, each consisting of 100 steps. For each episode, we randomly sampled AMD and Nvidia stock data to create diverse and realistic training scenarios. An epsilon-greedy strategy with decaying epsilon was employed for the Actor-Critic model to balance exploration and exploitation during the training phase [25].

3.3.3 Time Series Incorporation with ARIMA and LSTM

To enhance the performance of our Policy Gradient with Baseline model, we incorporated time series forecasting techniques: ARIMA [7] and LSTM [22]. ARIMA (Autoregressive Integrated Moving Average) is well-known for its capability to capture temporal dependencies and trends in time series data, making it suitable for predicting stock prices based on historical values. LSTM (Long Short-Term Memory) networks, on the other hand, are a type of recurrent neural network specifically designed to capture long-term dependencies and complex patterns in sequential data, which can be crucial for understanding market dynamics. By leveraging these models, we aimed to improve the trading agent's predictive accuracy and decision-making capabilities, thereby enhancing overall trading performance [12, 19]. The ARIMA model was trained to predict close prices based on historical close prices, while the LSTM model was trained to predict close prices based on all available features.

The ARIMA model was chosen for its effectiveness in capturing temporal dependencies within the data. To train the ARIMA model, we first differenced AMD and Nvidia's stock price time series to ensure stationarity, confirmed using the Augmented Dickey-Fuller test. Then ACF (Autocorrelation Function) and PACF (Partial Autocorrelation Function) plots were used to determine appropriate values for the ARIMA parameters p (Autoregressive order) and q (moving average order) [52]. Ultimately, the ARIMA model was trained with orders (2, 1, 2) for AMD and (3, 1, 3) for Nvidia.

For the LSTM (Long Short-Term Memory) model, we utilised its ability to capture long-term dependencies and complex patterns in the data. We prepared the data by creating sequences of length 5, using all available features as inputs [1]. The LSTM model was then trained for 100 epochs with a batch size of 256, using the Adam optimiser.

Both models were integrated into the StockData class to enhance the agent's decision-making by incorporating their predictions into the state representation. Due to the computational demands of statistical models, the ARIMA-enhanced model was trained for 5,000 episodes, while the LSTM-enhanced model, with its deep learning architecture, was trained for 20,000 episodes. This combined approach provided richer information for the agent, improving its performance in complex trading scenarios.

4 Results

4.1 Financial Evaluation for Reinforcement Learning Algorithms

The trading performance of the RL models was evaluated using several standard investment metrics to ensure a robust and comprehensive evaluation [15, 9]. These metrics include:

- **Cumulative Return:** Cumulative return measures the total return generated by the trading strategy over the evaluation period. It is calculated as:

$$\text{Cumulative Return} = \frac{\text{NetWorth}_{end}}{\text{NetWorth}_{start}} - 1$$

  This metric is essential as it directly reflects the overall profitability. In the context of stock market trading, a higher cumulative return indicates a more successful strategy in terms of capital appreciation. This metric is straightforward and provides a clear picture of the absolute gain or loss over the period.

- **Sharpe Ratio:** The Sharpe ratio measures how well a trading strategy compensates for the risk it takes by comparing the expected return above the risk-free rate to the variability of those returns:

$$\text{Sharpe Ratio} = \frac{\mathbb{E}[R_p - R_f]}{\sigma_p}$$

where R_p represents the portfolio return, R_f is the risk-free rate, and σ_p denotes the standard deviation of the portfolio's returns. This ratio is essential for evaluating how efficiently a strategy manages risk to achieve returns. In finance, higher volatility usually implies higher risk. The Sharpe ratio helps investors identify strategies that offer better returns for a given level of risk, making it a valuable metric for comparing different strategies on a risk-adjusted basis.

- **Maximum Drawdown:** Maximum drawdown measures the greatest drop from the highest point to the lowest point in a strategy's net worth, indicating the extent of potential losses and the associated downside risk:

$$\text{Maximum Drawdown} = \frac{\max_{t\in[0,T]}(\text{NetWorth}_{peak} - \text{NetWorth}_{trough})}{\text{NetWorth}_{peak}}$$

  This metric is particularly important for risk management as it highlights the worst possible loss an investor could experience from a peak in net worth to a subsequent trough. In stock market trading, large drawdowns can be detrimental, leading to significant capital erosion. Maximum Drawdown helps investors understand the potential for extreme losses, thus helping assess the risk profile of a trading strategy.

- **Hit Ratio:** Hit ratio measures the proportion of profitable trades to the total number of trades, reflecting the consistency of the trading strategy:

$$\text{Hit Ratio} = \frac{\text{Number of Profitable Trades}}{\text{Total Number of Trades}}$$

  This metric is vital for evaluating the consistency and reliability of a trading strategy. A higher Hit ratio indicates that the strategy generates profitable trades more frequently, suggesting a higher likelihood of sustained performance over time. In the volatile environment of stock trading, a strategy with a consistent Hit ratio is more desirable as it implies regular positive outcomes rather than occasional large gains interspersed with frequent losses.

These four metrics provide a comprehensive view of a trading strategy's performance. Cumulative return and Sharpe ratio together assess both the total profitability and risk-adjusted efficiency. Maximum Drawdown offers insight into potential risks and the worst-case scenarios, while Hit ratio gauges the strategy's consistency and reliability. Together, they form a robust framework for evaluating and comparing the effectiveness of different reinforcement learning models in stock market trading.

4.2 Trading Performance Comparison

Table 1 summarises the performance metrics of reinforcement learning algorithms across different environments:

RL Algorithms	**Environment**	**Return (%)**	**Sharpe Ratio**	**Maximum Drawdown ($)**	**Hit Ratio**
Deep Q-learning	Simple Trader	734.60	4.64	3,250,999.00	0.60
Policy Network	Complex Trader	585.17	4.22	3,009,615.92	0.59
Policy Network with Baseline	Complex Trader	1529.99	5.06	6,097,173.36	0.63
Actor-Critic with Epsilon-Greedy	Complex Trader	916.00	4.17	4,219,444.31	0.57
ARIMA in Policy Network with Baseline	Complex Trader	885.61	4.25	5,093,333.04	0.52
LSTM in Policy Network with Baseline	Complex Trader	2926.10	5.57	7,044,308.21	0.69

Table 1: Performance Comparison of Various RL Algorithms

The LSTM in Policy Network with Baseline model achieved the highest return of 2926.10% and the highest Sharpe ratio of 5.57, showcasing its superior performance. The deep learning architecture of LSTM effectively captured complex patterns, leading to more accurate predictions and informed trading decisions. This resulted in higher returns and a better return-to-risk ratio compared to other models, despite having the highest Maximum Drawdown. The Hit ratio of 0.69 further reflects its consistency in performance.

The Policy Network with Baseline model performed the best among the non-time-series models, achieving a return of 1529.99% and a Sharpe ratio of 5.06. The inclusion of a value network stabilised policy updates, enhancing performance. The ARIMA in Policy Network with Baseline provided minimal improvement compared to the original, but it was less effective than the LSTM model. The simpler methods, such as Deep Q-learning and standard Policy Network, showed lower returns and Sharpe ratios, demonstrating the advantages of sophisticated models and advanced prediction techniques.

4.2.1 Performance of Deep Q-learning

As summarised in Table 1, starting with an initial net worth of $1,000,000, the model achieved a final net worth of $14,388,227, yielding an impressive return of 1338.82%. The Sharpe ratio of 4.64 indicates a favourable return-to-risk balance, while the Maximum Drawdown of $3,250,999 reflects the worst observed loss from a peak to a trough. Additionally, a Hit ratio of 0.60 demonstrates consistent performance.

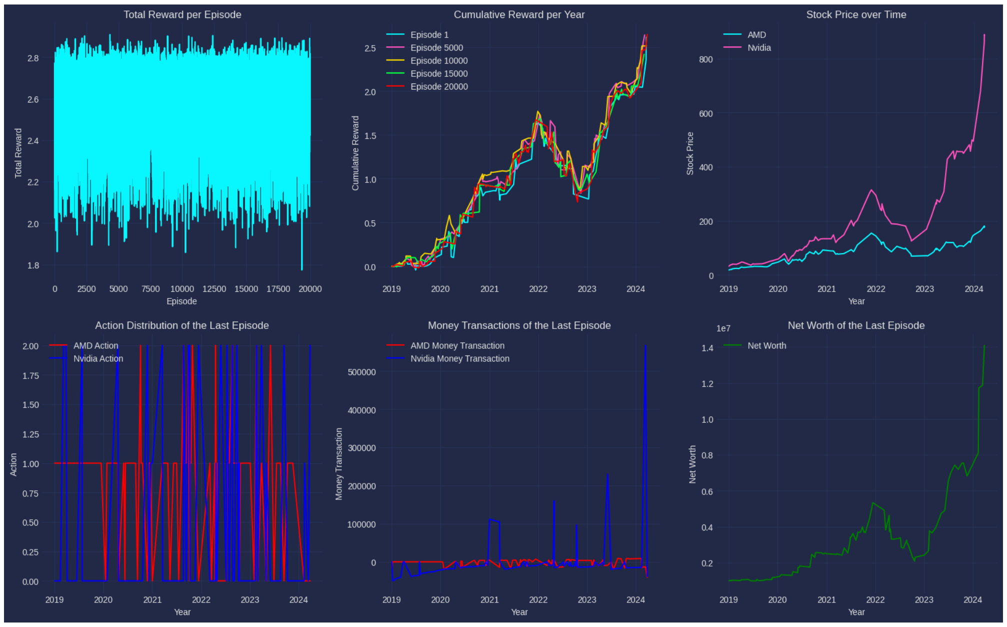

Figure 1: Deep Q-learning Training Performance

As displayed in Figure 1, the total reward per episode shows relatively stable performance with minor fluctuations, indicating that the simple model does not significantly improve rewards through additional training. The cumulative reward per year plot supports this, with minimal changes observed between distant episodes. The stock price over time plot displays the AMD and Nvidia stock prices throughout the training period, providing context for the trading decisions. The action distribution of the last episode plot reveals a strategic approach to trading, indicating the model's learning to take actions at optimal points. The money transactions plot aligns these actions with profitable opportunities. Although the simple trading rules limit the model's performance, they still set a solid foundation by achieving significant returns.

4.2.2 Performance of Policy Gradient, Baseline and Actor-Critic

As shown in Table 1, the Policy Gradient with Baseline method achieved the highest return of 1529.99% and the highest Sharpe ratio of 5.06, indicating a superior return-to-risk balance and overall performance. However, it also had the highest Maximum Drawdown of $6,097,173.36, suggesting significant fluctuations in net worth. The Policy Gradient model, while providing stable results, showed the lowest return and Sharpe ratio, suggesting that while the model maintained consistency, it lacked the ability to maximise gains effectively. The Actor-Critic model performed moderately well with a return of 916.00% and a Sharpe ratio of 4.17 but had a higher Maximum Drawdown compared to the Policy Gradient model, indicating more significant fluctuations in net worth, reflecting the model's aggressive exploration strategy. The Hit ratios reflect consistent performance across all models, with the Policy Gradient with Baseline showing a slight edge in consistency.

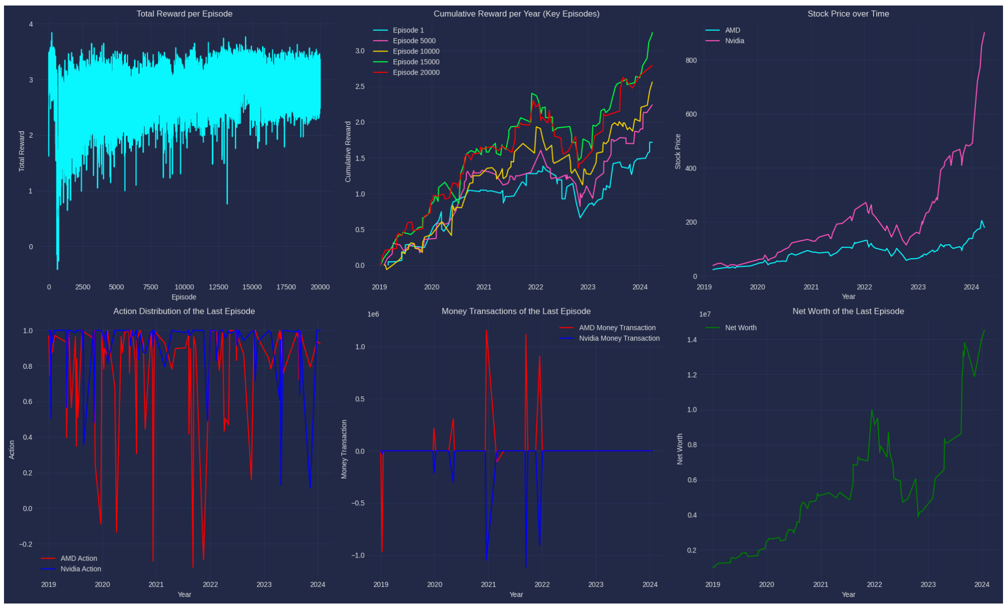

Figure 2: Policy Gradient with Baseline Training Performance

As depicted in Figure 2, the Policy Gradient with Baseline model demonstrated superior performance. The total reward per episode plot shows improved stability, with fewer large fluctuations, indicating a more stable learning process. The cumulative reward per year plot highlights substantial performance gains, particularly in later episodes, demonstrating the model's ability to learn and adapt effectively. The action distribution plot for the last episode reveals refined trading actions, suggesting that the model has learned to optimise trades more effectively, aligning with the money transactions and significant increase in net worth, reflecting the model's superior performance.

The superior performance of the Policy Gradient with Baseline model can be attributed to the inclusion of a value network, which provided better estimates of future rewards and reduced variance in policy updates. This led to more stable and effective learning, enabling the model to make more informed and profitable trading decisions. The combination of policy and value networks allowed the model to balance exploration and exploitation more effectively, resulting in higher returns and a better return-to-risk ratio compared to the other two approaches.

4.2.3 Performance of Time Series Incorporation with ARIMA and LSTM

As shown in Table 1, the ARIMA-enhanced Policy Gradient with Baseline model delivered a return of 885.61% and a Sharpe ratio of 4.25, slightly underperforming the original model, likely due to fewer training episodes. In contrast, the LSTM-enhanced model achieved a remarkable return of 2926.10% and the highest Sharpe ratio of 5.57, reflecting superior risk-adjusted performance. Its Hit ratio of 0.69 also indicates greater consistency, though it faced the highest Maximum Drawdown of $7,044,308.21, signalling increased exposure to market volatility.

As depicted in Figure 3, the LSTM-enhanced Policy Gradient with Baseline model demonstrated superior performance. The total reward per episode plot shows significant stability, with fewer large fluctuations, indicating a robust learning process. The cumulative reward per year plot highlights substantial performance gains, particularly in later episodes, demonstrating the model's ability to learn and adapt effectively. The action distribution plot for the last episode reveals optimised trading actions, aligning with the increased money transactions and substantial increase in net worth, reflecting the model's superior performance.

The LSTM-enhanced Policy Gradient with Baseline model's impressive performance stems from its ability to capture complex data patterns through deep learning. This is evidenced by its lower mean-squared error (MSE) in prediction. For AMD, LSTM's MSE of 58.86 outperformed ARIMA's 238.43. For Nvidia, LSTM's MSE was 446.29, far better than ARIMA's 3897.60. This enhanced accuracy led to better trading decisions, resulting in higher returns and an improved return-to-risk ratio.

Overall, the ability to incorporate a wide range of features into the LSTM model allowed for more comprehensive state representations, resulting in better-informed trading strategies and superior performance, demonstrating the effectiveness of combining deep learning techniques with reinforcement learning for stock trading.

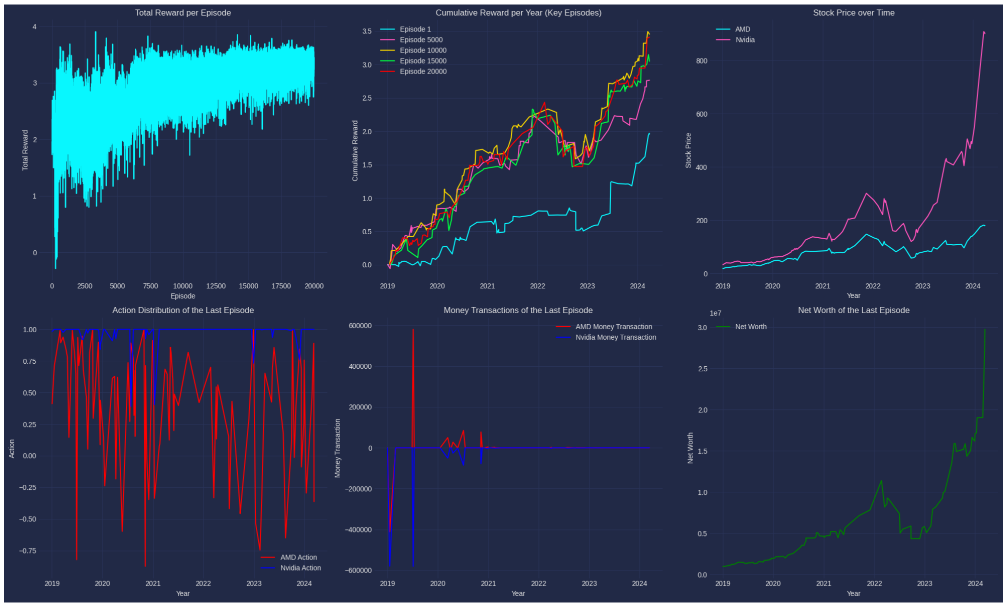

Figure 3: LSTM Training Performance

5 Conclusion

5.1 Summary of Findings

This paper investigated the application of reinforcement learning (RL) techniques to automate stock trading, with a focus on the semiconductor industry using AMD and Nvidia as case studies. The research explored a range of RL algorithms, including Deep Q-learning, Policy Networks, Policy Networks with Baseline, Actor-Critic with Epsilon-Greedy, as well as advanced time-series models such as ARIMA and LSTM within simulated trading environments.

The results revealed that Policy Networks integrated with LSTM in the Complex Trader environment yielded the highest return of 2926.10% and the highest Sharpe ratio of 5.57. This demonstrates the substantial advantages of combining sophisticated RL algorithms with deep learning methodologies, particularly in their ability to process and leverage complex temporal patterns in financial data. The integration of comprehensive indicators, spanning financial, economic, and social factors, has proven to enrich the trading strategy development process, enabling the RL agents to make more nuanced and informed decisions in dynamic market conditions.

From a practical standpoint, the study highlights the effectiveness of employing exploration strategies and no-action penalties in promoting active decision-making, thereby preventing the agent from defaulting to inaction and ensuring consistent performance improvements. Overall, this research contributes valuable insights into the potential of RL techniques in stock trading, providing a solid foundation for future advancements in this field.

5.2 Future Work

The integration of RL techniques in stock trading has shown promising results, particularly with advanced models like LSTM and Actor-Critic. However, there are several areas for future exploration:

- **Real-time Data Integration:** Incorporating real-time data of all the features to enhance model adaptability, responsiveness, and capabilities.
- **Sophisticated Risk Management:** Developing risk management strategies to mitigate potential losses.
- **Diverse Financial Instruments:** Extending the models to trade in diverse financial instruments such as options and futures.
- **Computational Efficiency:** Improving the computational efficiency of the models to handle larger datasets and more complex simulations.
- **Ethical Considerations:** Addressing ethical considerations in algorithmic trading to ensure fair and responsible trading practices.

References

[1] D. Ageng, C. Huang, and R. Cheng. A Short-Term Household Load Forecasting Framework Using LSTM and Data Preparation. *IEEE Access*, 9:167911–167919, 2021.

[2] W. B. Arthur. Complexity in economic and financial markets. *Complexity*, 1("'bibtex 1):20–25, 1995.

[3] M. Baig, M. Shahbaz, M. Imran, M. Jabbar, and Q. Ain. Relationship Between Gold and Oil Prices and Stock Market Returns. *Acta Universitatis Danubius*, 9(5):28–39, 2013.

[4] G. Batra, Z. Jacobson, S. Madhav, A. Queirolo, and N. Santhanam. Artificial-intelligence hardware: New opportunities for semiconductor companies. *McKinsey and Company*, 2019.

[5] Bloomberg. High-Frequency Trading Data Access, 2024. URL `https://www.bloomberg.com/professional/solution/bloomberg-terminal/`. Data accessed through Bloomberg Terminal for real-world automated trading applications, available via online brokers like Bloomberg.

[6] Boston Dynamics. Reinforcement Learning with Spot, 2024. Available at: `https://www.bostondynamics.com/robotics`.

[7] G. E. P. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, Hoboken, NJ, 5th edition, 2015.

[8] B. Butt, K. ur Rehman, M. Khan, and N. Safwan. Do Economic Factors Influence Stock Returns? A Firm and Industry Level Analysis. *African Journal of Business Management*, 4(5):583, 2010.

[9] A. Damodaran. *Investment Valuation: Tools and Techniques for Determining the Value of Any Asset*. John Wiley & Sons, 3rd edition, 2010.

[10] A. Dasgupta. *Deep Q Learning Applied to Stock Trading*. All Graduate Theses and Dissertations, Utah State University, 2020. URL `https://digitalcommons.usu.edu/etd/7983`.

[11] K. De Asis, J. Hernandez-Garcia, G. Holland, and R. Sutton. Multi-step reinforcement learning: A unifying algorithm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[12] K. Doya. Reinforcement learning in continuous time and space. *Neural computation*, 12(1):219–245, 2000.

[13] T. Eimer, M. Lindauer, and R. Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *International Conference on Machine Learning*, pages 9104–9149. PMLR, July 2023.

[14] M. El Hajj and J. Hammoud. Unveiling the influence of artificial intelligence and machine learning on financial markets: A comprehensive analysis of AI applications in trading, risk management, and financial operations. *Journal of Risk and Financial Management*, 16(10):434, 2023.

[15] F. J. Fabozzi and F. Modigliani. *Handbook of Financial Markets: Securities, Options, and Futures*. Prentice Hall, Upper Saddle River, NJ, 2004.

[16] Y. Fei, Z. Yang, Y. Chen, and Z. Wang. Exponential Bellman equation and improved regret bounds for risk-sensitive reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 20436–20446, 2021.

[17] M. Flannery and A. Protopapadakis. Macroeconomic Factors Do Influence Aggregate Stock Returns. *The Review of Financial Studies*, 15(3):751–782, 2002.

[18] A. Frino and D. Gallagher. Tracking S&P 500 Index Funds. *Journal of Portfolio Management*, 28(1), 2001.

[19] Y. Fu, D. Wu, and B. Boulet. Reinforcement learning based dynamic model combination for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6639–6647, June 2022.

[20] Google. Google Trends, 2024. URL `https://trends.google.com/`. Accessed: 2024-08-07.

[21] Google DeepMind. Cooling bill by 40%, 2016. Available at: `https://deepmind.com/blog/article/deepmind-ai-reduces-google-data-centre-cooling-bill-40`.

[22] S. Hochreiter and J. Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

[23] S. Ivanov, F. Jones, and J. Zaima. Analysis of DJIA, S&P 500, S&P 400, NASDAQ 100 and Russell 2000 ETFs and Their Influence on Price Discovery. *Global Finance Journal*, 24(3):171–187, 2013.

[24] G. Kaiser and B. Schwarz. Mathematical modelling as bridge between school and university. *ZDM*, 38:196–208, 2006.

[25] P. Kaleel and S. Sheen. Focused crawler based on reinforcement learning and decaying epsilon-greedy exploration policy. *International Arab Journal of Information Technology*, 20(5):819–830, 2023.

[26] T. Kim and H. Y. Kim. Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. *PLOS ONE*, 14(2):1–23, 02 2019. doi: 10.1371/journal.pone.0212320. URL `https://doi.org/10.1371/journal.pone.0212320`.

[27] M. Kiran and M. Ozyildirim. Hyperparameter Tuning for Deep Reinforcement Learning Applications. *arXiv preprint arXiv:2201.11182*, 2022.

[28] V. R. Konda and J. N. Tsitsiklis. Actor-Critic Algorithms. *Advances in Neural Information Processing Systems*, 12:1008–1014, 1999.

[29] R. Nguyen. *Impact of Tariffs on the Semiconductor Industry*. PhD Thesis, Institution Name, 2021.

[30] J. Oh, M. Hessel, W. M. Czarnecki, Z. Xu, H. P. van Hasselt, S. Singh, and D. Silver. Discovering reinforcement learning algorithms. In *Advances in Neural Information Processing Systems*, volume 33, pages 1060–1070, 2020.

[31] K. Pahwa and N. Agarwal. Stock market analysis using supervised machine learning. In *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pages 197–200. IEEE, 2019.

[32] J. Payne and W. Thomas. The Implications of Using Stock-Split Adjusted I/B/E/S Data in Empirical Research. *The Accounting Review*, 78(4):1049–1067, 2003.

[33] M. Pecka and T. Svoboda. Safe exploration techniques for reinforcement learning–an overview. In *Modelling and Simulation for Autonomous Systems*, pages 357–375. Springer International Publishing, 2014.

[34] A. Petropoulos, V. Siakoulis, E. Stavroulakis, P. Lazaris, and N. Vlachogiannakis. Employing google trends and deep learning in forecasting financial market turbulence. *Journal of Behavioral Finance*, 23(3):353–365, 2022.

[35] J. Pontiff and A. Woodgate. Shares outstanding and cross-sectional returns. *SSRN*, 2009.

[36] A. Preda. The sociological approach to financial markets. *Journal of Economic Surveys*, 21(3):506–533, 2007.

[37] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

[38] N. Raza, S. Shahzad, A. Tiwari, and M. Shahbaz. Asymmetric Impact of Gold, Oil Prices and Their Volatilities on Stock Prices of Emerging Markets. *Resources Policy*, 49:290–301, 2016.

[39] J. Rijsdijk, L. Wu, G. Perin, and S. Picek. Reinforcement Learning for Hyperparameter Tuning in Deep Learning-Based Side-Channel Analysis. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2021 (3):677–707, 2021.

[40] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal Policy Optimization Algorithms. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1987–1996. PMLR, 2017.

[41] B. Sousa, C. Alves, M. Mendes, and M. Au-Yong-Oliveira. Competing with Intel and Nvidia: The Revival of Advanced Micro Devices (AMD). In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–7. IEEE, 2021.

[42] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation*, pages 212–223. Springer Berlin Heidelberg, 2002.

[43] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 12:1057–1063, 1999.

[44] J. Taalbi. Origins and Pathways of Innovation in the Third Industrial Revolution. *Industrial and Corporate Change*, 28(5):1125–1148, Oct. 2019. doi: 10.1093/icc/dty053.

[45] I. Tsai. The Source of Global Stock Market Risk: A Viewpoint of Economic Policy Uncertainty. *Economic Modelling*, 60:122–131, 2017.

[46] M. Van Otterlo and M. Wiering. Reinforcement learning and markov decision processes. In *Reinforcement Learning: State-of-the-art*, pages 3–42. Springer Berlin Heidelberg, 2012.

[47] M. Vanhulsel, D. Janssens, G. Wets, and K. Vanhoof. Simulation of Sequential Data: An Enhanced Reinforcement Learning Approach. *Expert Systems with Applications*, 36(4):8032–8039, 2009.

[48] T. Wang and J. Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

[49] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3-4):279–292, 1992.

[50] R. J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8(3-4):229–256, 1992.

[51] Z. Xiong, X. Liu, S. Zhong, H. Yang, and A. Walid. Practical deep reinforcement learning approach for stock trading. *CoRR*, abs/1811.07522, 2018. URL `http://arxiv.org/abs/1811.07522`.

[52] D. Xue and Z. Hua. ARIMA Based Time Series Forecasting Model. *Recent Advances in Electrical & Electronic Engineering (Formerly Recent Patents on Electrical & Electronic Engineering)*, 9(2):93–98, 2016.

[53] N. Yue, M. Caini, L. Li, Y. Zhao, and Y. Li. A comparison of six metamodeling techniques applied to multi building performance vectors prediction on gymnasiums under multiple climate conditions. *Applied Energy*, 332:120481, 2023.

[54] B. Zhang, R. Rajan, L. Pineda, N. Lambert, A. Biedenkapp, K. Chua, F. Hutter, and R. Calandra. On the Importance of Hyperparameter Optimization for Model-Based Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, pages 4015–4023. PMLR, 2021.