HIGH SCHOOL EDITION
Journal of Student Research
Volume 12 Issue 1 (2023)

Generalized Deep Reinforcement Learning for Trading

Junyoung Sim[1] and Benjamin Kirk[#]

[1]Ithaca High School, Ithaca, NY, USA
[#]Advisor

ABSTRACT

This paper proposes generalized deep reinforcement learning with multivariate state space, discrete rewards, and adaptive synchronization for trading any stock held in the S&P 500. Specifically, the proposed trading model observes the daily historical data of a stock held in the S&P 500 and multiple market-indicating securities (SPY, IEF, EUR=X, GSG), selects a trading action, and observes a discrete reward that is based on the correctness of the selected action and independent of the volatility of stocks. The proposed trading model's reward-maximizing behavior is optimized by using a standard deep q-network (DQN) with adaptive synchronization that stabilizes and enables to track learning performance on generalizing new experiences from each stock. The proposed trading model was trained on the top 50 holdings of the S&P 500 and tested on the top 100 holdings of the S&P 500 starting from 2006 to 2022. Experimental results suggest that the proposed trading model significantly outperforms the 100% long-strategy benchmark in terms of annualized return, Sharpe ratio, and maximum drawdown.

Introduction

As data science became a prominent field, many recent works proposed deep learning as an effective statistical framework for leveraging traditional quantitative trading methods such as technical analysis. Standard deep neural networks, long-short-term memory neural networks (LSTM), and even hybrid models combining convolutional neural networks (CNN) and LSTMs are some major deep learning models that have been proposed to be successful in predicting the direction of future prices of various financial securities [1, 2, 3, 4]. Above all, a subfield of deep learning known as deep reinforcement learning is also frequently mentioned in current literature on quantitative trading. Deep reinforcement learning was introduced in [5] as an unsupervised deep learning algorithm where an agent interacts with some environment by observing a state that represents the environment and performing a reward-maximizing action. Thus, deep reinforcement learning is often used for sequential decision-making tasks performed in a complex environment, such as video games [6], that may consist of multiple variables. In the context of quantitative trading, many previous works used deep reinforcement learning to make practical and profit-maximizing decisions that outperform traditional quantitative trading methods and cutting-edge models by observing raw time series, convolution-processed time series, technical indicators, or correlated pairs of various assets, such as stocks, futures contracts, commodities, foreign exchanges, and even cryptocurrencies [7].

Despite the advantages of using deep reinforcement learning for trading, current literature [7, 8, 9, 10, 11] is limited to using the historical data related to one particular security being traded with very few exceptions [12, 13]. The tendency to heavily depend on the historical data of one particular security may not be as effective as observing the time series of multiple securities where more useful relationships and trading signals can be exploited. This is indeed crucial in a broader economic perspective because any security is related to other securities and a variety of external factors that may not be evident from the historical data of one particular

security. Generalizability and reproducibility are also some other limitations in current literature. Specifically, continuous reward functions based on observed profit or loss used to optimize a trading agent's behavior in previous works may have overfitting issues due to the volatility of certain securities and sectors. Moreover, many previous works proposed deep reinforcement learning algorithms that are specialized to trade a very limited number of securities. Institutionalized or undisclosed dataset, code, and important learning parameters also increased the difficulty in generalizing and reproducing such works. Furthermore, as previous works primarily focused on increasing profitability, fundamental issues in deep reinforcement learning, such as improving learning performance, are not addressed.

This paper proposes a novel trading model that utilizes the known advantages of deep reinforcement learning for trading while attempting to address the limitations in current literature. Specifically, instead of observing time series data related to one particular security being traded, the historical data of multiple market-indicating securities are used by the proposed trading model to trade any stock based on important market trends. Furthermore, a discrete reward function based on the correctness of a trading action and a standard deep q-network (DQN), one of the major algorithms in deep reinforcement learning, optimizes the trading model's reward-maximizing behavior on many stocks along with adaptive synchronization that stabilizes and enables to track learning performance. For better generalization, the trading model was trained on the top 50 holdings of the S&P 500 and tested on the top 100 holdings of the S&P 500, a major index fund in the US. For better reproducibility, daily historical data obtained from an open-sourced API were used, and the source code of the trading model is also open-sourced. Experimental results suggest that the trading model yields an annualized return of 23.75%, a Sharpe ratio of 0.6622, and a maximum drawdown of 48.70% from the top 100 holdings of the S&P 500, all of which significantly exceeds the 100% long-strategy benchmark.

The remainder of this paper is organized as follows. Section 2 outlines the technical background of the three major algorithms in deep reinforcement learning. Section 3 provides a review of current literature on deep reinforcement learning for trading. Section 4 outlines the method of the proposed trading model. Section 5 and 6 reports and discusses the results of the trading model. Lastly, section 7 concludes this paper with a summary and suggestions for further work.

Background

In a Markov Decision Process (MDP), an agent interacts with an environment by observing a state at time *t* that represents the environment (s_t), performing an action (a_t), and receiving a reward (R_{t+1}). The objective of an agent is to find an optimal policy that maximizes reward. An agent's reward-maximizing behavior can be optimized through the following three unsupervised algorithms in deep reinforcement learning: deep q-networks (DQN), policy gradients (PG), and advantage actor-critic models (A2C). This paper is primarily interested in DQNs since it is the algorithm in which the proposed trading model is based upon.

Deep Q-Networks (DQN)

In Q-learning, the optimal value of selecting an action (a) under some policy (π) when given some state (s) is defined as the maximum expected sum of the observed reward and future rewards discounted by γ [5, 6, 8, 14, 15].

Equation 1: Optimal action value in Q-learning:

$$Q^*(s,a) = \max_{\pi} \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \mid s,a\right]$$

Due to practical constraints, the optimal value of an action is defined as $Q^*(s,a) = \mathbb{E}\left[r + \gamma \max_{a'} Q^*(s',a') \mid s,a\right]$ where s is the current state, a is the selected action, r is the observed reward,

and a' is some action selected at s' that follows s. DQNs can be used to estimate $Q^*(s,a)$ through trial-and-error and converge to some reward-maximizing policy without modeling the environment. Specifically, a neural network with parameters θ_i uses the epsilon-greedy policy that either explores a random action by a certain probability (ε) or exploits a greedy action that is expected to yield the highest reward. Once a reward is observed for the selected action, a neural network with parameters θ_{i-1} chooses some action that is expected to yield the highest reward in the future. The optimal value of the selected action is then estimated as $Q^*(s,a) = r + \gamma \max_{a'} Q(s',a';\theta_{i-1})$. θ_i is updated by minimizing the squared temporal difference, $L(\theta_i)$, through gradient descent in the direction of $\nabla_{\theta_i} Q(s,a;\theta_i)$ based on random samples from previous experiences such that its approximation of $Q(s,a;\theta_i) \rightarrow Q^*(s,a)$ where $Q(s,a;\theta_i)$ is a linear approximator that follows a series of non-linear hidden layers in a DQN.

Equation 2: Loss function in DQNs:

$$L(\theta_i) = [Q^*(s,a) - Q(s,a;\theta_i)]^2 = \left[\left(r + \gamma \max_{a'} Q(s',a';\theta_{i-1})\right) - Q(s,a;\theta_i)\right]^2$$

Note that θ_i and θ_{i-1} are identical in shape, and θ_{i-1} is typically held constant until it is synchronized to θ_i after a fixed number of experiences. θ_i and θ_{i-1} are often referred to as the agent network and target network, respectively.

Policy Gradients (PG)

Unlike DQNs, PGs modify an existing policy to directly find an optimal policy that maximizes reward [8, 14, 15]. Specifically, if a neural network with parameters θ representing some policy is defined as $\pi(a \mid s;\ \theta)$ and the expected cumulative reward is defined as $\mathbb{E}[R_t]$, PGs update θ through gradient ascent in the direction of $\nabla_\theta \log \pi(a \mid s;\ \theta)\, R_t$ to find a reward-maximizing policy. Since PGs output a probability for each possible action, PGs are useful for tasks that involve continuous action spaces.

Advantage Actor-Critic (A2C)

To reduce variance and improve learning performance in PGs, A2C was proposed as an algorithm that can learn both a policy and a state-value function by using two distinct neural networks: an actor network and a critic network [8, 14, 15]. The objective of A2C is to update the actor network outputting a policy $\pi(a \mid s;\ \theta)$ by maximizing $J(\theta) = \mathbb{E}[\log \pi(a \mid s;\theta)\, A(s,a)]$ through gradient ascent where the advantage function is $A(s_t, a_t) = R_t + \gamma V(s_{t+1} \mid w) - V(s_t \mid w)$ and the state-value function $V(s \mid w)$ is outputted by the critic network that is updated through gradient descent. While A2C is a synchronous model having one actor, some works have shown effective applications of an asynchronous model known as A3C that has multiple parallel actors [16].

Literature Review

Current literature on deep reinforcement learning for trading primarily focused on futures contracts of various assets. For instance, [8] proposed an algorithm that can trade 50 futures contracts of various commodities, equity indexes, fixed income, and foreign exchanges. Similarly, the E-mini S&P 500 futures (ES) and the HS300 futures from the China Financial Futures Exchange (IF) were studied in [9] along with three stocks (AAPL, IBM, and PG). [10] also proposed an algorithm that can trade IF contracts as well as silver (AG) and sugar (SU) contracts, and ES was also studied in [11] along with two stocks (JPM and MSFT). Other studies such as [12] focused on foreign exchanges (EUR/USD) and [13] focused on trading the 30 stocks held in the Dow Jones Industrial Average.

Starting from futures contracts of various assets to stocks and foreign exchanges, current literature studied applications of deep reinforcement learning for trading financial securities in various markets. However, besides [8] where a relatively large number of securities were studied, the scope of most previous works is very limited. Although [9] studied both the futures market and stock market, only two futures contracts and three stocks were studied. Likewise, [10] only studied three futures contracts, [11] only studied one futures contract and two stocks, and [12] focused on one specific foreign exchange. In fact, the 30 stocks studied in [13] cannot be considered enough because those 30 stocks held in the Dow Jones Industrial Average are mainly value stocks having similar market behavior and volatility. Therefore, this may raise questions about the generalizability of current literature on deep reinforcement learning for trading because the algorithms proposed by previous works are specialized to trade a limited number of securities. Moreover, previous works mostly used intraday historical data that are often institutionalized, thereby increasing the difficulty of reproducing results. Thus, focusing on a particular financial market and maximizing the number and diversity of securities with more accessible historical data would be preferrable.

Given the type of securities studied in previous works, it is integral to discuss the Markov Decision Process formulation of current literature on deep reinforcement learning for trading as follows. The state space proposed in previous works heavily rely on technical indicators of a particular security. For instance, [8] used normalized price series and multiple momentum indicators (periodic returns, MACD, RSI) of a particular security being traded. Similarly, [10] used raw price and momentum changes of a particular security and [9] used various technical indicators (MACD, MA, EMA, ATR, ROC) and the open-close-high-low-volume data (OCHLV) of a particular asset being traded. It is interesting to note that [9] also included remaining cash and Sharpe ratio (return-over-risk) of previous transactions to provide interactive feedback. A more unique approach was used in [11] where the raw price data of some asset with different time intervals were combined to create an image such that convolutions can be applied for feature extraction. Very few works used time series data of multiple securities for their state space. For instance, [12] used the raw price series of multiple cross-correlating currencies and [13] used the raw prices series of 30 stocks held in the Dow Jones Industrial Average.

Both discrete and continuous action spaces were used in current literature on deep reinforcement learning for trading [7]. Specifically, the discrete action space $a \in \{-1, 0, 1\}$ was conventionally used in previous works to enable short, no holding, and long positions. For continuous action spaces, previous works used $a \in [-1, 1]$ such that multiple trading positions with different weightings can be executed for some asset. The reward functions proposed in current literature slightly varied, albeit they generally used a continuous reward function based on observed profit or loss [8, 9, 10, 11]. Transaction costs and price slippage in futures contracts were also included in such continuous reward functions, albeit [7] surveyed that the impact they have on converging to an optimal policy is unknown. A rare exception to using a continuous reward function was shown in [12] where discrete rewards (+1 or -1) were given based on the correctness of trend predictions.

A major limitation in the Markov Decision Process formulation of current literature on deep reinforcement learning for trading is the over-reliance on technical indicators and other time series data related to one particular security being traded. Although technical analysis is widely used in the financial industry for trend-following decision-making, a survey of previous works suggests that technical indicators and historical prices of a particular security are lagging indicators that cannot provide profitable forecasts [17]. Thus, rather than depending on the time series data being predicted, some works [7, 12] proposed that more useful relationships and trading signals can be exploited when the historical data of multiple correlated securities are used as inputs. The latter is indeed preferable because the price of any financial security is influenced by a variety of external factors: the relationship between correlated securities, corporate performance, popularity, competition, federal monetary policy, and domestic and international events are some examples to name a few. Therefore, using the time series data of multiple securities with known economic significance would provide a more effective state representation of the financial market that is integral to learning performance [18], yet previous works did not study such a state space [7].

The reward functions used in current literature on deep reinforcement learning for trading also raise some questions. Specifically, a major issue in continuous reward functions based on observed profit or loss is that a trading model can be restricted to trade a limited number of securities as shown in previous works. This is because observed profit or loss can vary and have extreme values depending on the volatility of certain securities and sectors, thereby increasing the likelihood of overfitting into a limited number of securities. Thus, an alternative approach could be using a discrete reward function where a trading model is rewarded +1 point for placing a winning bet that returns a profit and -1 point for a losing bet that returns a loss. Such a discrete reward function based on the correctness of a trading action and independent of volatility would enable a trading model to avoid overfitting issues and precisely estimate the optimal value of some trading action for many different securities.

A mix of DQNs, PGs, and A2Cs were used in current literature on deep reinforcement learning for trading. DQNs tend to be used as baseline models, whereas A2Cs and A3Cs were most used as an outperforming alternative to PGs that were rarely used. LSTMs were the most common neural network architecture used in previous works, albeit some applied other neural network architectures to enhance feature extraction from state representations. Fuzzy neural networks used in [10] and ensembled convolutional neural networks used in [11] are some examples. All three deep reinforcement learning algorithms significantly outperformed the 100% long-strategy benchmark and other traditional techniques in both returns and Sharpe ratio. However, [8] showed that DQNs outperformed PGs and A2Cs across multiple financial markets while others [9, 10] suggested that A2C and A3C models are better than DQNs and other baseline techniques.

Current literature on deep reinforcement learning for trading have proposed distinct Markov Decision Processes with different input variables and powerful deep learning models to improve profitability. However, [7] suggests that previous works rarely address the fundamental issues in deep reinforcement learning. For instance, [19] mentioned that synchronizing the target network to the agent network in DQNs after a fixed number of frames can cause the loss of properly learned networks, yet previous works in deep reinforcement learning for trading did not address these problems anyhow. Furthermore, [7] also suggests that it is difficult to identify and compare the individual factors that affected performance because previous works used drastically different datasets, Markov Decision Processes, and deep learning models. Reproducing such works are also difficult because most previous works did not open-source their code. Considering these limitations in current literature, implementing a new Markov Decision Process that improves generalization and using a standard DQN to effectively test the new environment while improving learning performance through some method adaptive synchronization and open-sourcing code would be preferrable.

Method

Figure 1 shows the pipeline of generalized deep reinforcement learning with multivariate state space, discrete rewards, and adaptive synchronization that enables the proposed trading model to trade any stock held in the S&P 500. Note that the S&P 500 is one of the largest index funds in the US that holds a diverse list of publicly tradeable companies. Thus, unlike previous works that proposed trading models specialized to trade a very limited number of securities, training and testing the proposed trading model on the S&P 500 holdings ensures generalizability to many US stocks as possible. A multivariate state space consisting of the historical data of a stock and multiple market-indicating securities allows the trading model to trade a stock based on major market trends. A discrete reward function based on the correctness of a trading action and a standard DQN optimizes and generalizes the trading model's behavior on any stock held in the S&P 500. Adaptive synchronization stabilizes the trading model's learning performance and enables to track its improvement on generalizing new experiences from each stock during training. A detailed outline of such methodologies is provided throughout this section.

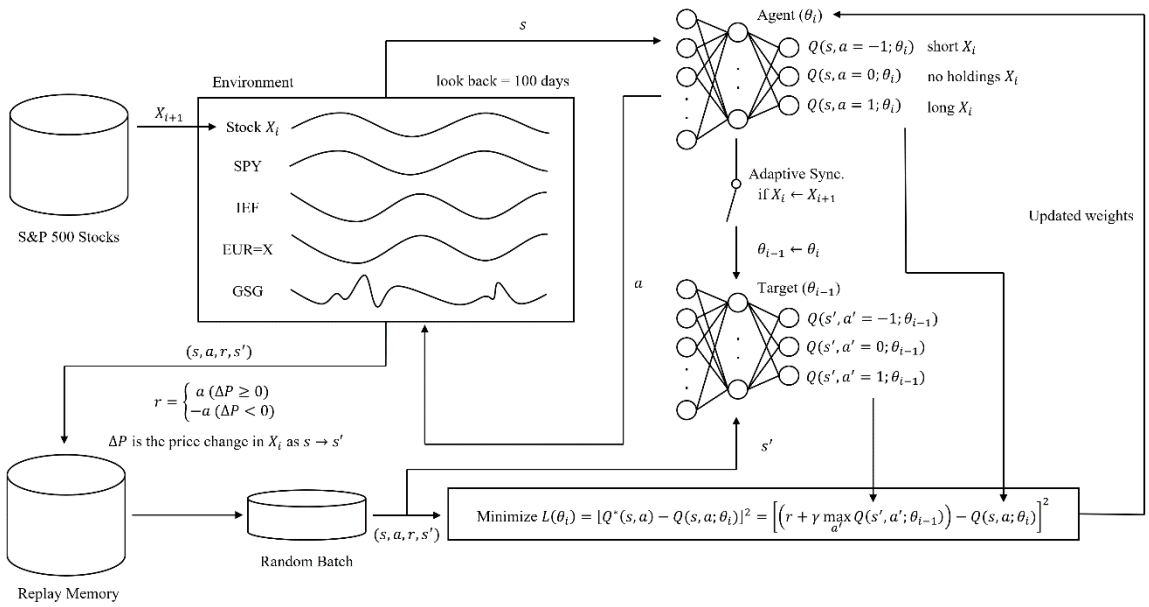

Figure 1. Model pipeline of generalized deep reinforcement learning with multivariate state space, discrete rewards, and adaptive synchronization for trading.

Markov Decision Process Formulation

Multivariate State Space

The state space proposed in previous works on deep reinforcement learning for trading have shown over-reliance on technical indicators and other time series data related to one particular security being traded [7, 8, 9, 10, 11]. Very few exceptions [12, 13] can be found where the historical data of multiple correlated securities were used to exploit more useful information and trading signals [7]. Therefore, a novel multivariate state space consisting of the historical data of several market-indicating securities is used by the trading model as follows. To trade a stock held in the S&P 500 at the end of every market day (t), the proposed trading model observes a state (s_t) that consists of the adjusted and standardized daily price series of the following five securities during the past 100 days before t inclusive: the stock of interest, the S&P 500 (SPY), the 10-year Treasury Bond (IEF), the USD/EUR exchange rate (EUR=X), and the S&P GSCI Commodity-Indexed Trust (GSG). The latter four (SPY, IEF, EUR=X, GSG) were selected as market-indicating securities used by the trading model because of their macroeconomic significance. Thus, with the adjusted and standardized daily price series of the five securities being concatenated to each other to form a single state, the trading model is given a multivariate state space to trade a stock based on important market trends. Table 1 provides a detailed reasoning behind the selection of the four market-indicating securities used by the trading model.

Table 1. Market-indicating securities included in the multivariate state space of the proposed trading model.

Indicator	Ticker	Description
S&P 500	SPY	The historical data of the S&P 500 is used by the trading model because the index fund is representative of the overall stock market and can be used to compare the relative performance of the stock being traded.

10-year Treasury Bond	IEF	The historical data of the 10-year treasury bond is used by the trading model because the bond has a significant role in macroeconomic cycles impacted by US federal monetary policy.
USD/EUR	EUR=X	The historical data of the USD/EUR exchange rate is used by the trading model because the exchange rate indicates macroeconomic cycles in terms of the relative value of the dollar impacted by US federal monetary policy.
S&P GSCI Commodity-Indexed Trust	GSG	The historical data of the S&P GSCI Commodity-Indexed Trust is used by the trading model because the commodity index is a leading indicator that holds multiple commodity assets, such as energy, industrial metals, agricultural materials, and livestock, that are indicative of overall market and supply chain health.

Action Space

Some previous works used the continuous action space $a \in [-1, 1]$ such that short and long positions can be placed with varying weightings for a particular security, albeit the discrete action space $a \in \{-1, 0, 1\}$ is most prevalent in current literature where each of three possible discrete actions typically correspond to short, no holdings, and long, respectively [7]. Thus, the proposed trading model also uses the discrete action space $a \in \{-1, 0, 1\}$ to trade a particular stock held in the S&P 500. Note that the trading model updates its trading action for a stock at the end of every market day and that no changes are made to the trading model's position if the same action is selected consecutively.

Discrete Reward

Most previous works used a continuous reward function based on observed profit or loss along with transaction costs and price slippage that occur in futures contracts [8, 9, 10, 11]. However, such continuous reward functions can only specialize in trading a limited number of securities because observed profit or loss can vary and have extreme values depending on the volatility of certain securities and sectors. Besides this overfitting issue, it is also unknown whether the restrictions included in those continuous reward functions, such as transaction costs and price slippage, significantly impact the optimal policy being learned [7].

Equation 3: Discrete reward function used by the proposed trading model.

$$R_{t+1} = \begin{cases} a_t \ (\Delta P \geq 0) \\ -a_t \ (\Delta P < 0) \end{cases}$$

Taking these limitations into account, the discrete reward function shown above is used such that the proposed trading model is rewarded at the end of every market day based on whether a_t was a winning bet when the daily price change of the stock being traded is denoted as $\Delta P = P_{t+1} - P_t$. Specifically, the trading model is rewarded +1 point for a long position and -1 point for a short position when $\Delta P \geq 0$ and -1 point for a long position and +1 point for a short position when $\Delta P < 0$. The trading model is rewarded 0 points for no holdings regardless of the sign of ΔP. Although simple, it is important to note that the proposed discrete reward function is only based on the correctness a selected trading action and independent of the volatility of stocks. Therefore, when combined with the proposed state space consisting of the historical time series of multiple market-indicating securities, the discrete reward function enables the trading model to effectively generalize which trading action is best rewarded based on major market trends without overfitting to the behavior of a limited number of stocks.

Generalized Deep Reinforcement Learning

Previous works [8, 9, 10, 11, 12, 13] have shown effective applications of deep reinforcement learning for trading various financial securities. However, many were specialized to a limited number of securities, did not consider the fundamental challenges in learning performance, and made it difficult to identify which individual factor affected the reported results as previous works used different datasets, Markov Decision Processes, and deep learning models [7]. With important learning parameters, code, and datasets being institutionalized or undisclosed, reproducing the results of previous works is also difficult. Therefore, the proposed trading model uses a standard DQN to effectively test its Markov Decision Process that is designed to improve generalizability by having a multivariate state space based on publicly accessible historical data of several market-indicating securities and a discrete reward function based on the correctness of trading actions. Furthermore, a method of synchronization that is adaptive of learning sequence is also used to stabilize and precisely track the trading model's improvement on generalizing new experiences from each stock during training.

Data Preprocessing

The dataset used for training the proposed trading model is denoted as $\mathcal{D} = (\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \dots, \mathcal{D}_{50})$ where $\mathcal{D}_i$ contains the adjusted daily historical data of a stock included in the top 50 holdings of the S&P 500 and the four market-indicating securities (SPY, IEF, EUR=X, GSG). The adjusted daily historical data of stocks and the four market-indicating securities used for the trading model are retrieved from Yahoo Finance where historical data are publicly available. For every $\mathcal{D}_i$, the historical data of the stock of interest and the four market-indicating securities are ensured to have no mismatching dates, though the matching dates (typically 2006 ~ 2022) varies for each $\mathcal{D}_i$ depending on the stock of interest. Moreover, using the look back duration of 100 days and the number of matching dates in each $\mathcal{D}_i$, the number of observable states in each $\mathcal{D}_i$ is determined. For training purposes, the size of entire dataset $\mathcal{D}$ is subsequently determined by adding the number of observable states in each $\mathcal{D}_i$.

Learning Setup and Algorithm

Among the various algorithms in deep reinforcement learning, DQNs are consistently used as baseline models to test and compare novel solutions [8, 9, 10]. Thus, to effectively test the proposed trading model's Markov Decision Process, the trading model's reward-maximizing behavior is optimized by using a standard DQN with five fully connected hidden layers (450-400-350-300-250) having rectified linear units. The pseudocode of the algorithm used to train the trading model is shown as follows.

Shuffle $\mathcal{D}$ (prevents the ranking of the top 50 holdings of the S&P 500 from causing learning bias)
Initialize replay memory $\mathcal{M}$
for $\mathcal{D}_\text{i}$ in $\mathcal{D}$
…. start = look_back – 1 (look_back = 100 days)
…. terminal = $\mathcal{D}_\text{i}$.stock().len() – 2
…. for t in [start, terminal]
…. …. Sample s (100-day closing price series of a stock and the four market-indicating securities)
…. …. Select a through the ε-greedy policy ($a \in \{-1, 0, 1\}$; short, no holdings, long)
…. …. Observe reward $r = \begin{cases} a\ (\Delta P \geq 0) \\ -a\ (\Delta P < 0) \end{cases}$ where $\Delta P = P_{t+1} - P_t$ is the price change of the stock in $\mathcal{D}_\text{i}$
…. …. Calculate $Q^*(s, a) = r + \gamma \max_{a'} Q(s', a'; \theta_{i-1})$
…. …. Store new experience $(s, a, Q^*(s, a))$ in $\mathcal{M}$
…. …. if $\mathcal{M}$.size() is $\mathcal{M}$.capacity()
…. …. …. Sample a random batch from $\mathcal{M}$
…. …. …. Update θ_i through gradient descent by minimizing $L(\theta_i)$ in the direction of $\nabla_{\theta_i} Q(s, a; \theta_i)$
…. …. …. Remove oldest experience in $\mathcal{M}$
…. Adaptive synchronization ($\theta_{i-1} \leftarrow \theta_i$ when $\mathcal{D}_\text{i} \leftarrow \mathcal{D}_{\text{i}+1}$)

Relevant learning parameters are outlined as follows. The exploration probability (ε) decreases linearly from 1.00 during the first 10% of the entire dataset $\mathcal{D}$ until it is held constant once it reaches a minimum of 0.10. The discount rate for future rewards is $\gamma = 0.80$ to better account for long-term rewards. The replay memory ($\mathcal{M}$) capacity is set to 10% of the size of the entire dataset $\mathcal{D}$. To learn which action is best rewarded when given certain states through experience replay, a random batch of 10 samples from $\mathcal{M}$ is selected to train the agent network (θ_i) of the trading model through gradient descent where the learning rate has an initial value of 1.0×10^{-5} that decreases exponentially until it reaches 1.0×10^{-8} at the last frame in $\mathcal{D}$. L2 regularization constant of 0.10 is also used throughout the entire training process to prevent exploding gradients. It is important to note that the target network (θ_{i-1}) is adaptively synchronized to the agent network (θ_i) after observing all states in $\mathcal{D}_i$ before transitioning to $\mathcal{D}_{i+1}$ rather than after observing a fixed number of frames as widely practiced in previous works such as [8]. The issue with synchronizing after a fixed number of frames is that the number of states in each $\mathcal{D}_i$ are not necessarily identical due to date differences in the historical data of each stock, hence learning can be unstable and difficult to track if synchronization occurs in the middle of learning to trade some stock in $\mathcal{D}_i$. However, the proposed adaptive synchronization updates the target network of the trading model after testing the existing parameters on each $\mathcal{D}_i$, thereby stabilizing learning performance by preserving well-learned parameters and allowing to precisely track how well the trading model generalize new experiences from each stock in $\mathcal{D}_i$.

Data Collection and Implementation

The proposed trading model was compared to the 100% long-strategy benchmark during training and testing. To track the trading model's learning performance, the mean return-on-investment of the trading model and the 100% long-strategy benchmark were recorded as the trading model gained experience on the top 50 holdings of the S&P 500. After training on the top 50 holdings of the S&P 500, the trading model was tested on the top 100 holdings of the S&P 500. The following metrics [8] were collected to evaluate the performance of the trading model throughout the entire historical period available (typically 2006 ~ 2022) for each of the top 100 holdings of the S&P 500.

Table 2. Metrics collected to evaluate the proposed trading model.

Metric	Description
E(R)	Annualized return
S(R)	Standard deviation of annualized returns
Sharpe	E(R) / S(R) measures the return-over-risk ratio
MDD	Maximum drawdown measures maximum volatility

The mean of annualized returns, standard deviation of annual returns, and the maximum drawdown of the trading model on the top 100 holdings of the S&P 500 were calculated. The mean of annualized returns and standard deviation of annual returns were then used to derive the mean Sharpe ratio of the trading model. The actions selected by the trading model during the entire historical period of each stock were also recorded to compare the decisions made by the trading model for different stocks. Full build and test data are open-sourced at [20] along with the source-code of the trading model that is built in C++ from scratch.

Results

Figure 2 shows how the cumulative mean return-on-investment of the trading model and the 100% long-strategy benchmark compared to each other as the trading model gained experience on the top 50 holdings of the S&P 500. Note that the number of experiences is the number of stocks the trading model learned to trade out of the top 50 holdings of the S&P 500 and the y-axis represents mean return-on-investment on x stocks in terms of n-times the initial investment.

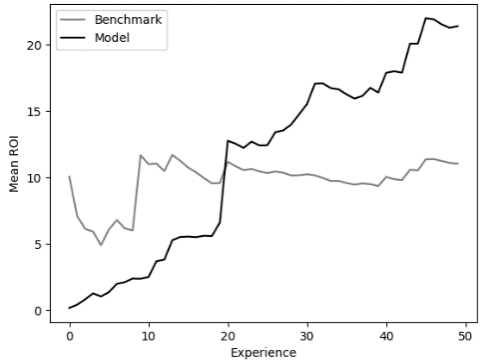

Figure 2. Cumulative mean return-on-investment of the trading model and 100% long-strategy benchmark as the trading model gained experience in trading the top 50 holdings of the S&P 500.

The box plots in Figures 3 and 4 compare the distribution of annualized returns, Sharpe ratios, and maximum drawdowns of the trading model's performance and the 100% long-strategy benchmark's performance on the top 100 holdings of the S&P 500. Table 3 shows the mean of annualized returns, standard deviation of returns, Sharpe, and maximum drawdown of the trading model's performance and the 100% long-strategy benchmark's performance on the top 100 holdings of the S&P 500.

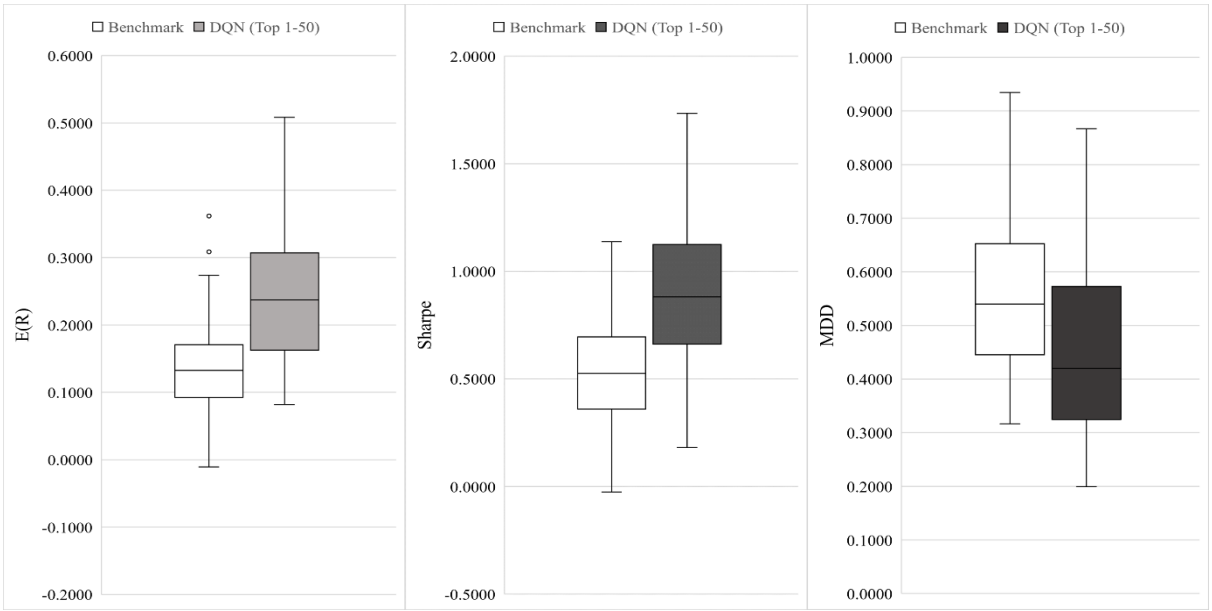

Figure 3. Comparing the distribution of annualized returns, Sharpe ratios, and maximum drawdowns of the trading model and the 100% long-strategy benchmark on the top 1st to 50th holding of the S&P 500 (trained).

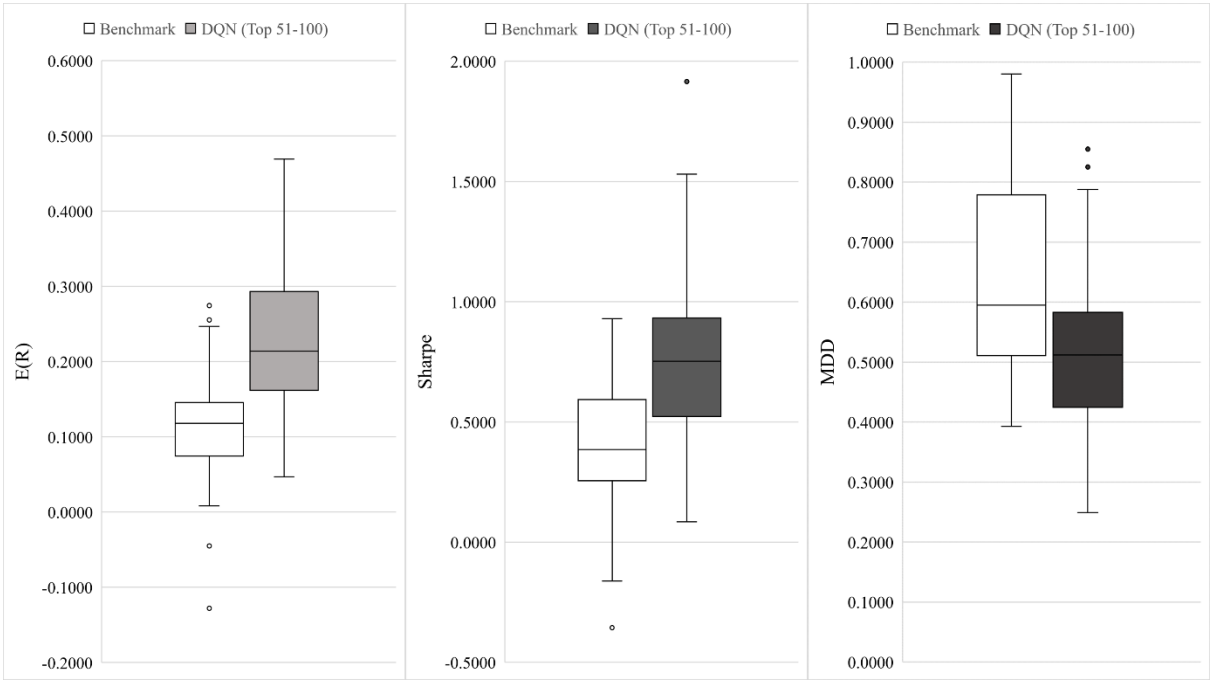

Figure 4. Comparing the distribution of annualized returns, Sharpe ratios, and maximum drawdowns of the trading model and the 100% long-strategy benchmark on the top 51st to 100th holding of the S&P 500 (untrained).

Table 3. Mean performance of the trading model and 100% long-strategy on the top 100 holdings of the S&P 500.

S&P 500 Top 100 Holdings (Mean Performance)				
	E(R)	**S(R)**	**Sharpe**	**MDD**
Benchmark	0.1251	0.3093	0.4044	0.5975
DQN	0.2375	0.3587	0.6622	0.4870

Figures 5 through 14 compare the return-on-investment of the trading model (light gray) and the 100% long-strategy benchmark (dark gray) from the top 10 holdings of the S&P 500. Note that the black graphs below the comparison of historical return-on-investments in Figures 5 through 14 show the actions selected by the trading model where 0, 1, and 2 indicates short, no holdings, and long, respectively. Table 4 in the Appendix contain full test results on the top 100 holdings of the S&P 500.

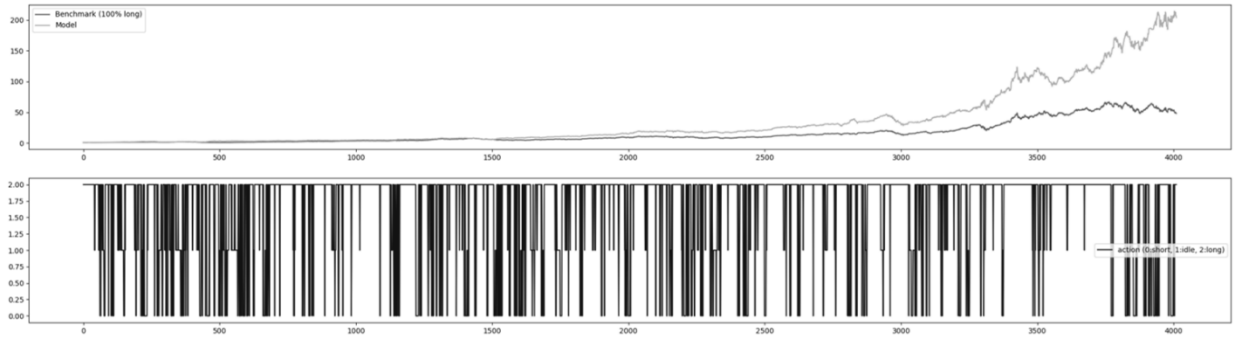

Figure 5. Return-on-investment of the trading model and 100% long-strategy on AAPL.

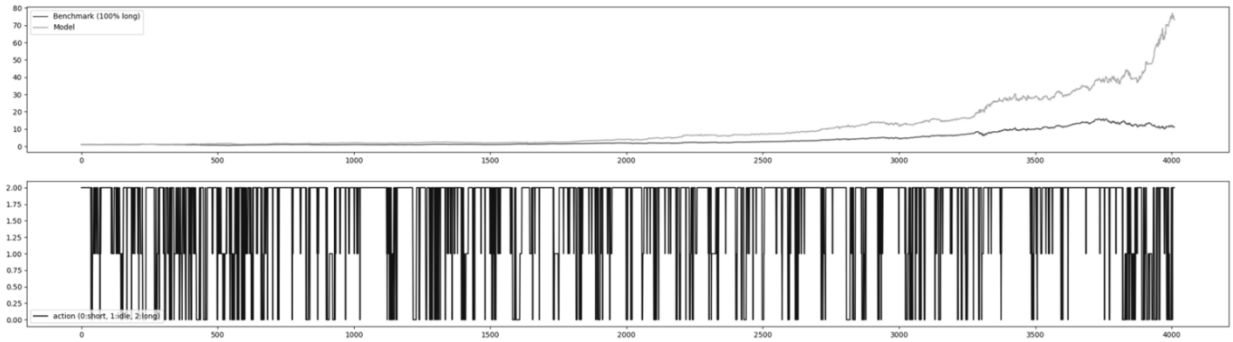

Figure 6. Return-on-investment of the trading model and 100% long-strategy on MSFT.

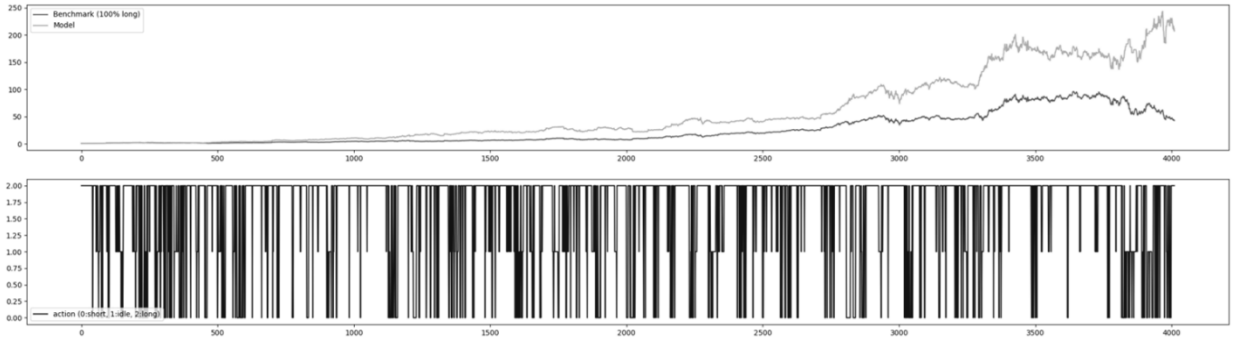

Figure 7. Return-on-investment of the trading model and 100% long-strategy on AMZN.

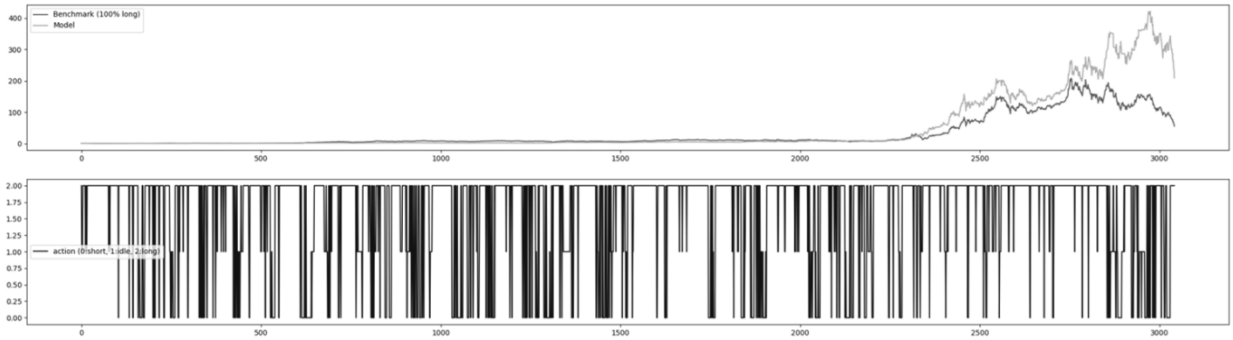

Figure 8. Return-on-investment of the trading model and 100% long-strategy on TSLA.

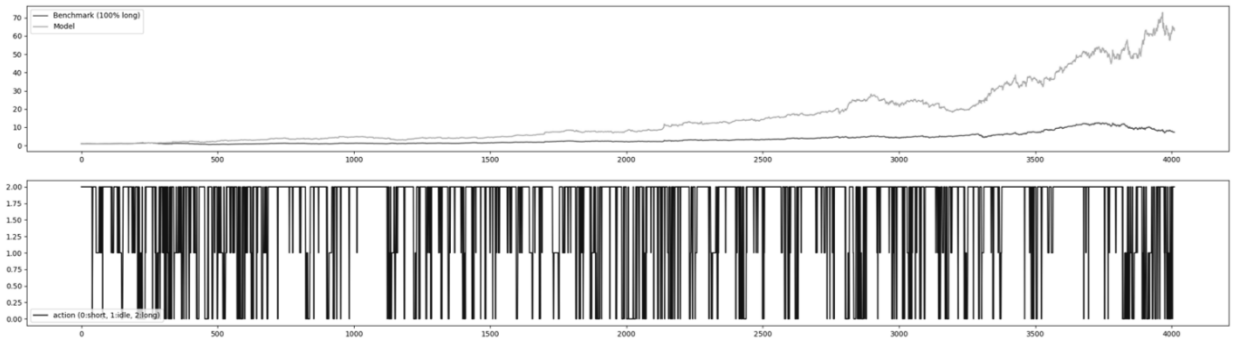

Figure 9. Return-on-investment of the trading model and 100% long-strategy on GOOG.

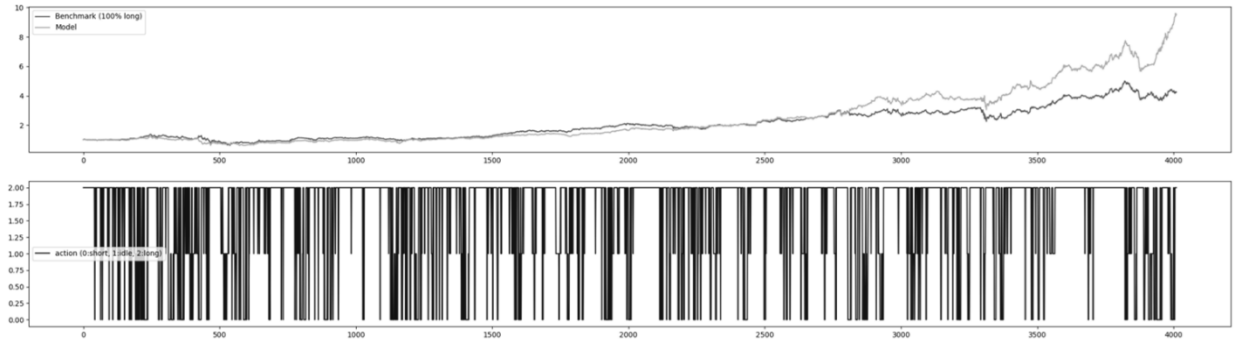

Figure 10. Return-on-investment of the trading model and 100% long-strategy on BRK-B.

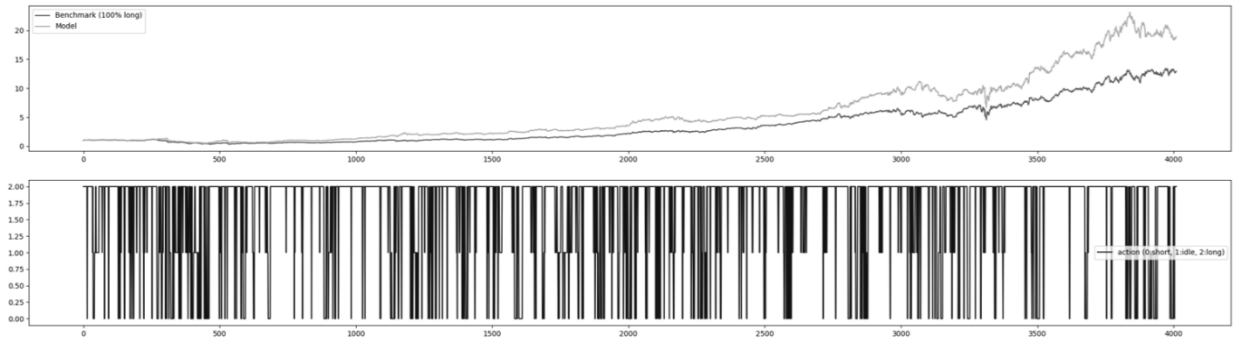

Figure 11. Return-on-investment of the trading model and 100% long-strategy on UNH.

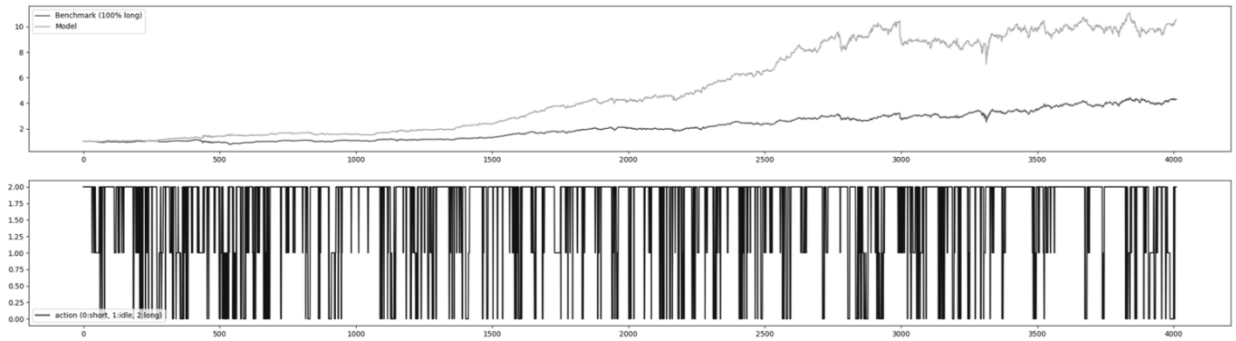

Figure 12. Return-on-investment of the trading model and 100% long-strategy on JNJ.

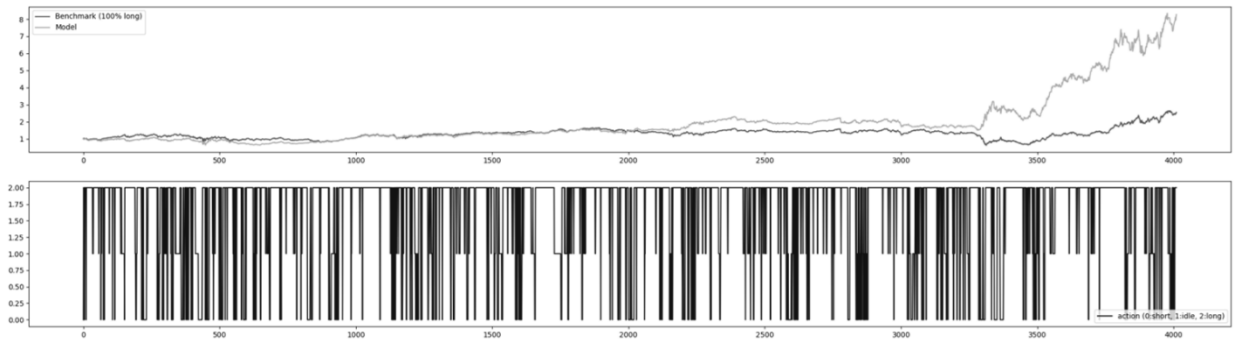

Figure 13. Return-on-investment of the trading model and 100% long-strategy on XOM.

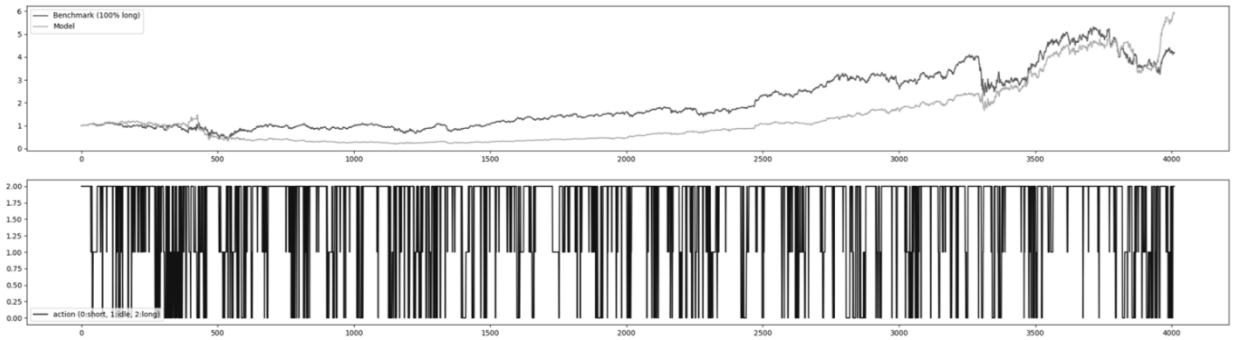

Figure 14. Return-on-investment of the trading model and 100% long-strategy on JPM.

Discussion

The proposed adaptive synchronization occurs after completely learning to trade each of the top 50 holdings of the S&P 500. This stabilizes learning by allowing to test and update the trading model's existing parameters on each stock and precisely track how the trading model improved on generalizing new experiences form each stock. Indeed, this can be observed from Figure 2 where the mean return-on-investment of the trading model consistently increased as it gained experience in trading the top 50 holdings of the S&P 500. In the beginning, the trading model underperformed the 100% long-strategy benchmark, yet the trading model surpassed the benchmark after learning to trade 20 stocks out the top 50 holdings of the S&P 500, suggesting that the trading model quickly acquired a reward-maximizing behavior that converted to higher returns.

Further data suggests that the trading model significantly outperforms the 100% long-strategy benchmark on the top 100 holdings of the S&P 500. Specifically, Figure 3 shows that, in terms of annualized return, Sharpe ratio, and maximum drawdown, the trading model outperforms the 100% long-strategy benchmark on stocks it was trained on (top 50 holdings of the S&P 500). Figure 4 shows the trading model also outperforms the 100% long-strategy benchmark on stocks it was not trained on (top 51st to 100th holding of the S&P 500). Randomized simulations suggest that the observed differences in mean annualized return, Sharpe ratio, and maximum drawdown of the trading model and the 100% long-strategy benchmark on the top 100 holdings of the S&P 500, both trained and untrained, are statistically significant with p-values substantially below the significance level of 0.01, indicating that the trading model effectively yields higher returns with lower risk than the 100% long-strategy benchmark. It is also notable that the trading model yields higher returns with lower risk from stocks it was trained on (top 50 holdings of the S&P 500) than from stocks it was not trained on (top 51st to 100th holding of the S&P 500). However, the observed differences in trained and untrained performance have p-values above the significance level of 0.01, suggesting that such differences are likely due to differences in the volatility and growth of the top 100 holdings of the S&P 500 rather than an overfitting issue. Overall, Table 3 shows that, on average, the trading model yields an annualized return of 23.75% a Sharpe ratio of 0.6622, and a maximum drawdown of 48.70%, all of which significantly exceeds the 100% long-strategy benchmark that yields an annualized return of 12.51%, a Sharpe ratio of 0.4044, and a maximum drawdown of 59.75% from the top 100 holdings of the S&P 500. Such high performance of the trading model can also be observed from Figures 5 through 14 where the trading model yields a significantly higher historical return-on-investment than the 100% long-strategy benchmark. Note that the trading model behaves in a similar manner for different stocks as shown in the action graphs in Figures 5 through 14, albeit the actions selected for each stock are not necessarily identical. This suggests that the trading model generally chooses similar trading actions based on major market trends represented by the four market-indicating securities with some variability depending on how the price history of a stock relates to such major market trends. As a cohesive whole, these results ultimately suggest that the trading model's high performance on many stocks held in the S&P 500 can be attributed to the

proposed multivariate state space based on relationship between the price history of a stock and major market-indicating securities and the proposed discrete reward function that is based on the correctness of trading actions and independent of volatility. Indeed, the use of a standard DQN instead of other deep reinforcement learning algorithms and neural network architectures makes it reasonable to identify the proposed multivariate state space and discrete reward function as the primary contributors to the trading model's high performance and generalizability. However, it is important to note that some inconsistency in performance was observed when the trading model did not perform well in trading JPM compared to other top 10 holdings of the S&P 500 as shown in Figures 5 through 14. This likely occurred because the trading model may not have quickly exited from shorting JPM after the 2008 financial crisis. Exiting from a short position later than preferred is a common issue the trading model had when tested on some stocks (VRTX, MO, CI, AMD, PG) where the annualized return of the trading model underperforms the 100% long-strategy benchmark.

Conclusion

This paper proposed generalized deep reinforcement learning with multivariate state space, discrete rewards, and adaptive synchronization for trading any stock held in the S&P 500. Experimental results obtained by testing the proposed trading model on the top 100 holdings of the S&P 500 suggest that the trading model, on average, yields higher returns with lower risk than the 100% long-strategy benchmark. Such results can be attributed to the proposed multivariate state space and discrete reward function that allows the trading model to generalize on many stocks held in the S&P 500 based on major market trends. Furthermore, adaptive synchronization helps to stabilize learning performance and track the trading model's improvement on generalizing new experiences from each stock. To improve the trading model's performance and robustness, implementing a method to prevent losses from short positions by adding interactive feedback for past actions and rewards to the state space of the trading model would be an important topic to study in the future. Researching other effective market-indicating securities, studying other financial securities, and testing other deep reinforcement learning algorithms with more robust neural network architectures would also be some valuable topics for further work.

Acknowledgments

I would like to thank my advisor for the valuable insight provided to me on this topic.

References

[1] A. W. Li and G. S. Bastos, "Stock Market Forecasting Using Deep Learning and Technical Analysis: A Systematic Review," *IEEE Access*, vol. 8, pp. 185232–185242, 2020, doi: 10.1109/ACCESS.2020.3030226.
[2] B. Lim, S. Zohren, and S. Roberts, "Enhancing Time-Series Momentum Strategies Using Deep Neural Networks," *The Journal of Financial Data Science*, 2019, doi: 10.3905/jfds.2019.1.015.
[3] S. Siami-Namini and A. S. Namin, "Forecasting Economics and Financial Time Series: ARIMA vs. LSTM." arXiv, 2018. doi: 10.48550/ARXIV.1803.06386.
[4] Q. Guo, S. Lei, Q. Ye, and Z. Fang, "MRC-LSTM: A Hybrid Approach of Multi-scale Residual CNN and LSTM to Predict Bitcoin Price," in *2021 International Joint Conference on Neural Networks (IJCNN)*, 2021, pp. 1–8. doi: 10.1109/IJCNN52387.2021.9534453.
[5] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

[6] V. Mnih *et al.*, “Playing Atari with Deep Reinforcement Learning.” arXiv, 2013. doi: 10.48550/ARXIV.1312.5602.
[7] A. Millea, “Deep Reinforcement Learning for Trading—A Critical Survey,” *Data*, vol. 6, no. 11, 2021, doi: 10.3390/data6110119.
[8] Z. Zhang, S. Zohren, and S. Roberts, “Deep Reinforcement Learning for Trading,” *The Journal of Financial Data Science*, vol. 2, no. 2, pp. 25–40, 2020, doi: 10.3905/jfds.2020.1.030.
[9] Y. Li, W. Zheng, and Z. Zheng, “Deep Robust Reinforcement Learning for Practical Algorithmic Trading,” *IEEE Access*, vol. 7, pp. 108014–108022, 2019, doi: 10.1109/ACCESS.2019.2932789.
[10] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, “Deep Direct Reinforcement Learning for Financial Signal Representation and Trading,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 3, pp. 653–664, 2017, doi: 10.1109/TNNLS.2016.2522401.
[11] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, “A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning,” *Applied Intelligence*, vol. 51, no. 2, pp. 889–905, Feb. 2021, doi: 10.1007/s10489-020-01839-5.
[12] F. Rundo, “Deep LSTM with Reinforcement Learning Layer for Financial Trend Prediction in FX High Frequency Trading Systems,” *Applied Sciences*, vol. 9, no. 20, 2019, doi: 10.3390/app9204460.
[13] X.-Y. Liu, Z. Xiong, S. Zhong, H. Yang, and A. Walid, “Practical Deep Reinforcement Learning Approach for Stock Trading.” arXiv, 2018. doi: 10.48550/ARXIV.1811.07522.
[14] Y. Li, “Deep Reinforcement Learning: An Overview.” arXiv, 2017. doi: 10.48550/ARXIV.1701.07274.
[15] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996, doi: https://doi.org/10.1613/jair.301.
[16] V. Mnih *et al.*, “Asynchronous Methods for Deep Reinforcement Learning,” 2016, doi: 10.48550/ARXIV.1602.01783.
[17] L. Menkhoff, “The use of technical analysis by fund managers: International evidence,” *Journal of Banking & Finance*, vol. 34, no. 11, pp. 2573–2586, 2010, doi: https://doi.org/10.1016/j.jbankfin.2010.04.014.
[18] J. W. Lee, J. Park, J. O, J. Lee, and E. Hong, “A Multiagent Approach to Q-Learning for Daily Stock Trading,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 37, no. 6, pp. 864–877, 2007, doi: 10.1109/TSMCA.2007.904825.
[19] S. A. Badran and M. Rezghi, “An adaptive synchronization approach for weights of deep reinforcement learning.” arXiv, 2020. doi: 10.48550/ARXIV.2008.06973.
[20] https://github.com/junyoung-sim/quant