Stock Trading using Deep Reinforcement Learning

Dharun Shoban Ganesh
Faculty of Computing and Informatics
Multimedia University (MMU)
Cyberjaya, Malaysia
dharunshoban@gmail.com

Tong Gee Kok
Faculty of Computing and Informatics
Multimedia University (MMU)
Cyberjaya, Malaysia
gktong@mmu.edu.my

***Abstract*— Advanced computational techniques are crucial in navigating the complexities of the stock market, particularly as the financial technology industry grows. This paper explores the application of Deep Reinforcement Learning in optimizing stock trading strategies to adapt to volatile market conditions. The research aims to improve stock trading models' performance in both normal and bearish markets using DRL and technical indicators. Historical stock data from Amazon, Google, and Microsoft spanning 10 years is used to train and evaluate the model. Deep-Q Networks was employed with Long Short-Term Memory neural networks architecture. After data preprocessing and exploratory data analysis, the datasets are split in an 80:20 train-test ratio. The model's performance is evaluated using Cumulative Return, Sharpe Ratio, and Maximum Drawdown, followed by hyperparameter optimization via Bayesian Optimization. The model achieved a CR of 82.73%, SR of 0.93, and a MDD of -24.15%. This research contributes to the optimization of stock trading strategies through DRL and provides a structured model for addressing financial market complexities.**

Keywords—Deep Reinforcement Learning, DRL, Stock Trading, Reinforcement Learning, Deep Learning, Deep Q-Networks, Long Short-Term Memory (LSTM), Technical Indicators, Bayesian Optimization.

I. INTRODUCTION

Stock trading, involving buying and selling shares for profit, is fundamental in global financial markets like stock exchanges [1]. Accessible to both amateur and professional traders, it employs strategies including technical and fundamental analysis and algorithmic trading. Influenced by industry trends, company performance, economic indicators, and global events, stock trading is inherently challenging due to market volatility and noise [2]. Deep reinforcement learning (DRL), which combines deep learning (DL) and reinforcement learning (RL), has significantly advanced AI applications in stock trading by training agents to make decisions from environmental feedback. DRL models, such as Deep Q-Networks (DQN) and Double Deep Q-Networks (DDQN), process large datasets to identify patterns and trends, creating flexible, end-to-end trading systems [3], [4]. Despite challenges like data imbalance and market volatility, DRL's ability to handle complex, dynamic environments makes it valuable for stock trading, enhancing strategies and outcomes in a rapidly changing financial landscape [5].

Due to high data noise in stock trading, DRL agents need extensive training to learn optimal strategies, risking overfitting and poor performance [6]. To counter this, researchers like [7] suggest incorporating technical indicators into DRL models to reduce noise impact and improve performance. Stock trading, akin to a game with incomplete information, presents serialization decision challenges for traditional models [8]. Researchers, including [9], have enhanced decision-making in trading models by applying DRL algorithms such as DQN, Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and Soft Actor Critic (SAC), demonstrating superior performance. The dynamic and volatile nature of stock prices complicates developing accurate trading strategies [10]. To address this, researchers like [11] have used DRL algorithms such as DQN, AAC, and PG to improve profitability during volatile market conditions.

This research aims to enhance the performance of stock trading models by applying feature engineering techniques to obtain technical indicators used by DRL models. Additionally, the study aims to improve performance during normal stock market conditions by using DRL in simulated trading environments, evaluating performance with metrics such as Cumulative Return (CR) and Sharpe Ratio (SR). Furthermore, the research seeks to enhance model performance during bearish and volatile market conditions by employing DRL in simulated environments, using Maximum Drawdown (MDD) as a performance metric.

II. LITERATURE REVIEW

This section presents a complete overview of the literature related to stock trading, covering methodologies and techniques utilized in the field. It covers the concepts of stock trading, DRL, performance metrics, technical indicators, data sampling and hyperparameter tuning.

A. Stock Trading

The field of stock trading has been dominated by traditional methods and strategies. [12] evaluated multiple ML algorithms on large-scale stock datasets to assess daily trading performance with technical indicators. [13] integrated technical analysis indicators and fundamental analysis such as sentiment data from news articles to predict market trends. [14] conducted a systematic review of stock market prediction studies, categorizing them into technical, fundamental, and combined analyses based on dataset nature. [15] optimized investment strategies using ML and economic value-added (EVA) techniques. [16] examined ML algorithms for selecting stocks, focusing on predicting relative returns of individual stocks weekly.

B. Deep Reinforcement Learning

[17] introduced and explained RL concepts, their integration with DL, and explore their potential applications in cyber security. [18] presented a complete literature review of DRL applications in communications and networking. [19] provided a systematic review of DRL algorithms and their applications, highlighting challenges and prospects. [11] examined DRL's application in power systems, emphasizing its advantages in handling high-dimensional data. [20] offered an overview of DRL applications in economics, showcasing their efficiency in handling complex problems.

C. DRL in Stock Trading

[21] proposed data augmentation in RL for daily financial assets trading. [8] applied DRL to financial markets to enhance data processing, feature extraction, and transaction abilities. [4] used deep neural networks (DNN) and RL techniques to develop adaptive stock trading strategies. [22] developed and evaluate a novel DRL solution, specifically the Trading Deep Q-Network (TDQN) algorithm, for algorithmic trading. [23] combined DL and RL to create an automatic trading system utilizing DQN with LSTM networks. A new rule-based strategy for risk curiosity-driven learning and real-time decision making in automated trading was introduced by [24]. [25] proposed an Long Short-Term Memory with DDPG (LSTM-DDPG) model for making trading decisions with flexible positions. [10] developed an automated stock market trading decision support system using DRL, taking historical and future movements into consideration. [26] combined supervised learning and RL techniques to develop an optimal trading strategy. [6] introduced a novel Dual Action and Dual Environment with Deep Q-Networks (DADE-DQN) trading strategy.

D. Technical Indicators

Technical indicators are important tools in stock trading analysis. [16] emphasized integrating technical indicators with ML models for enhanced stock selection accuracy. Researchers have utilized various technical indicators, with [12] focusing on traditional ones like Simple Moving Average (SMA) and Exponential Moving Average (EMA) for analysing market momentum. [13] integrated technical indicators such as Moving Average Converge/Divergence (MACD) with sentiment data for trend prediction, while [14] explored a range including On-Balance Volume (OBV) and Bollinger Bands for market volatility and trend analysis. Advanced indicators like EVA are utilized by [15] to measure a company's economic profit.

E. Feature Selection

The process of identifying and selecting the most pertinent features from a dataset to enhance model performance, reduce dimensionality, and increase interpretability is known as feature selection. Researchers such as [6] have used Mutual Information with Univariate Analysis to evaluate the strength of the relationship between individual features and the target variables. On the other hand, [27] have utilized techniques such as Principal Component Analysis (PCA), Information Gain (IG), and Recursive Feature Elimination (RFE).

F. Data Sampling

Data sampling is a critical aspect of stock trading analysis. [12] utilized walk-forward analysis for training ML models. [21] used minute-candle data for training. However, most researchers such as [10], [22], [23] and more utilized the historical stock data and split it into training and testing sets for DRL models. These sampling methods ensure that the models are trained and evaluated on representative datasets.

G. Hyperparameter Optimization

Hyperparameter optimization is important for optimizing the performance of ML models, including models that incorporate neural network architectures. [12] specified hyperparameters for many ML algorithms, such as learning rates and kernel functions. [13] adjusted the learning rates, neuron counts in neural network layers, and regularization parameters. [15] constructed ML models, including the number of layers, units, and choice of activation, loss, and optimization functions. [28] utilized Bayesian Optimization (BO), focusing on parameters such as number of decision trees, maximum depth, learning rate, number of hidden layers, and neuron counts in models. [29] used Random Search tuning for their random forest model, tuning parameters such as number of trees, the maximum depth of the trees, the minimum samples split and samples leaf. [16] performed detailed hyperparameter tuning for each model, including the number of layers in neural networks, regularization techniques, and specific parameters for Random Forest and boosting. These hyperparameter tuning strategies ensure the models are finely tuned for optimal performance in stock trading applications.

H. Performance Metrics

The evaluation of stock trading strategies requires robust performance metrics. [12] used accuracy, precision, recall, Area Under Curve (AUC), winning rate, Annualized Return (AR), SR, and MDD. [13] measured accuracy, recall, AR, SR, and MDD. [14] use Mean Absolute Percentage Error (MAPE), Mean Squared Error (MSE), Mean Absolute Error (MAE), and Root Mean Square Error (RMSE). [15] utilized MAE and MSE. [16] considered classification accuracy, sensitivity, specificity, and Mean Squared Forecast Error (MSFE). [6] used multiple evaluation metrics such as SR, CR, AR, and MDD. These diverse performance metrics help assess the effectiveness of trading strategies and provide a basis for comparison.

III. THEORETICAL FRAMEWORK

With the in-depth literature review from the previous section as a foundation, this section aims to examine the concepts, methods, and applications of current stock trading methods.

A. Stock Trading Methods and Weaknesses

Stock trading has been a core activity of financial markets for centuries. It is essential to the global economy because many financial activities have an impact on the economic growth of multiple countries [30]. Global exchanges such as the New York Stock Exchange (NYSE) are used to conduct stock trading, but recently it can be done in the comfort of investors homes through electronic devices. However, due to the market's volatility and rapid changes, identifying profitable investments might be difficult.

Strategies in the field of stock trading have changed significantly throughout the years. For stock market strategies, there are three primary approaches that can be utilized: technical analysis, fundamental analysis, and ML [14]. This section analyses these various approaches, highlighting their benefits, drawbacks, and characteristics.

1) Traditional Stock Trading: For analyzing and predicting stock market behavior, fundamental and technical analysis are the most popular techniques [30]. Fundamental analysis evaluates a company's value through its financial statements, market position, and economic indicators, making it ideal for long-term investments due to its thorough understanding of business growth and sustainability. However, modeling stock fluctuations is challenging, making fundamental analysis less common [31]. Technical analysis, the most common method, focuses on statistical patterns in market movements like trade volumes and stock prices. It is useful for short-term trading decisions, providing insights into investor psychology and market trends, though it heavily relies on historical data, which may not always predict future market behavior.

2) Algorithmic Trading and Machine Learning: Algorithmic trading is an area of finance where trading decisions are automatically made by machines using mathematical formulas [31]. The stock trading industry has been transformed by the development of ML and algorithmic trading, which utilize data analysis and automated decision-making processes based on predetermined criteria. ML, a branch of AI, has significantly revolutionized the trading environment over the past decade due to large datasets for training complex models, advancements in ML algorithms, and increased processing power [16]. ML involves training algorithms to predict outcomes or make decisions based on data, applied in risk management, portfolio optimization, and stock price prediction. It excels at analyzing the nonlinear patterns often present in the stock market [15]. However, ML models risk overfitting, becoming overly tailored to historical data and failing to generalize to future market conditions.

B. Deep Reinforcement Learning

1) Reinforcement Learning: A type of ML that addresses sequential decision issues is called RL [32]. It consists of an environment, states, actions, reward, and agent. The agent interacts with the environment by taking actions, and its objective is to maximize cumulative reward. [32]. Unlike other ML fields, RL can handle situations where the rewards are delayed, making it optimal for complex and sequential decision-making tasks. However, because it needs to investigate and learn about a whole system, this learning method is not suitable for large-scale networks and takes a long time to get the optimal policy [18].

2) Deep Learning: DL is another type of ML that consists of multiple artificial neural networks (ANN) layers. Multiple layers of nodes make up DNNs, which use the layers to gradually extract higher-level features from unprocessed input data. Usually, DNNs are made up of multiple hidden layers organized in deep nested network architectures. DL's primary goal is to remove the need for human data structures by using artificial data learning [18]. This approach uses ANNs to convert a set of inputs into a set of corresponding outputs, which mimics the learning process of the human brain. The development of DL as an innovative approach that can get over RL's weaknesses has recently opened new possibilities for RL advancement [18].

3) Components of DRL: DRL combines DL's perceptual function with RL's ability for decision-making [11]. RL supervises the computational agent's decisions as it learns through trial and error. By using DNNs to train the learning process, DRL accelerates learning and enhances the efficiency of RL algorithms [18]. When presented with a specific scenario, DRL determines which course of action to take to maximize the benefit, and address decision optimization problems [19]. The agent acts and observes the outcomes of these actions to learn how to achieve a goal in a complex, uncertain environment. DRL algorithms can take in very large inputs and make decisions on what action to perform next to optimize and objective, such as CR. The architecture of a DRL model is visualized in Figure 1. The main components of a standard DRL model are formulated as follows:

- **Agent, A**: An entity that takes charge of making decisions.
- **Environment, E**: An area where the agent operates. In stock trading, the environment consists of factors such as historical price data, market trends etc.
- **State, S**: A state represents what information the model uses to represent the current market conditions and environment.
- **Action, A**: An action represents a move that the agent can take.
- **Reward, R**: The reward is feedback to the model in the form of incentives or penalties.
- **Policy, π**: A policy determines how an agent behaves at a specific point in time. It can be stochastic or deterministic.

- **Value Function, V**: The expected return or cumulative reward of being in a particular state or executing an action in that state under a given policy is estimated by the value function.

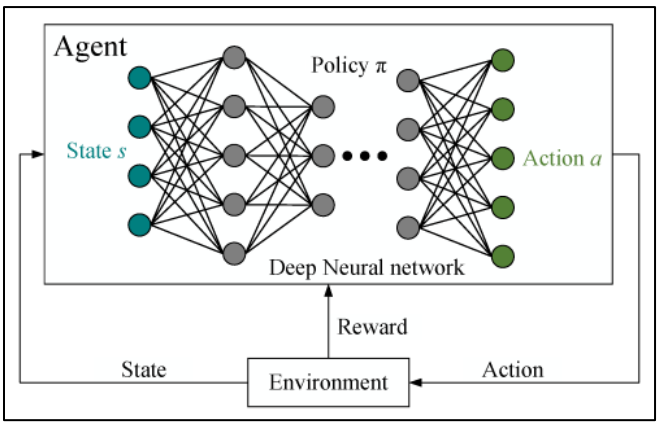

Fig. 1. Architecture of a DRL Model [33]

C. DRL in Stock Trading

For this research project, the DQN algorithm was chosen for modelling and analysis. Among the most effective and simplest DRL algorithms is the DQN algorithm [8]. Originally, it was developed for understanding control policies in video games. Recently, it has been adapted to the domain of stock trading due to their robustness in dealing with high-dimensional space data.

1) Markov Decision Process: Stock trading in the stock market is an interactive, stochastic process that can be defined as a Markov Decision Process (MDP) [34]. MDP is a mathematical framework used to simulate decision-making scenarios, allowing an agent to interact with the environment and achieve specific goals [6]. It is applied in various fields, including finance and economics, to evaluate and enhance decision-making processes [6].

MDP is described by a set where *S* is the state space, containing all states in the environment; *A* is the action space, representing all possible actions the agent can take, T is the transfer probability, indicating the likelihood that an action taken in a certain state will result in a state transfer; and *R* is the reward received after the action [35]. The discount factor γ, with values $\gamma \in \{0, 1\}$, balances long-term and immediate rewards [10]. The set *M* is denoted as *M* = {*S, A, T, R, γ}:*

- **S = {s1, s2, s3}**: The state space includes all possible states of the stock trading environment, such as price levels and trading volumes.
- **A = {a1, a2, a3}:** he action space includes all possible actions the trading agent can take in any given state.
- **T = P (s' | s, a):** The transfer probability represents the likelihood that taking an action a in state s will result in a new state s'.
- **R = R (s, a, s'):** The immediate reward received when transitioning from state s to s' after taking action a. Rewards can be gains, losses, or other financial metrics.
- **γ = {0,1}**: The discount factor determines the trade-off between short-term and long-term rewards. A discount factor of 0 indicates a focus on immediate rewards, while a value of 1 indicates a focus on future rewards.

2) Deep Q-Learning: Q-learning is a model-free method that solves the problem by using trial and error in a reward-based system rather than a transition probability distribution [36]. It is an approach for using an action-value function to determine the suitable action for an agent [10]. This function, also known as the Q-function, determines the expected cumulative rewards that an agent would receive if they performed a certain action, a, in a particular state, s. Updating these Q-values iteratively in response to the agents' experiences is the key concept of Q-learning. The Q-function is denoted as follows:

$$Q(s,a) = r(s,a) + \gamma * (max)_a\, Q(s',a) \qquad (1)$$

where Q (s, a) is the Q-value of taking an action in a state, r (s, a) is the immediate reward received after taking an action in state, γ is the discount factor and $(max)_a\, Q(s', a)$ is the maximum predicted Q-value for the next state considering all possible actions (Hariharan and Anand, 2022).

By integrating randomness into the decision-making process, the Bellman equation, which is under MDP, is used to update the Q-values in the Q-table [36]. It is important to determine how important every action is in a certain situation. The Bellman equation calculates the sum of the immediate reward for an action and the maximum expected future rewards, discounted by a certain factor. This will ensure that it considers both the short- and long-term effects of an action, which in turn impacts the neural network's transition probabilities [36]. The formula for the Bellman equation is denoted as follows:

$$V(s) = (max)_a\, [R(s,a) + \gamma \sum_{s'} P(s,a,s') * V(s')] \qquad (2)$$

where V(s) denotes the value of the action, R(s, a) is the reward for the action taken in the state, V(s') is the value of the future state and P(s, a, s') is the probability that the action taken in the state leads to the future state [36].

DQN is an advanced RL agent that maps the relationships between states and actions using DNNs, which is like the Q-Table in Q-learning [10]. The basic concept of DQN is a technique based on Q-learning that uses DNN to approximate the Q-function [37]. Similar to Q-learning, the DNN-based agent can observe the sequence of states from the environment, act upon them, and receive rewards based on

that action. The Bellman equation is used to update the DNN-based agent's weights [10]. Specifically, each action an agent takes on a state results in the production of Q-values. DNN's primary goal is to learn and update its parameters in response to the rewards and penalties that it receives. The difference between Q-learning and Deep Q-learning is visualized in Figure 2.

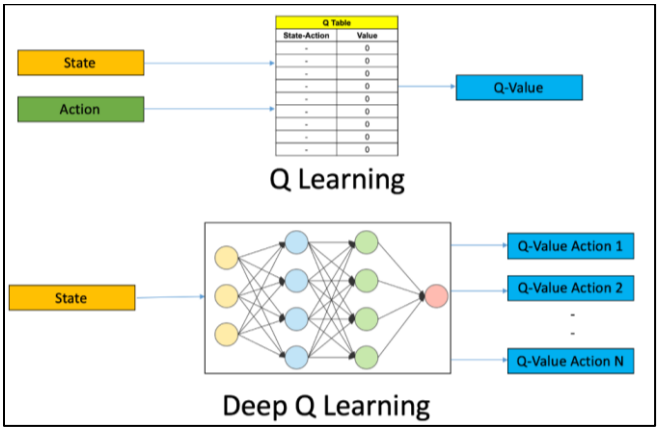

Fig. 2. Pictorial Representation of Q-Learning and Deep Q-Learning [38]

3) LSTM Neural Networks: The LSTM neural network architecture, a unique type of RNN developed by Hochreiter and Schmidhuber, has gained popularity in the financial industry for its ability to consistently forecast time series data [15], [16]. An LSTM neural network consists of an input layer, one or more hidden layers, and an output layer [16]. The input and output layers contain regular neurons, with the number of input features (state space) equating to the number of neurons in the input layer. The hidden layer comprises memory-controlling cells with three control gates: a forget gate to discard unnecessary data, an output gate to determine the data used as output from cell memory, and an input gate to decide which data is added to the cell memory [16]. The DQN approach in RL uses two networks: a Q-network that approximates the Q-function and a target network that approximates the target value required to update the Q-network. The target network, a copy of the main neural network with fixed parameters, is periodically updated to prevent the overestimation of Q-values [37].

4) Reward Function: The reward function, R is implemented directly in the environment. It calculates the immediate reward received after transitioning from one state to another due to an action. In stock trading, this could be the profit or loss from selling stocks, the change in portfolio value, or any other metric to indicate the agent's performance. When implementing RL techniques in trading strategies, the function's design is important [4]. The reward signal is crucial for training the DQN model, as it guides the agent to learn which actions are beneficial in different states.

$$R = \frac{Selling\ Price - Buying\ Price}{Buying\ Price} \quad (3)$$

where Selling Price is the price when the stock is sold and Buying Price when the stock is bought [4].

5) Exploration vs Exploitation: For selecting actions, the DQN algorithm employs an epsilon-greedy, also known as ε-greedy. It is the most common exploration-exploitation strategy in DRL algorithms, particularly DQN. In the epsilon-greedy policy, Q values are determined using the model rather than selecting the action based on policy [23]. If a given condition is satisfied, the agent will perform a random action. The algorithm chooses an action (exploration) at random with probability ε at each step, or it chooses the best action (exploitation) based on the current estimated Q-value function with a probability of 1 – ε.

6) Experience Replay: An important component in the DQN model's training process is the replay memory buffer. In experience replay, past states have an impact on all states, actions, and rewards [39]. It stores and randomly samples past experiences (transitions) from the agent's interaction with the environment. Correlations are eliminated by the experience replay, which buffers the event and extracts the learning data at random [39].

D. Technical Indicators

1) Simple Moving Average: The term "arithmetic moving average," or SMA, is derived from summing up the closing prices of the stock across a number of time periods and dividing the sum by the total number of time periods. The indicator's straightforward approach contributes valuable insights into the longer-term trends within historical stock data. The SMA formula is described below:

$$SMA = \frac{X_1 + X_2 + X_3 + \cdots + X_n}{n} \quad (4)$$

where n is the number of total periods, and X_n is values (prices) at period n [40].

2) Exponential Moving Average: EMA is a type of moving average which gives more weight to the most recent price and less weight to historical price data. A longer EMA period can be used as a strong support signal, while a shorter EMA length can be used as a short-term trending signal. In EMA, the weights of earlier data are correspondingly decreased as the most recent data is assigned a higher weight. The most recent price is given more weight when the EMA's term is shorter [41]. The EMA formula is described below:

$$EMA_n = \propto \cdot X_n + (1 - \propto)\ EMA_{n-1} \quad (5)$$

where is $\propto = \frac{2}{n+1}$ the smoothing factor, X_n is the price of the stock at time n, and EMA_n is the EMA at time n [42].

3) Moving Average Convergence/Divergence: The combination and division of the long-term index average index and the short-term index average index by MACD represents the present bearish situation and possible development tendency of stock prices [26]. The trend-following momentum indicator calculates the difference between two price moving averages (the 26-day EMA minus the 12-day EMA) to display the relationship between them. Next, a nine-day EMA is placed on top of the MACD to serve as a trigger for buy and sell signals. This is known as the "signal line" [43]. The formulas are described as follows:

$$MACD = 12\ day\ EMA - 26\ day\ EMA \quad (6)$$

$$Signal\ Line = 9\ day\ EMA\ of\ MACD\ Line \quad (7)$$

E. Feature Selection

Recursive Feature Elimination: By eliminating the least relevant features, a method known as RFE is used to determine which features are most important in forecasting a target [27]. It starts by using all of the available features to train a model, and then it determines each feature's relevance by giving the model weights or coefficients. In an iterative process, less important features are progressively removed until the necessary number of features, or a certain threshold is reached. RFE often evaluates the model's performance at each iteration using evaluation metrics like accuracy or precision.

E. Data Partitioning

Sequential Train-Test Split: The train-test split sampling is a popular data sampling technique used in ML. It is an important step in preparing the data for building and evaluating models. This method involved partitioning the historical stock price dataset into two distinct subsets. The two subsets consist of a training set and a test set. The DRL model learns the complex patterns and dynamics in the historical stock data from the training set, and is evaluated using the test set.

F. Hyperparameter Optimization

Bayesian Optimization: Grid and random search strategies used to be the main methods for hyperparameter optimization, but they are ineffective since they consistently identify local maxima as global maxima [28]. BO is a sequential design strategy for hyperparameter optimization, especially in DNNs. It is used to fine-tune the hyperparameters of the model architecture. BO utilizes a probabilistic model to map the hyperparameters to a probability distribution of possible objective function values and uses the model to select the most promising hyperparameters to evaluate next. It is particularly useful for DRL models due to its efficiency in exploring the hyperparameter space with a limited number of evaluations, which is important as the computational expense of training DRL models can be high. It also systematically improves the search based on previous evaluations, which leads to superior model performance with less trial and errors compared to Grid Search or Random Search techniques. Unlike random search, which scans the parameter space blindly, this method encourages using intelligence to select the next set of parameters that will improve the performance of the model (Tran M et al., 2023).

G. Performance Metrics

1) Cumulative Return: The CR metric is also defined as the difference between the initial and final investment amounts [26]. The CR metric is calculated as follows:

$$CR = \frac{Final\ Value - Initial\ Value}{Initial\ Value} \times 100 \quad (8)$$

where the initial value is the initial value of the portfolio, and final value is the final value of the portfolio [44].

2) Sharpe Ratio: The average return over the risk-free rate per unit of volatility or total risk, represented by the standard deviation of the portfolio, is known as the SR metric [13]. It is formulated as follows:

$$SR = \frac{Rp - Rf}{\sigma p} \quad (9)$$

where Rp is the return, Rf is the risk-free rate, and σp is the standard deviation of excess returns [21].

3) Maximum Drawdown: MDD, which measures the largest amount of money a fund has lost over a period of time [26], is important in evaluating the trading strategy's risk features. It is formulated as follows:

$$MDD = \frac{Peak\ Value - Valley\ Value}{Peak\ Value} \times 100 \quad (10)$$

where the peak value is the highest value before the largest drop and valley value is the value before a new high value is established [34], [45].

IV. RESEARCH METHODOLOGY

In this section, the methodology followed in the research is described. The section explains the approach taken to develop, implement, and assess the model that utilizes DRL for stock trading.

A. Data Collection

The datasets used in this study consist of historical stock data for AMZN, GOOGL, MSFT. The datasets span from 1st January 2013 to 29th December 2023 (10 years) to ensure that the size of the datasets is sufficient for analysis. These datasets

were obtained from Yahoo Finance and include daily stock prices featuring the date, open, high, low, close, adjusted close, and volume.

B. Data Preprocessing and Partitioning

1) Addressing Duplicates and Zero/Null Values: The datasets were checked for duplicate rows and zero or null values to ensure quality and reliability of the datasets.

2) Remove Irrelevant Features: Features that did not contribute to the model evaluation such as 'Date' and 'Adj. Close' were removed from the datasets to reduce dimensionality and overfitting.

3) Feature Engineering: Additional features based on technical indicators were incorporated into the datasets which include a 20-day SMA, 12-day EMA, MACD indicator utilizing a 12-day short-term EMA and 26-day long-term EMA, and its 9-day signal line. The rows with missing values produced after feature engineering were removed.

4) Feature Selection: Feature selection was employed through the RFE method to choose the most relevant features. The top eight features ranked by RFE method are kept, while the rest were removed.

5) Data Partitioning: The preprocessed datasets were partitioned into an 80:20 sequential train-test split to train and evaluate the DRL model performance on testing data.

C. DQN-LSTM Algorithm

1) Trading Environment: The components of the trading environment are defined in Table I.

TABLE I. COMPONENTS OF THE TRADING ENVIRONMENT

Components	Description
State Space (S)	Open, High, Low, Close, SMA, EMA, MACD, Signal Line
Action Space (A)	Sell (0), Hold (1), Buy (2)
Reward Function (R)	$Selling\ Price - Buying\ Price$
Transfer Probability (T)	Implicitly learned by the model.
Discount Factor (γ)	0.95

2) Model Construction: After defining the trading environment, the DQN model integrated with LSTM neural networks is constructed. This integration allows the model to leverage long-term dependencies crucial for understanding market trends. The DQN agent makes decisions based on current state inputs, including stock price features and indicators, processed through the LSTM to capture temporal dependencies. The LSTM consists of input, hidden, and output layers, with the output determining the trading action. The agent follows an epsilon-greedy policy for decision-making, balancing exploration and exploitation. Adaptive learning rate methods like the Adam optimizer and regularization techniques such as dropout and batch normalization are used to optimize learning and generalization. The trading environment includes a state space, action space, transfer probability, reward function, and discount factor. A replay memory buffer stores experience tuples (state, action, reward, next state) to break observation correlations and enhance learning stability. These experiences are sampled and processed through the LSTM, refining the agent's policy to maximize expected cumulative rewards and optimize trading decisions. The model architecture construction is illustrated in Figure 3.

3) Model Training: A training loop is designed to enable the model to learn and adapt to stock market complexities through RL. This process develops effective trading strategies by coordinating the agent's interactions with the trading environment. The training loop starts with initializing key parameters and defining the trading environment using a pre-processed dataset. The state and action spaces are configured, and the agent is instantiated with these sizes. The training process is guided by parameters such as batch size and episodes. For each episode, the loop resets the environment, acquires the initial state, and reshapes it to match the LSTM's input format. The agent then explores and learns by selecting actions based on its current state, executing them, and receiving the next state, reward, and termination signal from the environment. These experiences are stored in the agent's memory. During the training loop, the agent periodically engages in experience replay and model learning. When the memory buffer exceeds the batch size, the agent performs experience replay, using past experiences to update its neural network weights. The loop continues for the defined number of episodes, iteratively improving the model's performance in the simulated trading environment.

D. Hyperparameter Optimization

The objective of BO in this research is to efficiently tune the hyperparameters of the DRL model to maximize total profit over a fixed number of trading episodes. BO starts with defining the hyperparameter search space and uses Gaussian Processes (GP) to predict performance metrics. The optimization loop iteratively selects, evaluates, and updates hyperparameters based on the acquisition function's strategy, training and validating the model until the stopping criteria, typically the maximum number of iterations, is met. This process identifies the optimal hyperparameters for the highest estimated performance on the validation set.

V. EVALUATION OF FINDINGS

In this section, the results of this study are presented, focusing on the performance of the model using evaluation metrics. The performance is evaluated on three datasets, AMZN, GOOGL, and MSFT. The performance of the model with and without BO is compared, and then the model with the higher performance is compared to past work by researchers.

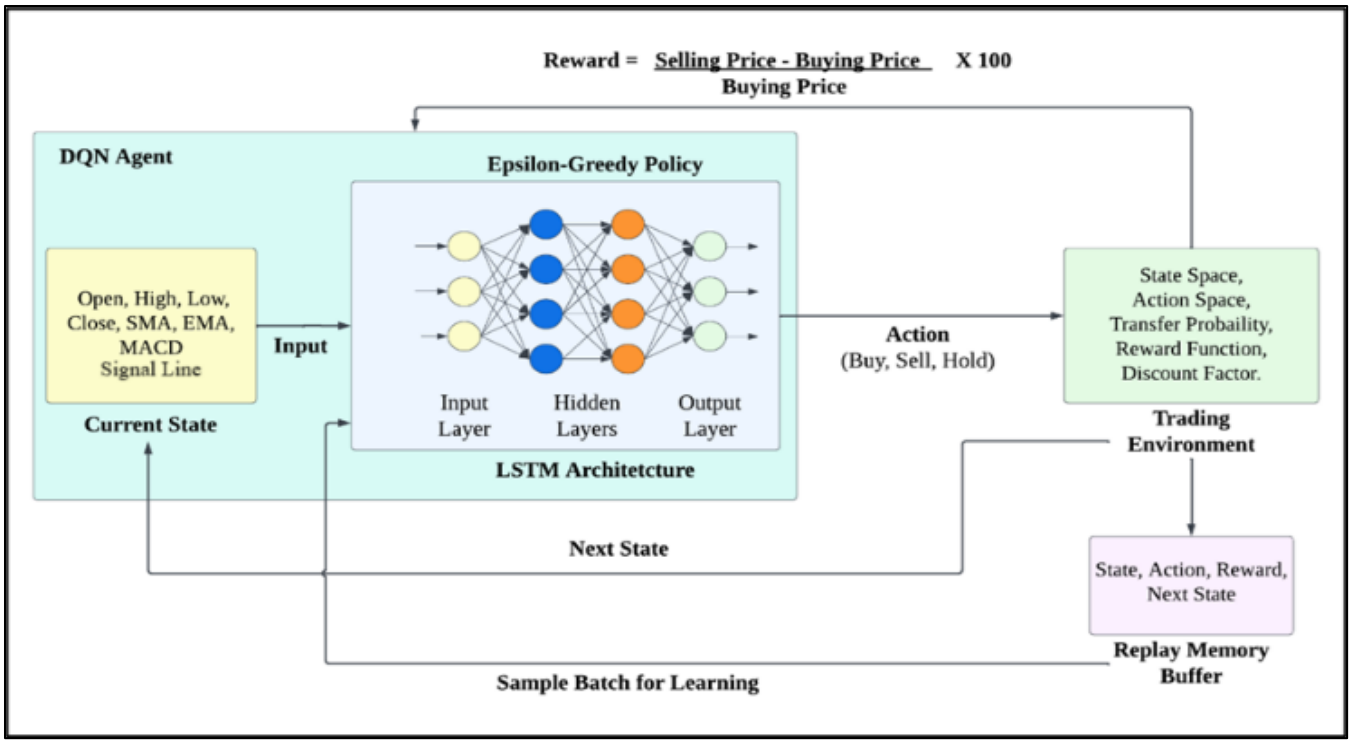

Fig. 3. Architecture of the Trading Model

A. Model Evaluation Results

1) DQN-LSTM: Out of all three datasets, the model achieved the best performance results on the MSFT dataset. With an initial capital of $100, the average final capital stands at $182.73, showing an average total profit of $82.73. The average CR is 82.73%, indicating a notable gain on the initial investment. The SR averages at 0.93, suggesting a favourable trade-off between risk and return. The average MDD is -24.15%, indicating a low portfolio decline during adverse periods. The results are displayed in Table II.

TABLE II. PERFORMANCE METRICS RESULTS WITHOUT BO

Dataset	Initial Capital	Final Capital	Total Profits	CR	SR	MDD
AMZN	100	148.45	48.45	48.45	0.87	-31.76
GOOGL	100	128.75	28.75	28.75	0.53	-20.42
MSFT	100	270.99	170.99	170.99	1.40	-20.28
Average			**82.73**	**82.73**	**0.93**	**-24.15**

2) DQN-LSTM with BO: Out of all three datasets, the tuned model once again achieved the best performance results on the MSFT dataset. With an initial capital of $100, the average final capital stands at $168.81, showing an average total profit of $68.81. The average CR is 68.81%, indicating a notable gain on the initial investment. The SR averages at 0.71, suggesting a reasonable trade-off between risk and return. Moreover, the MDD averages at –29.40%, indicating reduced portfolio decline during adverse market conditions. The results are displayed in Table III.

TABLE III. PERFORMANCE METRICS RESULTS WITH BO

Dataset	Initial Capital	Final Capital	Total Profits	CR	SR	MDD
AMZN	100	134.05	34.05	34.05	0.55	-25.82
GOOGL	100	133.29	33.29	33.29	0.55	-25.93
MSFT	100	239.08	139.08	139.08	1.02	-36.45
Average			**68.81**	**68.81**	**0.71**	**-29.40**

B. Comparison with Past Works

The performance comparison of the current DQN-LSTM model with past research, highlights significant achievements and areas for improvement. The current model's SR of 0.93 indicates a high level of risk-adjusted return, outperforming models like DQN-GRU (0.69) [10] and LSTM-DDPG (0.30, 0.74) [25], [26]. However, it falls short of the TDQN model (1.48) [21] and the DADE-DQN model (1.20) [6]. In terms of MDD, the current model records -24.15%, performing better than the LSTM-DDPG model (-30.91%) and DQN-GRU model (-34.20%), but less favourably compared to the LSTM-DDPG model (-14.30%) and the DADE-DQN model (-13.95%). This indicates a competitive performance with room for further enhancements in risk management. The comparisons are displayed in Table IV.

TABLE IV. RESULTS COMPARISON WITH PAST WORK

Author	Model Used	SR	MDD (%)
Théate and Ernst [21]	TDQN	1.48	-17.31
Jia Z et al. [25]	LSTM-DDPG	0.30	-14.30
Ansari Y et al. [10]	DQN-GRU	0.69	-34.20
Fu K et al. [26]	LSTM-DDPG	0.74	-30.91
Huang Y et al. [6]	DADE-DQN	1.20	-13.95
Current Model (2024)	**DQN-LSTM**	**0.93**	**-24.15**

VI. CONCLUSION

A. Research Contribution

The primary contribution of this research is the development of a DQN-LSTM model for stock trading, effectively combining the strengths of DQN and LSTM networks to handle sequential data and temporal dependencies. This integration allows the model to capture and utilize patterns in financial time series data, advancing the application of DRL in financial trading. A thorough evaluation of the DQN-LSTM model's performance, with and without BO, reveals the impact of hyperparameter tuning on model effectiveness. While the unoptimized model achieved higher returns and a better risk-return trade-off, the optimized model highlighted the challenges of fine-tuning in volatile markets, underscoring the importance of careful hyperparameter selection. The integration of technical indicators into the stock datasets provided additional dimensions of data to enhance the model's performance. The study uses performance metrics like SR and MDD for a balanced evaluation of returns and risk management. By benchmarking against state-of-the-art DRL models, the research validates the model's strengths and identifies areas for improvement. The findings emphasize the importance of sufficient testing and validation to ensure generalization across different market conditions, providing practical insights for traders and practitioners aiming to apply advanced RL models in real-world trading scenarios.

B. Future Work

This project can be improved in several ways. Implementing additional technical indicators such as Bollinger Bands, RSI, and Stochastic Oscillator could provide the model with more diverse signals, potentially enhancing performance. Incorporating external factors like macroeconomic indicators (GDP growth rates, interest rates), political event data (election outcomes, policy changes), and market sentiment analysis from news and social media could further improve the model's decision-making process. Increasing the number of training and testing episodes would enhance learning and robustness, leading to better generalization. Additionally, more iterations in the BO process could refine hyperparameters more effectively, resulting in a more optimized, accurate, and efficient model. These improvements could develop a more sophisticated and robust DRL model, capable of delivering better trading strategies and handling the complexities of financial markets.

REFERENCES

[1] Hayes, A. (2023, March 1). What Is Stock Trading? Investopedia. https://www.investopedia.com/what-is-stock-trading-7109934#:~:text=Stock%20trading%20involves%20buying%20and%20selling%20shares%20of%20publicly%20traded

[2] Li, Y. (2022). Stock quantitative prediction analysis method based on deep learning transformer self-attention mechanism. ACM International Conference Proceeding Series. https://doi.org/10.1145/3558819.3565216

[3] Chen, L., & Gao, Q. (2019). Application of deep reinforcement learning on automated stock trading. Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS, 2019-October. https://doi.org/10.1109/ICSESS47205.2019.9040728

[4] Wu, X., Chen, H., Wang, J., Troiano, L., Loia, V., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. Information Sciences, 538. https://doi.org/10.1016/j.ins.2020.05.066

[5] Sahu, S. K., Mokhade, A., & Bokde, N. D. (2023). An Overview of Machine Learning, Deep Learning, and Reinforcement Learning-Based Techniques in Quantitative Finance: Recent Progress and Challenges. In Applied Sciences (Switzerland) (Vol. 13, Issue 3). https://doi.org/10.3390/app13031956

[6] Huang, Y., Lu, X., Zhou, C., & Song, Y. (2023). DADE-DQN: Dual Action and Dual Environment Deep Q-Network for Enhancing Stock Trading Strategy. Mathematics, 11(17). https://doi.org/10.3390/math11173626

[7] Li, F., Wang, Z., & Zhou, P. (2022). Ensemble Investment Strategies Based on Reinforcement Learning. Scientific Programming, 2022. https://doi.org/10.1155/2022/7648810

[8] Li, Y., Ni, P., & Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. Computing, 102(6). https://doi.org/10.1007/s00607-019-00773-w

[9] Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. (2021). FinRL: A Deep Reinforcement Learning Library for Automated Stock Trading in Quantitative Finance. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3737859

[10] Ansari, Y., Yasmin, S., Naz, S., Zaffar, H., Ali, Z., Moon, J., & Rho, S. (2022). A Deep Reinforcement Learning-Based Decision Support System for Automated Stock Market Trading. IEEE Access, 10. https://doi.org/10.1109/ACCESS.2022.3226629

[11] Zhang, Z., Zhang, D., & Qiu, R. C. (2020). Deep reinforcement learning for power system applications: An overview. In CSEE Journal of Power and Energy Systems (Vol. 6, Issue 1). https://doi.org/10.17775/CSEEJPES.2019.00920

[12] Lv, D., Yuan, S., Li, M., & Xiang, Y. (2019). An Empirical Study of Machine Learning Algorithms for Stock Daily Trading Strategy. Mathematical Problems in Engineering, 2019. https://doi.org/10.1155/2019/7816154

[13] Picasso, A., Merello, S., Ma, Y., Oneto, L., & Cambria, E. (2019). Technical analysis and sentiment embeddings for market trend prediction. Expert Systems with Applications, 135. https://doi.org/10.1016/j.eswa.2019.06.014

[14] Nti, I. K., Adekoya, A. F., & Weyori, B. A. (2020). A systematic review of fundamental and technical analysis of stock market predictions. Artificial Intelligence Review, 53(4). https://doi.org/10.1007/s10462-019-09754-z

[15] Li, J., Wang, X., Ahmad, S., Huang, X., & Khan, Y. A. (2023). Optimization of investment strategies through machine learning. Heliyon, 9(5). https://doi.org/10.1016/j.heliyon.2023.e16155

[16] Wolff, D., & Echterling, F. (2024). Stock picking with machine learning. Journal of Forecasting, 43(1). https://doi.org/10.1002/for.3021

[17] Nguyen, T. T., & Reddi, V. J. (2023). Deep Reinforcement Learning for Cyber Security. IEEE Transactions on Neural Networks and Learning Systems, 34(8). https://doi.org/10.1109/TNNLS.2021.3121870

[18] Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., & Kim, D. I. (2019). Applications of Deep Reinforcement Learning in Communications and Networking: A Survey. In IEEE Communications Surveys and Tutorials (Vol. 21, Issue 4). https://doi.org/10.1109/COMST.2019.2916583

[19] Wang, H. nan, Liu, N., Zhang, Y. yun, Feng, D. wei, Huang, F., Li, D. sheng, & Zhang, Y. ming. (2020). Deep reinforcement learning: a survey. In Frontiers of Information Technology and Electronic Engineering (Vol. 21, Issue 12). https://doi.org/10.1631/FITEE.1900533

[20] Mosavi, A., Faghan, Y., Ghamisi, P., Duan, P., Ardabili, S. F., Salwana, E., & Band, S. S. (2020). Comprehensive review of deep reinforcement learning methods and applications in economics. Mathematics, 8(10). https://doi.org/10.3390/MATH8101640

[21] Yuan, Y., Wen, W., & Yang, J. (2020). Using data augmentation based reinforcement learning for daily stock trading. Electronics (Switzerland), 9(9). https://doi.org/10.3390/electronics9091384

[22] Théate, T., & Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. Expert Systems with Applications, 173. https://doi.org/10.1016/j.eswa.2021.114632

[23] Cheng, L. C., Huang, Y. H., Hsieh, M. H., & Wu, M. E. (2021). A novel trading strategy framework based on reinforcement deep learning for financial market predictions. Mathematics, 9(23). https://doi.org/10.3390/math9233094

[24] Hirchoua, B., Ouhbi, B., & Frikh, B. (2021). Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. Expert Systems with Applications, 170. https://doi.org/10.1016/j.eswa.2020.114553

[25] Jia, Z., Gao, Q., & Peng, X. (2021). LSTM-DDPG for trading with variable positions. Sensors, 21(19). https://doi.org/10.3390/s21196571

[26] Fu, K., Yu, Y., & Li, B. (2023). Multi-Feature Supervised Reinforcement Learning for Stock Trading. IEEE Access, 11. https://doi.org/10.1109/ACCESS.2023.3298821

[27] Paramita, A. S., & Winata, S. V. (2023). A Comparative Study of Feature Selection Techniques in Machine Learning for Predicting Stock Market Trends. Journal of Applied Data Sciences, 4(3). https://doi.org/10.47738/jads.v4i3.99

[28] Liu, W., Suzuki, Y., & Du, S. (2023). Forecasting the Stock Price of Listed Innovative SMEs Using Machine Learning Methods Based on Bayesian optimization: Evidence from China. Computational Economics. https://doi.org/10.1007/s10614-023-10393-4

[29] Yin, L., Li, B., Li, P., & Zhang, R. (2023). Research on stock trend prediction method based on optimized random forest. CAAI Transactions on Intelligence Technology, 8(1). https://doi.org/10.1049/cit2.12067

[30] Kumar, G., Jain, S., & Singh, U. P. (2021). Stock Market Forecasting Using Computational Intelligence: A Survey. Archives of Computational Methods in Engineering, 28(3). https://doi.org/10.1007/s11831-020-09413-5

[31] Bustos, O., & Pomares-Quimbaya, A. (2020). Stock market movement forecast: A Systematic review. In Expert Systems with Applications (Vol. 156). https://doi.org/10.1016/j.eswa.2020.113464

[32] Jonsson, A. (2019). Deep reinforcement learning in medicine. In Kidney Diseases (Vol. 5, Issue 1). https://doi.org/10.1159/000492670

[33] Yi, J., & Liu, X. (2023). Deep Reinforcement Learning for Intelligent Penetration Testing Path Design. Applied Sciences (Switzerland), 13(16). https://doi.org/10.3390/app13169467

[34] Yu, X., Wu, W., Liao, X., & Han, Y. (2023). Dynamic stock-decision ensemble strategy based on deep reinforcement learning. Applied Intelligence, 53(2). https://doi.org/10.1007/s10489-022-03606-0

[35] Li, Y., Liu, P., & Wang, Z. (2022). Stock Trading Strategies Based on Deep Reinforcement Learning. Scientific Programming, 2022. https://doi.org/10.1155/2022/4698656

[36] Hariharan, N., & Anand, P. G. (2022). A Brief Study of Deep Reinforcement Learning with Epsilon-Greedy Exploration. International Journal of Computing and Digital Systems, 11(1). https://doi.org/10.12785/ijcds/110144

[37] Park, H., Sim, M. K., & Choi, D. G. (2020). An intelligent financial portfolio trading strategy using deep Q-learning. Expert Systems with Applications, 158. https://doi.org/10.1016/j.eswa.2020.113573

[38] Choudhary A. (2023, August 21). Introduction to Deep Q-Learning for Reinforcement Learning (in Python). Analytics Vidhya. https://www.analyticsvidhya.com/blog/2019/04/introduction-deep-q-learning-Python/

[39] Jang, B., Kim, M., Harerimana, G., & Kim, J. W. (2019). Q-Learning Algorithms: A Comprehensive Classification and Applications. IEEE Access, 7. https://doi.org/10.1109/ACCESS.2019.2941229

[40] Prami Swari, M. H., Susila Handika, I. P., & Susila Satwika, I. K. (2021). Comparison of Simple Moving Average, Single and Modified Single Exponential Smoothing. Proceedings - 2021 IEEE 7th Information Technology International Seminar, ITIS 2021. https://doi.org/10.1109/ITIS53497.2021.9791516

[41] Khand, S., Anand, V., Qureshi, M. N., & Katper, N. K. (2019). The Performance of Exponential Moving Average, Moving Average Convergence-Divergence, Relative Strength Index and Momentum Trading Rules in the Pakistan Stock Market. Indian Journal of Science and Technology, 12(26), 1–22. https://doi.org/10.17485/ijst/2019/v12i26/145117

[42] Vergura, S. (2020). Bollinger bands based on exponential moving average for statistical monitoring of multi-array photovoltaic systems. Energies, 13(15). https://doi.org/10.3390/en13153992

[43] Pramudya, R., & Ichsani, S. (2020). Efficiency of Technical Analysis for the Stock Trading. International Journal of Finance & Banking Studies, 9(1).

[44] Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep Reinforcement Learning for Automated Stock Trading: An Ensemble Strategy. SSRN Electronic Journal. https://doi.org/10.2139/ssrn.3690996

[45] Carta, S., Corriga, A., Ferreira, A., Podda, A. S., & Recupero, D. R. (2021). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. Applied Intelligence, 51(2). https://doi.org/10.1007/s10489-020-01839-5