Journal of Innovations in Computer Science and Engineering
Shahid Beheshti Unversity

January 2024, Volume 1, Issue 2

Dynamic Stock Trading with Gated-Convolutional-Attention Neural Network and Deep Reinforcement Learning

Mahdi Shahbazi Khojasteh ORCID: 0009-0007-6262-9460,
Mohammad Mahdi Setak ORCID: 0009-0000-3943-3274,
Armin Salimi-Badr ORCID: 0000-0001-6613-7921,✉
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran,
email: m.shahbazikhojasteh@mail.sbu.ac.ir
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran,
email: m.setak@mail.sbu.ac.ir
Faculty of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran,
email: a_salimibadr@sbu.ac.ir.

Abstract

The stock market plays an imperative role in the entire financial market. The intricate and multifaceted nature of the stock market poses a challenge for investors seeking to establish a reliable and profitable trading approach. This paper aims to address this issue by leveraging two methodologies based on Deep Reinforcement Learning (DRL), namely Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG), incorporating Convolutional Neural Network (CNN) and Gate Recurrent Unit (GRU) architectures, along with an attention mechanism to boost the decision-making based on time-series stock data. This adaptation enables the model to focus on essential features and time periods within the stock data, leading to more successful and higher-quality trading choices. Following extensive experimentation and analysis, our proposed RL-based trading demonstrates improved accuracy and profitability compared to similar approaches. The proposed methodology strives to offer investors a dependable and lucrative trading strategy, ultimately leading to a more prosperous and efficient stock trading experience.

Keywords: Stock markets, trading strategies, gated recurrent unit (GRU), deep reinforcement learning (DRL), deep q-network (DQN), deep deterministic policy gradient (DDPG).

1. Introduction

The financial markets have always been a hub of technical innovation for traders seeking a competitive edge. Using conventional analytical approaches to analyze stock market trading strategies undergoes difficulties when applied to economics and finance. Adapting to the complex and constantly evolving financial landscape proves difficult using these methodologies that rely on human intuition and rule-based systems. These approaches lack credibility in substantiating their accuracy and are influenced by the quality of analysts. With the introduction of computational methods in finance, much research has focused on applying Artificial Intelligence (AI) to financial investments in the stock market [1]. Consequently, the financial industry has undergone a notable change due to the increased usage of Machine Learning (ML) methods and the advancement of computing power. ML algorithms can scrutinize market information, identify trends and patterns, and predict potential buying and selling opportunities that may go unnoticed by humans [2], [3][4], [5], [6]. However, it is insufficient to rely solely on time-series forecasting models for accurate stock price predictions [7]. The current trend in Deep Learning (DL) for financial time series revolves around improving hybrid approaches by integrating various techniques and leveraging the most effective available algorithms [8].

The stock market is renowned for its volatile fluctuations and numerous intervening factors [9]. Creating a market model to predict stock prices is inherently challenging due to the uncertainty and risk associated with estimating stock values [10]. Many studies [11][12], [13], [14] have primarily concentrated on employing supervised learning methodologies to develop adaptive automated trading systems tailored to meet investors' objectives. These methods involve training predictive models on past data to anticipate market trends. Despite their widespread use, these techniques are fraught with limitations that often result in less-than-ideal outcomes [15]. Additionally, supervised learning models often struggle with the complexities of sequential decision-making, as they prioritize minimizing prediction error without considering the associated risks [16]. Consequently, there is a growing need for alternative approaches to address these shortcomings and improve the efficacy of stock market prediction. In this context, Reinforcement Learning (RL) emerges as a promising method to overcome these obstacles [17], offering a more effective means of addressing the complexities and uncertainties inherent in stock market prediction.

Submit Date: 2023-11-02
Revise Date: 2024-05-27
Accept Date: 2024-07-09
✉ Corresponding author

Deep Reinforcement Learning (DRL) is an emerging area that combines principles from DL and RL, enabling an agent to learn an optimal behavioral strategy in a specific environment by utilizing feedback in the form of rewards or punishments to improve its decision-making process. DRL facilitates processing large amounts of inputs and determines actions to optimize objectives without manual engineering of the state space [18]. The employment of DRL in economics has experienced a remarkable surge in popularity due to its scalability, rendering it ideal for handling high-dimensional problems, especially in noisy and nonlinear patterns frequently observed in economic data [10].

DRL offers advantages in stock market trading, specifically in generating profitable trades and making strategic decisions [9], [16], with [16] particularly noting the superiority of DRL over other machine learning methods. A range of studies have explored the application of DRL algorithms in stock market trading. For instance, [19] uses the Deep *Q*-Network (DQN) algorithm in financial quantitative trading, emphasizing its ability to capture hidden dependencies and potential dynamics in stock data. [20] found that the DQN outperformed the Deep Deterministic Policy Gradient (DDPG) in limiting financial losses and making effective trading decisions. The study by [21] trains DQN and Double DQN (DDQN) agents in a simulated trading environment. Surprisingly, DQN outperformed DDQN in generating profitable stock trading strategies. [22] applied DDQN to a pair trading strategy, successfully learning stock price patterns where two stocks tend to move in opposite directions and eventually balance each other out. [23] and [16] both report significant outperformance of DDPG-based strategies compared to other methods. [24] and [25] further highlight the potential of DDPG in automating trading decisions and adapting to changing market conditions. These studies collectively underscore the potential of DRL in stock market trading. However, one significant limitation of these methods is their reliance on large amounts of historical data to learn optimal trading strategies, which can be computationally intensive and time-consuming. Additionally, DRL models can struggle with the high volatility and non-stationary nature of financial markets, leading to suboptimal performance during sudden market shifts or crashes [9][26]. Furthermore, DRL models like DQN and DDPG may suffer from overfitting to historical data, making them less adaptable to new, unseen market conditions [27].

This paper presents a methodology that develops profitable stock trading strategies using two RL approaches: Gated DQN (GDQN) and Gated DDPG (GDPG). We utilize Convolutional Neural Networks (CNNs) and gated recurrent units (GRUs) for feature extraction and capturing temporal dependencies in stock market data, respectively. An attention mechanism is also employed to focus on critical time periods in the stock data. This integration allows the model to focus on principal characteristics and particular time frames within the stock information, ultimately improving its capacity to make well-informed investment choices. Using this framework, a DRL agent can make suitable trades by accurately predicting market fluctuations, effectively overcoming the limitations of conventional techniques, and delivering a more reliable and precise strategy. The main contributions are as follows:

1. Our architecture combines CNNs to extract key features, GRUs to capture temporal relationships, and an attention mechanism to pinpoint critical periods in the stock data;
2. This integration with GDQN and GDPG creates a robust framework. It empowers the RL agents to make informed trading decisions by analyzing relevant features and temporal patterns;
3. The attention mechanism stands as a novel addition, allowing the model to actively focus on specific, potentially market-moving moments within the data, leading to more accurate and profitable trading decisions.

The remainder of this paper is structured as follows: Section 2 outlines the related literature. The foundational concepts are described in Section 3. Section 4 explicates the methodology employed. Section 5 presents the experimental outcomes together with their analyses, and Section 6 concludes the paper by summarizing the major findings and their implications.

2. Related work

Existing methodologies for predicting stock prices and trading stocks fall into two main categories [28]: traditional approaches such as fundamental and technical analysis, which rely on historical data and market trends [29]; and technological methods such as ML methodologies, which utilize advanced computational techniques [1].

2.1. *Traditional Approaches*

There are two primary traditional approaches for analyzing stock markets: fundamental analysis and technical analysis [30]. Fundamental analysis seeks to determine the intrinsic value of stocks by examining a company's financial statements, economic forecasts, and various qualitative and quantitative factors [31]. This approach assesses whether a company's stock is undervalued or overvalued by analyzing key metrics such as the Price-to-Earnings ratio (P/E) and the Price-to-Book ratio (P/B), where a low P/E ratio indicates higher returns and a high P/B ratio suggests overvaluation. However, the effectiveness of fundamental analysis can be susceptible to false signals and is constrained by the non-linear nature of the systems governing stock prices and a lack of comprehensive knowledge [30].

Technical analysis involves examining past price trends and patterns to predict future price movements, based on the premise that historical trends influence future market performance [32]. This approach employs various technical indicators and principles, such as the belief that prices reflect all available information. Although primarily used by short-term investors, this method has several drawbacks. It relies on predetermined guidelines set by experts that evolve slowly and often exclude numerous factors that can affect stock prices [30]. [33] cautioned that while technical analysis methods can generate returns that exceed the investment value, they have limited predictability.

The debate between fundamental and technical analysis has existed for a long time, with supporters of each method holding differing perspectives [34]. However, some studies have explored the potential of combining technical and fundamental analysis in stock market trading strategies. [35] suggested that combining technical analysis with fundamental analysis can mitigate the risks associated with technical analysis. An integrated system that combines both methods is found to be more effective for stock selection [36], [37]. This is supported by evidence that some technical indicators can predict market trends [38]. [39] and [40] found that such a combination can lead to market outperformance. However, the process of integrating both approaches can be time-consuming and complex, requiring expertise in interpreting and reconciling potentially conflicting signals [41].

2.2. *Machine Learning Approaches*

With more advanced computational methods and machine learning techniques, researchers have explored ways to overcome traditional methods' limitations [37][42][43][44][45]. These models often employ sophisticated algorithms and neural networks to identify patterns and relationships in large datasets, potentially providing more accurate and reliable trading signals. For instance, [46] presents a DL model that utilizes 55 diverse features, including stock data, technical indicators, exchange rates, commodity prices, and investor data. These inputs are fed into a 4-layer Long Short-Term Memory (LSTM), resulting in accurate predictions of stock prices and market index volumes. [47] takes a different approach, developing an algorithmic trading system in the Tehran Stock Exchange. Their study proposes a trading system by employing the three tools of genetic algorithm, fuzzy logic, and neural network, to address the shortcoming of human subjectivity in technical analysis. The results show that the new system is statistically better with higher profitability potential. [48] compares various ML and DL models for stock market trend prediction, finding that Recurrent Neural Network (RNN) and LSTM perform best, particularly when using continuous data. In a separate study, [49] focuses on the future prediction of stock market groups, with LSTM again showing the most accurate results. These studies collectively highlight the potential of DL and ML methods in stock market trading, particularly when applied to continuous data and using algorithms such as RNN and LSTM. [50] develops a method to predict stock price fluctuations by analyzing financial news and sentiment. This method utilizes GRU and introduces the two-stream GRU, an advanced DL approach that outperforms the LSTM model in performance metrics. In another related study, [8] explores the fusion of CNN with LSTM, using CNN for feature extraction from historical stock prices and technical indicators, and transferring these features to LSTM for modeling temporal relationships and patterns in time-series data. The combination of CNN and LSTM significantly improves the predictive performance of stock forecasting. [51] presents a hybrid stock prediction model combining CNN and bi-directional GRU. The model includes a feature selection component to optimize input data performance. The role of CNN involves focusing on localized information and minimizing computational complexity, while bi-directional GRU processes time-series data to boost performance. The study reveals that the hybrid model outperforms alternative single models. [52] proposes a multi-input LSTM model for stock price prediction that uses the target stock's price history as the primary factor to control input gates and adaptively filter auxiliary factors such as related stock prices and market indices. Additionally, an attention mechanism is employed to weight the different inputs when combining them into the memory cell state.

Recent research has shown the potential of DRL in stock market trading, with studies reporting improved investment returns and profitability. The primary advantages of DRL include its capacity to learn adaptive trading strategies from large financial datasets and its ability to outperform traditional methods. For instance, [9] utilize three DRL algorithms, namely DQN, DDQN, and Dueling DDQN (D3QN), for stock trading. The profitability of DRL agents is assessed by trading in ten randomly selected stock markets. The results of the experiments indicate that DQN outperforms the other two methods for most of the stocks, albeit with varying levels of success. [53] proposes an enhanced stock trading strategy based on DQN, incorporating a dual-action selection and dual-environment mechanism to improve performance. The study by [54] explores the construction of stock trading models through the application of DRL algorithms. Specifically, the DDPG approach demonstrates superior performance over the advantage actor-critic method in terms of convergence, stability, and evaluation metrics. In a different study, [55] introduces a Deep Recurrent *Q*-Network (DRQN) for financial trading, which integrates a RNN to facilitate training at intermediate intervals rather than solely at the end. This approach eliminates the necessity of random exploration during reinforcement learning by offering supplementary feedback to the agent. However, it is restricted to financial trading under distinct market assumptions, such as fixed transaction costs concerning the traded foreign exchange currency's value. While DRL exhibits promise for developing profitable stock trading agents, further progress is required to enhance its generalization capability. [16] proposes a sentiment analysis-based approach to automate stock trading. They address the limitations of prior supervised learning methods by using a Partially Observed Markov Decision Process (POMDP). The approach unifies prediction and capital allocation with the twin delayed DDPG algorithm, handling continuous action spaces across multiple assets.

The combination of CNN for extracting informative features before inputting to DDPG has shown to be an effective approach [56]. The study by [57] proposes a novel DRL framework for developing an AI trader for intraday trading in financial markets. This framework utilizes technical indicators and the DDPG algorithm to discover the best trading policy. Additionally, two CNNs are employed in the actor and critic parts of the DDPG to extract informative features from the multivariate time series of technical indicators. The evaluations show the effectiveness of this approach, which outperforms other methods such as random trading and a basic RL trading agent. [58] proposes a method composed of CNN, bi-directional LSTM, and an attention mechanism for predicting the next day's stock closing price. Specifically, the attention mechanism captures the influence of past feature states on the stock price. The evaluation of the method on 1000 trading days of data shows the best performance compared to seven other methods. These studies collectively highlight the potential of DRL, particularly when combined with CNNs and attention mechanisms, in improving stock market trading performance.

3. Preliminaries

In the complex and dynamic financial market, where data is often noisy and nonparametric, DRL emerges as a promising approach. In DRL, the agent is treated as an entity, while everything else is considered the environment. The agent's primary goal is to optimize the cumulative reward by learning from feedback in the form of reward signals, which are obtained through interactions with the environment. This approach enables the agent to adapt and make informed decisions in response to ever-changing market conditions.

3.1. *Deep Q-Network*

An RL problem involves the representation of an agent-environment system, in which the agent makes decisions based on the environment's state and receives feedback for its actions. The typical approach to modeling RL problems is through the utilization of the Markov Decision Process (MDP) under the assumption of fully observable environments. Nevertheless, in practical settings, achieving the Markov property is challenging [59], leading to unobserved factors that introduce uncertainty into decision-making processes. To address this issue, we propose a POMDP model that expands the MDP framework to accommodate incomplete or noisy state observations. The POMDP can be defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O})$ with each component representing states, actions, transition probabilities, rewards, observations $o \in \Omega$, and the conditional probability distribution over Ω based on the state-action pairs $(o_t|s_{t+1}, a_t)$.

The goal of the agent is to enhance the overall cumulative reward throughout its operational timeline. The agent progressively acquires knowledge to enhance its policy π towards actions that generate the most favorable outcomes by maximizing the anticipated cumulative rewards expressed as Eq. (1), where τ represents trajectories and γ is the discount factor.

$$\mathbb{E}_{s_t=s, a_t=a, \tau\sim\pi}[R_t] = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t}^{\infty} \gamma^t r_t\right]. \quad (1)$$

A solid trading strategy concentrates on generating the maximum profit out of the bulk of deals rather than every single trade to assure long-term success in the stock markets [25]. This can be accomplished through the use of the DQN framework [60]. Learning Q^π function involves leveraging the Markov property, which implies that the next state is entirely reliant on the current state, making the future conditionally separate from the past, provided that the present state is known [18]. This property has a recursive structure as follows:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma Q^\pi\left(s_{t+1}, \pi(s_{t+1})\right)\right]. \quad (2)$$

In Eq. (2), s_t stands for the current state, a_t for the action performed at the current time step, s_{t+1} for the next state, r_t for the feedback received at the time step t, and π for the agent's policy. The Bellman equation is formulated recursively, enabling the improvement of Q^π through bootstrapping, wherein the current values of estimated Q^π are utilized to refine the estimation:

$$Q^\pi(s_t, a_t) \leftarrow Q^\pi(s_t, a_t) + \alpha\delta. \quad (3)$$

In Eq. (3), α is referred to as the learning rate and δ is the Temporal Difference (TD) error, which is expressed as follows:

$$\delta = r_t + \gamma \max_a Q^\pi(s_{t+1}, a) - Q^\pi(s_t, a_t). \quad (4)$$

In Eq. (4), the variable a is used to represent all potential actions in state s_{t+1} when finding the maximum value of the Q^π function. The potency of the DQN is found in its capacity to succinctly depict multidimensional states and Q-function estimation by utilizing neural networks. The stock market has countless states and state-action pairs, making it infeasible to construct an infinite Q-table with unlimited memory [25]. Accordingly, DQN uses experience replay memory to train, which retains transitions in a cyclic buffer of the form $(s_t, a_t, s_{t+1}, r_{t+1})$ allowing for offline training that decreases interactions with the environment, decreases variance, and disrupts temporal correlations.

Another approach to enhance stability entails utilizing a fixed target network featuring weights sourced from the policy network. The policy network relies on the target network to avoid the need for frequent TD error calculations. The policy network adjusts its weights to correspond with the target network after a certain number of training steps [18].

3.2. *Double Deep Q-Network*

DQN overestimates the value of actions, making it challenging for the agent to determine the best course of action [18]. To address this issue, the primary Q-network (Q^θ) selects the action a_{t+1}. Subsequently, the target Q-network $(Q^{\theta'})$ computes the action-value of a_{t+1} using the primary Q-network's output [61]. The computation process is as follows:

$$Q^{\theta'}(s_t, a_t) = r + \gamma Q^{\theta'}(s_{t+1}, \underset{a_{t+1}}{argmax} Q^\theta(s_{t+1}, a_{t+1})). \quad (5)$$

The selection of actions during the *argmax* operation is affected by the weights θ of the primary Q-network. The current values defined by θ are crucial for estimating the policy's value. However, for a precise evaluation of the policy's value, a separate set of parameters denoted as θ' is employed [61]. This method can help reduce the overestimation of actions in stock trading, leading to more dependable and consistent learning.

3.3. *Deep Deterministic Policy Gradient*

DDPG is an actor-critic approach that operates off-policy [62], [63] and employs target networks and experience replay to address challenges in continuous action scenarios. This technique demonstrates effective generalization and proficient learning of optimal actions. The DDPG distinguishes itself through its deterministic policy gradient, which anticipates gradient calculation on the action-value function, leading to enhanced evaluation efficiency compared to conventional stochastic policy gradients. Two neural networks constitute the actor and critic modules, sharing an identical structure but exhibiting distinct parameters. Hence, a total of four neural networks exist cumulatively. The actor chooses actions in the following manner [63]:

$$a_t = \mu(s_t|\theta^\mu) + \epsilon_t. \quad (6)$$

In Eq. (6), the most probable action is denoted by μ, whereas ϵ denotes stochastic noise, incorporated into the chosen action. θ^{μ} represents the deterministic policy network parameterized by θ. The value network is updated similarly as is done in Eq. (5) and can be obtained by the Bellman equation as follows:

$$y_t = r_t + \gamma Q'\left(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})\middle|\theta^{Q'}\right). \tag{7}$$

In Eq. (7), $\theta^{\mu'}$ is the target policy network weights and $\theta^{Q'}$ is the target Q-network weights. However, in DDPG, the next-state Q' values are calculated with the target value network $\theta^{Q'}$ and target policy network $\theta^{\mu'}$. Then, the Mean Squared Error (MSE) between the updated and the original Q values of the critic network are minimized as expressed in Eq. (8), with y_t defined in Eq. (7).

$$\mathcal{L}(\theta^{Q}) = \frac{1}{N}\sum_{t=1}^{N}\left(y_t - Q(s_t, a_t; \theta^{Q})\right)^2. \tag{8}$$

The objective of an agent is to maximize the expected return. For a deterministic policy $\mu(s;\theta^{\mu})$, the objective function $J(\theta)$ is defined as the expected return starting from an initial state distribution:

$$J(\theta) = \mathbb{E}_{s\sim\rho^{\mu}}\left[Q^{\mu}\left(s, \mu(s;\theta^{\mu})\right)\right]. \tag{9}$$

In Eq. (9), ρ^{μ} is the state distribution under the policy μ. To find the gradient of $J(\theta)$ with respect to θ^{μ}, we apply the chain rule [62]:

$$\nabla_{\theta^{\mu}}J(\theta) = \nabla_{\theta^{\mu}}\mathbb{E}_{s\sim\rho^{\mu}}\left[Q^{\mu}\left(s, \mu(s;\theta^{\mu})\right)\right]. \tag{10}$$

Using the linearity of the expectation operator:

$$\nabla_{\theta^{\mu}}J(\theta) = \mathbb{E}_{s\sim\rho^{\mu}}\left[\nabla_{\theta^{\mu}}Q^{\mu}\left(s, \mu(s;\theta^{\mu})\right)\right]. \tag{11}$$

By applying the chain rule to the inner gradient, the gradient of the Q-value function with respect to the policy parameters θ^{μ} can be decomposed into two parts:

$$\nabla_{\theta^{\mu}}Q^{\mu}\left(s, \mu(s;\theta^{\mu})\right) = \nabla_a Q^{\mu}(s,a)|_{a=\mu(s)}\nabla_{\theta^{\mu}}\mu(s;\theta^{\mu}). \tag{12}$$

The deterministic policy gradient theorem [62] states that the gradient of the expected return $J(\theta)$ with respect to the policy parameters θ^{μ} can be written as:

$$\nabla_{\theta^{\mu}}J(\theta) = \mathbb{E}_{s\sim\rho^{\mu}}\left[\nabla_a Q^{\mu}(s,a)|_{a=\mu(s)}\nabla_{\theta^{\mu}}\mu(s;\theta^{\mu})\right]. \tag{13}$$

Since we don't have access to the true state distribution ρ^{μ} in Eq. (13), we can approximate the expectation using a mini-batch of transitions (s_t, a_t, r_t, s_{t+1}) sampled from a replay buffer. The policy function is optimized by computing the derivative of the objective function with respect to the policy parameters and then averaging the sum of gradients calculated from the mini-batch during policy updates in an off-policy manner. Therefore, to update the policy parameters θ^{μ}, we perform gradient ascent on the estimated policy gradient. By replacing the expectation with a sample-based estimate, we can express it as:

$$\nabla_{\theta^{\mu}}J(\theta) \approx \frac{1}{N}\sum_{t}\left[\nabla_a Q(s_t, a_t;\theta^{Q})|_{a_t=\mu(s_t)}\nabla_{\theta^{\mu}}\mu(s_t;\theta^{\mu})\right]. \tag{14}$$

In Eq. (14), $\nabla_a Q(s_t, a;\theta^{Q})$ is the gradient of the Q-value with respect to the action, evaluated at s_t and $a = \mu(s_t)$, $\nabla_{\theta^{\mu}}\mu(s_t;\theta^{\mu})$ is the gradient of the policy with respect to its parameters, evaluated at $s = s_t$. To update the policy parameters θ^{μ}, we perform gradient ascent on the estimated policy gradient with α as the learning rate as follows:

$$\theta^{\mu_{\text{new}}} \leftarrow \theta^{\mu_{\text{old}}} + \alpha\frac{1}{N}\sum_{t=1}^{N}\left[\nabla_a Q(s_t, a_t;\theta^{Q})|_{a_t=\mu(s_t)}\nabla_{\theta^{\mu}}\mu(s_t;\theta^{\mu})\right]. \tag{15}$$

The DDPG algorithm uses target networks for both the Q-value function and the policy to improve stability during training. The target networks are updated slowly towards the current networks using a technique called soft updates or Polyak averaging, expressed as follows:

$$\begin{aligned}\theta^{Q'} &\leftarrow \rho\theta^{Q} + (1-\rho)\theta^{Q'},\\ \theta^{\mu'} &\leftarrow \rho\theta^{\mu} + (1-\rho)\theta^{\mu'}.\end{aligned} \tag{16}$$

In Eq. (16), ρ is the Polyak averaging coefficient, which controls the balance between old and new network parameters. It is a value ranging from 0 to 1, and this parameter influences the gradual adjustment of target network weights by the formulas to align with the current network weights.

3.4.*Gated Recurrent Unit*

While LSTM cells boast a higher learning capacity than standard recurrent cells, their increased number of parameters leads to a higher computational burden [64]. This is where GRU shines. By effectively capturing temporal dynamics within their memory using a streamlined gating mechanism with fewer parameters, GRUs offer superior efficiency [65]. This efficiency makes GRUs a compelling choice for tasks like analyzing time-series stock market data, where capturing long-term dependencies is crucial [67][66]. The hidden state of the GRU is formulated as follows:

$$\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}h_{t-1} + b_{hr}), \\ z_t &= \sigma(W_{iz}x_t + b_{iz} + W_{hz}h_{t-1} + b_{hz}), \\ \widetilde{h_t} &= \tanh\left(W_{ih}x_t + b_{ih} + r_t \odot (W_{hh}h_{t-1} + b_{hh})\right), \\ h_t &= (1 - z_t) \odot \widetilde{h_t} + z_t \odot h_{t-1}. \end{aligned} \tag{17}$$

In Eq. (17), x_t is the input, h_t is the hidden state, r_t is the reset gate, z_t is the update gate, $\widetilde{h_t}$ is the new candidate hidden state all at time step t. The sigmoid and hyperbolic tangent activation functions are denoted by σ and tanh, and W and b are the weight and bias matrices/parameters, respectively. The GRU cell simplifies the number of parameters by merging the forget and input gates from the LSTM cell into a single update gate. In contrast to the LSTM, the GRU cell is equipped with only two gates: an update gate and a reset gate. This design allows for preserving a single gating signal and its associated parameters, resulting in a more efficient architecture [65][64].

3.5. *Attention Mechanism*

When working with time-series data, incorporating an attention mechanism that divides the data into distinct segments is a practical approach that has the potential to enhance the decoder's efficacy in producing new values [66][65]. The central concept of an attention model is to determine and disseminate the weight of attention with emphasis on significant material by elevating its weight [68][67].

In the context of trading in stock markets, a common strategy involves incorporating attention mechanisms to refine trading decisions. One way of employing an attention mechanism to enhance the effectiveness of trading models is through methods like Bahdanau style attention (additive attention). In this setup, the encoder calculates a context vector based on input features, and the attention layer refines this vector by assigning weights and performing element-wise multiplication. This process contributes to a more refined and informed decision-making process within the trading model. The decoder utilizes the previous hidden state and the context vector to generate trading predictions informed by historical data and changing market conditions [67].

4. Methodology

In this section, we elaborate on the state and action spaces, followed by the description of the reward function. Subsequently, we introduce the proposed methodology for training the agent with detailed mathematical models and formulas to ensure clarity and comprehensiveness.

4.1. *State Space*

The primary obstacle in stock trading lies in identifying the optimal trading time based on market conditions and executing appropriate trading strategies. In the case of stock market data, the commonly used information comprises regular time intervals of the opening and closing prices, adjusted closing prices, high and low values, and volume. However, raw market data is laden with high levels of complexity and noise, rendering it challenging for deep neural networks to regress over time.

The state space, functioning as the agent's guiding eye during trades, should encompass valuable information that the agent can leverage to achieve optimal efficiency. The first step is to derive technical indicators, which are mathematical calculations emanating from stock data, to provide a comprehensive overview of the market conditions from different angles [25]. Technical indicators employed include Moving Average (MA) and Exponential Moving Average (EMA), which represent average technical indicators. Moving Average Convergence/Divergence (MACD) that characterizes a tendency. The On Balance Volume (OBV) assesses volume flow and predicts price trends by analyzing trading volume [25]. The Average True Range (ATR) gauges asset volatility and helps spot potential price movement and trend strength. Thus, the state space in our model includes open, high, low, close, and adjusted close prices, as well as the technical indicators mentioned derived as follows:

1. **MA:** The simple moving average over n days is defined as Eq. (18), with pv representing price and volume data.

$$MA_t = \frac{1}{n}\sum_{i=0}^{n-1} pv_{t-i}. \tag{18}$$

2. **EMA:** The exponential moving average over n days, which assigns more weight to recent prices than to data from the distant past, is defined as Eq. (19). Here, p_t represents the price data at time t, and α denotes the smoothing factor, typically ranging between 0 and 1, where smaller values give more weight to older data points.

$$EMA_t = \alpha \times x_t + (1 - \alpha) \times EMA_{t-1}. \tag{19}$$

3. **MACD:** The difference between a long-term moving average and a short-term moving average is defined as Eq. (20), where the short-term is 12-day and the long-term is 26-day, and *EMA* is defined in Eq. (19).

$$MACD_t = EMA_{12,t} - EMA_{26,t}. \tag{20}$$

4. **OBV:** A momentum indicator that utilizes volume flow to forecast changes in stock price is defined as Eq. (21), where v_t represents the volume at time t.

$$OBV_t = OBV_{t-1} + \begin{cases} v_t & \text{if } p_t > p_{t-1}, \\ -v_t & \text{if } p_t < p_{t-1}, \\ 0 & \text{if } p_t = p_{t-1}. \end{cases} \quad (21)$$

5. **ATR**: A measure of volatility, defined as Eq. (22), which is defined as the average of True Ranges (TR) over n days, where the true ranges is defined as Eq. (23), with H_t and L_t being the high and low prices of the day.

$$ATR_t = \frac{1}{n}\sum_{i=0}^{N-1} TR_{t-i}, \quad (22)$$

$$TR_t = max(H_t - L_t, |H_t - P_{t-1}|, |L_t - P_{t-1}|). \quad (23)$$

4.2. Action Space

The RL agent has three options for executing actions such as buy, sell, or hold. Incorporating additional actions introduces unnecessary complexity without yielding significant positive effects, thereby prolonging the learning process. Skilled agents avoid holding to maximize gains. Thus, utilizing only the three key actions suffices for successful agent training. These actions are represented as:

$$a_t \in \{\text{Buy,Sell,Hold}\}. \quad (24)$$

The exchange of stocks ought to correlate with the amount traded, aligning with market demands and offers. Nonetheless, for the purpose of simplicity, our experiment disregards this factor and focuses on a nominal quantity of one share per transaction as done in [25].

4.3. Reward Function

An agent utilizing RL techniques can learn the most efficient policy for trading stocks to attain the highest level of profit. Therefore, the development of a practical reward function holds paramount importance. After a transaction in the stock market, the agent receives an immediate reward based on the asset value change. The reward function is typically defined in terms of the Rate of Return (ROR) according to the following manner:

$$R_t = \frac{V_t - V_{t-1}}{V_{t-1}}. \quad (25)$$

In Eq. (25), V_t represents portfolio value at time t, and V_{t-1} represents the portfolio value at time $t-1$. The rate of return is determined by computing the percentage increase or decrease in the stock price by multiplying the result by 100%.

4.4. Proposed Method

In the context of the stock market, discerning patterns from price series and technical indicators poses a significant challenge due to their elusive nature. To address this, we propose two model variants based on GRU. This strategic choice empowers to effectively extract informative features, thereby facilitating the exploration of valuable patterns within the market data.

4.4.1. CNN-GRU Model

The extraction of features from financial data represents a crucial challenge within the domain of market prediction, for which numerous methodologies have been proposed [68]. CNNs find wide application in image processing, yet their utility extends to sequential data such as stock market price movements. In our first proposed model, a 1D-CNN layer initially extracts pertinent features from stock market price data and its affiliated details. The convolutional layer applies a set of filters to the input data to extract features. For a 1D-CNN, the convolution operation can be expressed as:

$$y_i = \sum_{j=0}^{k-1} x_{i+j} \cdot w_j + b. \quad (26)$$

In Eq. (26), y_i is the output feature map, x is the input sequence, w is the filter (or kernel) of size k, and b is the bias term. After the convolutional layer, Batch Normalization (BN) is applied to stabilize and accelerate the training process by normalizing the output of the previous layer. For a 1D sequence, the batch normalization operation can be expressed as Eqs. (27) and (28), where $\hat{x}_i$ is the normalized input, μ is the mean of the input, σ^2 is the variance of the input, ϵ is a small constant to avoid division by zero, and γ and β are learnable parameters for scaling and shifting.

$$\hat{x}_i = \frac{x_i - \mu}{\sqrt{\sigma^2 + \epsilon}}, \quad (27)$$

$$y_i = \gamma\hat{x}_i + \beta. \quad (28)$$

After the data is processed by the batch normalization layer, the output is fed to the GRU layer. The role of the GRU layer is to capture the temporal dependencies crucial for trading and further enhance the capabilities of the models. The GRU layer follows the mathematics provided in Section 3.4. By combining these layers, the model can effectively extract and process features from stock market data, capturing both spatial and temporal dependencies.

4.4.2. *CNN-GRU-Attention Model*

In our second proposed model, which builds upon the first model, an additive attention mechanism is employed post-GRU output, as illustrated in Fig. 1. This mechanism assigns varying weights to input sequence components, facilitating focused attention on pivotal information. This approach proves beneficial in stock market trading, considering the varying impact levels of different time periods on the decision-making process. This model extends the CNN-GRU model by calculating attention weights and the context vector based on the current hidden state and the sequence of hidden states from the previous layer, as expressed in Eqs. (29) and (30). Here, $e_{t,i}$ is the attention score for the i-th element of the sequence at time step t, V, W, and U are learnable weight matrices, h_i is the hidden state of the i-th element in the sequence, h_{t-1} is the previous hidden state of the GRU, b is the bias term, and $\alpha_{t,i}$ is the attention weight for the i-th element at time step t. Furthermore, the context vector is calculated as expressed in Eq. (31), with c_t being the context vector at time step t, and $\alpha_{t,i}$ the attention weight for the i-th element at time step t.

$$e_{t,i} = V^T \tanh(Wh_i + Uh_{t-1} + b), \tag{29}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{k=1}^{T} \exp(e_{t,k})}, \tag{30}$$

$$c_t = \sum_{i=1}^{T} \alpha_{t,i} h_i. \tag{31}$$

The models depicted in Fig. 1 function as the preprocessing networks that take normalized data as input. During training, the networks incorporate a Fully Connected (FC) layer along with the MSE loss, which quantifies the discrepancy between the predicted and actual values. During the testing phase or trading, the FC layer is bypassed, and the weights of all layers are frozen, allowing the output to seamlessly interface with the DRL modules and serve as their input.

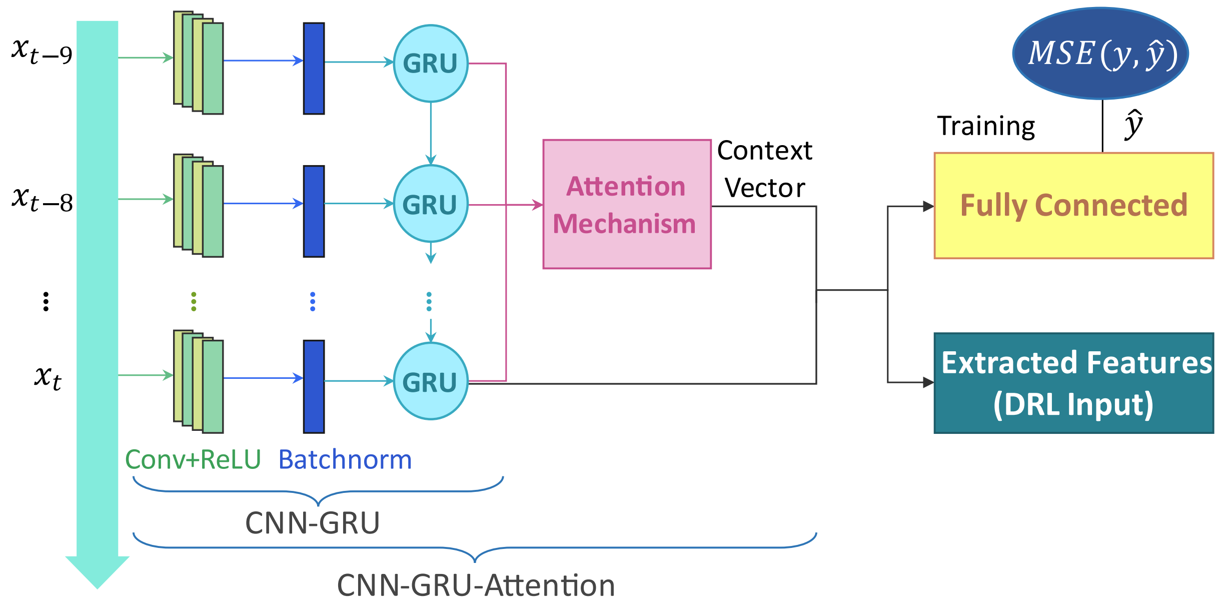
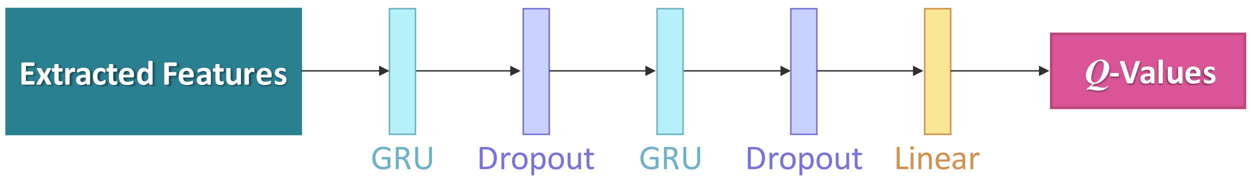


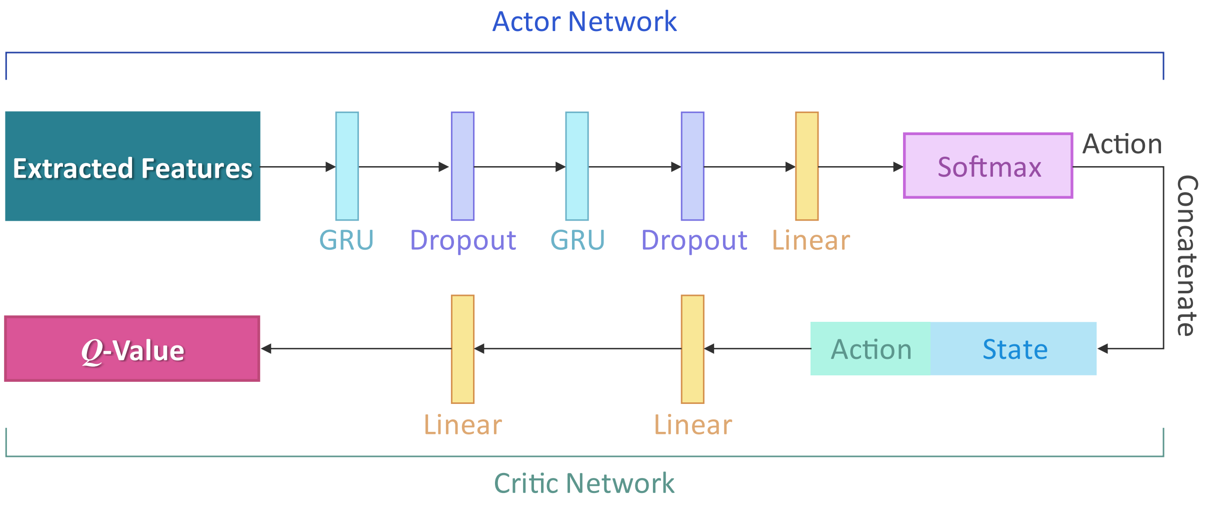
Figure 1: Architectural overview of the proposed models. The network undergoes initial pre-training on stock data, after which the learned weights are frozen, and the generated output is directed to the RL module for the RL training.

4.4.3. *DRL Modules*

The attention-based gated DDQN and DDPG methods constitute robust strategies and seamlessly integrate RL and DL paradigms. Fig. 2(a) and Fig. 2(b) illustrate the diagram of the GDQN and GDPG frameworks, respectively, as is proposed in [25]. Both the GDQN and actor network of GDPG derive their input from the preprocessing network situated before them, which undertakes initial data preprocessing, and subsequently supplies the refined input to the respective RL modules. The gated architecture, which includes components such as gating mechanisms, enhances the models' ability to handle various market conditions and adapt to changes over time. These mechanisms can modulate the flow of information within the network, allowing the model to dynamically adjust its focus based on the current market state. This collaborative architecture enhances the synergy between DL and RL components, augmenting the overall performance of the models in stock trading. The pseudocode of the GDQN and GDPG algorithms is illustrated in Algorithm 1 and 2, respectively, providing a detailed step-by-step guide on implementing these frameworks.



(a) GDQN



(b) GDPG

Figure 2: Architectural Diagrams of RL modules. The diagrams depict the compositions of the RL modules, illustrating the flow of information and interactions among components.

5. Experimental results and discussion

All experimental procedures were executed on a standard laptop featuring an Intel Core i7-6700HQ processor operating at 2.6 GHz, accompanied by 16.0 GB of RAM, with a GeForce GTX 960M GPU featuring 4 GB of onboard RAM.

The selected empirical stock data is derived from the U.S. market, spanning from January 2, 1992, to July 27, 2023. The symbol of each stock data and the company representing that symbol is described in Table 1. The dataset is strategically divided: the interval from 1992 to 2020 is employed for training, while the subsequent period from 2020 to 2023 is dedicated to testing the proposed methodology.

Algorithm 1 The GDQN algorithm

```
1:  Input: state of the stock $\boldsymbol{s_t}$, $Q$ network and its parameters $\boldsymbol{\theta}$, target $Q^{target}$ network and its parameters $\boldsymbol{\theta}'$
2:  Output: weights $\boldsymbol{\theta}$ for $Q$ network
3:  Initialize the $\boldsymbol{s_t}$
4:  Initialize the memory replay repository $\boldsymbol{D}$ to capacity $\boldsymbol{N}$
5:  Initialize the $Q$ network with random weights $\boldsymbol{\theta}$, and initialize the target $Q^{target}$ network with $\boldsymbol{\theta}' = \boldsymbol{\theta}$
6:  for episode = 1, $M$ do
7:      for $t$ = 1, $T$ do
8:          Fetch state of the stock from $D$ and form input $\boldsymbol{s_t}$
9:          Select $\boldsymbol{a_t} = \underset{a}{argmax}\mathrm{Q}(\boldsymbol{s_t}, \boldsymbol{a}; \boldsymbol{\theta})$ according to the $\boldsymbol{s_t}$
10:         Otherwise select random action $\boldsymbol{a_t}$ with probability $\boldsymbol{\epsilon}$
11:         Execute action $\boldsymbol{a_t}$, reward $\boldsymbol{r_t}$ and calculate $\boldsymbol{s_{t+1}}$
12:         Store the transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{s_{t+1}})$ in $\boldsymbol{D}$
13:         Sample minibatch $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{s_{t+1}})$ randomly from $\boldsymbol{D}$
14:         Set $\boldsymbol{y}$ as Eq. (5)
15:         Train the network with loss function $\boldsymbol{L}(\boldsymbol{\theta}) = \mathbb{E}\left[\left(\boldsymbol{y} - Q(\boldsymbol{s}, \boldsymbol{a}; \boldsymbol{\theta})\right)^{2}\right]$
16:         if $t$ mod $C$ = 0 then
17:             Update target $Q$-network: $\boldsymbol{\theta}^{Q'} = \boldsymbol{\theta}^{Q}$
18:         end if
19:     end for
20: end for
21: Return weights $\boldsymbol{\theta}'$ for $Q$ network
```

We investigated three preprocess network architecture models for experimental evaluation: 1) the sole GRU network of [25], 2) the combined CNN-GRU network, and 3) the extended CNN-GRU-Attention network. For simplicity, we refer to the three variants as, respectively, models A, B, and C. The shared hyperparameters across all three models were consistent, and the history sequence length was uniformly set to ten. The complete details about the applied settings are demonstrated in Table 2.

For the experimental evaluation of preprocess networks, performance metrics such as MSE, Root MSE (RMSE), Mean Absolute Error (MAE), and Coefficient of Determination (R^2) are used to assess the models during the testing phase. These metrics provide insights into the model's ability to accurately predict stock prices. Lower values for MSE, MAE, and RMSE indicate more accurate predictions. Conversely, a higher R^2 value indicates a better fit of the model to the actual data, as it measures the proportion of the variance in the dependent variable that is predictable from the independent variables. These metrics are used to evaluate the performance of models that preprocess the data for the DRL modules. The results are provided in Table 3, with the best-performing model being outlined.

Algorithm 2 The GDPG algorithm

```
1:  Randomly initialize critic network $Q(\boldsymbol{s_t}, \boldsymbol{a_t}|\boldsymbol{\theta}^Q)$ and actor $\boldsymbol{\mu}(\boldsymbol{s_t}|\boldsymbol{\theta}^{\mu})$ with weights $\boldsymbol{\theta}^Q$ and $\boldsymbol{\theta^\mu}$.
2:  Initialize target network $Q'$ and $\boldsymbol{\mu}'$ with weights $\boldsymbol{\theta}^{Q'} \leftarrow \boldsymbol{\theta}^Q, \boldsymbol{\theta}^{\mu'} \leftarrow \boldsymbol{\theta^\mu}$.
3:  Initialize the memory replay repository $\boldsymbol{D}$ to capacity $\boldsymbol{N}$
4:  for episode = 1, $M$ do
5:      Initialize a random process $\boldsymbol{\epsilon}$ for action exploration
6:      Receive initial state stock $\boldsymbol{s_1}$
7:      for $t$ = 1, $T$ do
8:          Select action according to Eq. (6)
9:          Execute action $\boldsymbol{a_t}$ and observe reward $\boldsymbol{r_t}$ and new state $\boldsymbol{s_{t+1}}$
10:         Store the transition $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{s_{t+1}})$ in $\boldsymbol{D}$
11:         Sample minibatch $(\boldsymbol{s_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{s_{t+1}})$ randomly from $\boldsymbol{D}$
12:         Set $\boldsymbol{y}$ as Eq. (7)
13:         Update critic by minimizing the loss as Eq. (8)
14:         Update the actor policy using the policy gradient as Eq. (14)
15:         Update the target networks as Eq. (16)
16:     end for
17: end for
```

Table 1: Sample stocks in the U.S. stock market.

Symbol	Company
AXP	American Express
CSCO	Cisco Systems
GE	General Electric
IBM	International Business
MSFT	Microsoft Corporation

From Table 3, we can infer that models B and C consistently outperform model A across all five stock markets due to lower error metrics and higher R^2. The performance plots of the three models are also demonstrated in Figs. 3 and 4. The superior performance of models B and C can be attributed to two factors. Firstly, CNNs are adept at capturing patterns within sequential data like stock prices. Secondly, the attention mechanism in model C focuses on the most relevant data points during learning, leading to potentially more accurate predictions. For AXP stock, model B achieves the best performance. This is also visualized in Fig. 3(a), where models B and C fit the data better compared to model A. In the context of CSCO stock, models B and C outperform model A, with model C having the upper hand in performance metrics. When considering MSFT stock, all three models present similar metrics. However, model C shows slightly higher error rates than models A and B, indicating lower performance, as is illustrated in Fig. 4(b).

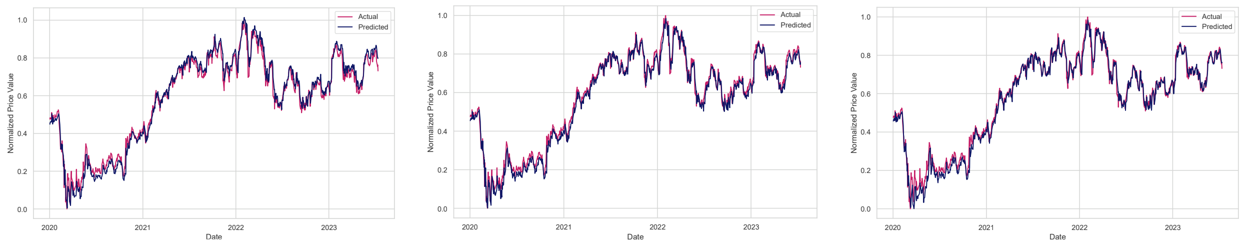
Table 2: Configuration setting of each model. This table details the layer structure, output shape, number of parameters, and shared weight information for both the CNN-GRU and CNN-GRU-Attention architectures.

Model	Layer	Output Shape	Parameters	Shared
B	Conv (ReLU)	(16,10)	160	16 filters, kernel 2, same padding
	Batchnorm	(16,10)	16	16 features
	GRU	(10,8)	384	8 neurons
C	Attention (W)	(1,8)	64	8 neurons
	Attention (U)	(10,8)	64	8 neurons
	Attention (V)	(10,1)	8	1 output

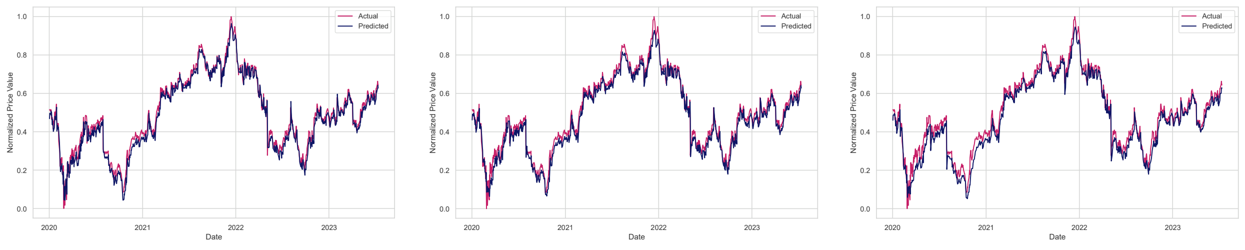
Table 3: Summary of model's performance in U.S. stock markets. The best-performing model (lowest RMSE, MSE, MAE; highest R^2) for each stock market is outlined.

Symbol	Model	RMSE	MSE	MAE	$\boldsymbol{R^2}$
AXP	A	0.0494	0.0024	0.0381	0.9394
	B	0.0344	0.0012	0.0259	0.9716
	C	0.0359	0.0013	0.0266	0.9673
CSCO	A	0.0428	0.0018	0.0336	0.9496
	B	0.0366	0.0013	0.0282	0.9629
	C	0.0358	0.0013	0.0281	0.9624
GE	A	0.0281	0.0008	0.0198	0.9778
	B	0.0241	0.0006	0.0172	0.9809
	C	0.0228	0.0005	0.0166	0.9844
IBM	A	0.0439	0.0019	0.0342	0.9332
	B	0.0415	0.0017	0.0316	0.9404
	C	0.0448	0.0020	0.0338	0.9326
MSFT	A	0.0311	0.0010	0.0234	0.9777
	B	0.0311	0.0010	0.0223	0.9775
	C	0.0313	0.0010	0.0221	0.9770

We assessed the effectiveness of DRL models using two key metrics: 1) the ROR calculated using Eq. (25) and expressed as a percentage (multiplied by 100%), and 2) the Sortino Ratio (SR), as detailed in [25]. The SR is a risk-adjusted performance measure that considers not only the average returns but also focuses on downside volatility, which makes it particularly relevant in the context of stock market trading, where minimizing losses during market downturns is of utmost significance. The hyperparameters used in the experiments for GDQN and GDPG are listed in Table 4, which provides a detailed overview of the hyperparameter settings for each model. The performance of the trading strategies is presented in Table 5, and a graphical summary of this table is provided in Fig. 5 for enhanced comparability. Furthermore, the performance of GDQN and GDPG in conjunction with each model is illustrated separately for each stock dataset in Figs. 6 and 7, thereby facilitating a more comprehensive comparison of the strategies. The analyses provide valuable insights into the performance of these models across various stock markets. As illustrated in Fig. 5, models B and C demonstrate notable advantages and benefits in efficiently anticipating trends in trading stocks. Additionally, it is noteworthy that model B's performance stands out in Table 3. This trend is also reflected in Table 5, where model B excels with GDQN and ranks second with GDPG. Moreover, this pattern is replicated for CSCO stock, where model C's predictions exhibit superior accuracy and profitability. Interestingly, model B also exhibits an advantage in predicting IBM stock prices. However, it is essential to note that the attention model's performance was slightly lower for this stock, suggesting that the attention mechanism did not effectively capture the key features.

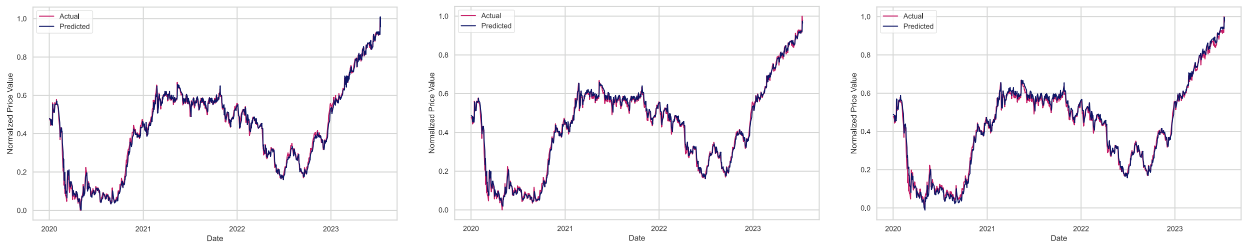


(a) AXP

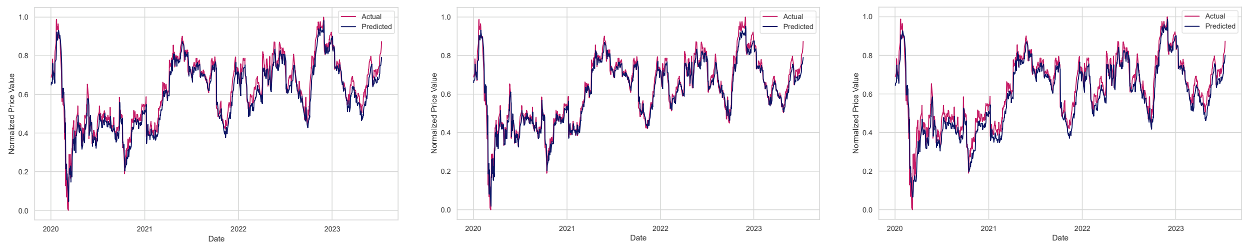


(b) CSCO

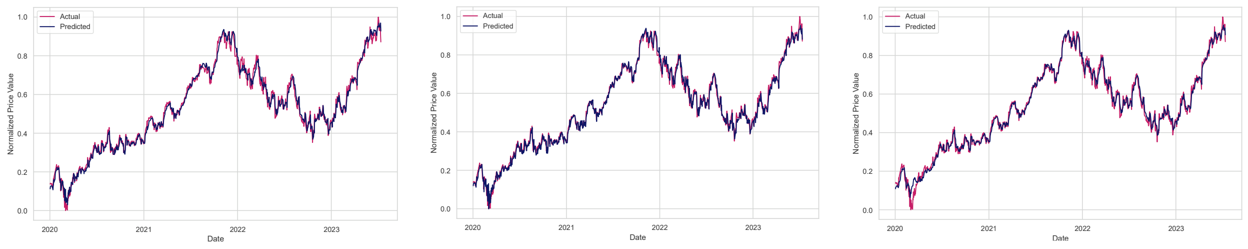
Figure 3: The forecasting results of the preprocessing network for the unseen stock price data of AXP and CSCO. Each column from left to right represents models A, B, and C, respectively. See Fig. 4 for GE, IBM and MSFT stocks.



(a) GE



(b) IBM

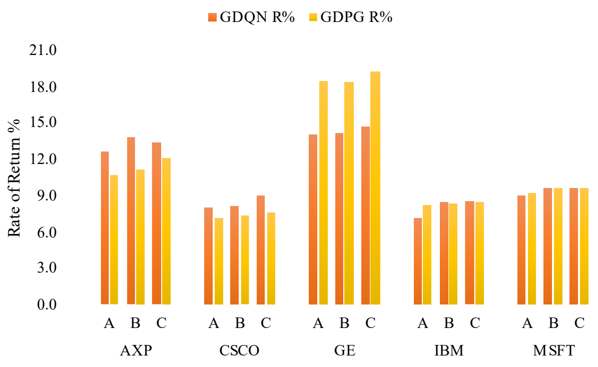


(c) MSFT

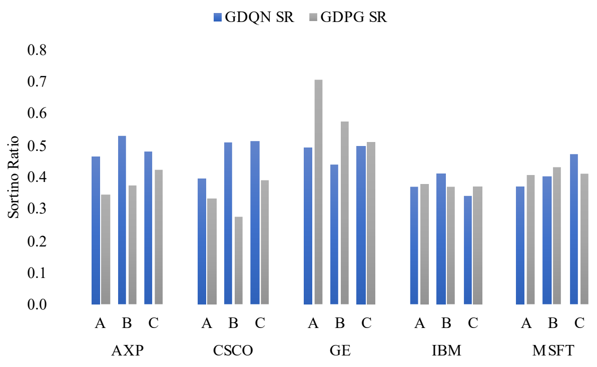
Figure 4: The forecasting results of the preprocessing network for the unseen stock price data of GE, IBM and MSFT. Each column from left to right represents models A, B, and C, respectively. See Fig. 3 for AXP and CSCO stocks.

Table 4: Training hyperparameters.

	GDQN	GDPG
Hyperparameters	Value	
Optimizer	Adam	Adam
History length	10	10
Learning rate	0.001	0.001
Discount factor	0.991	0.994
Batch size	128	128
Episodes	2000	2000
Steps	150	150
Epsilon max	1	1
Epsilon min	0.1	0.1
Epsilon decay rate	0.9972	0.9972
Memory capacity	1000000	700000
Hard update frequency	2000	-
Soft update coefficient	-	0.001



(a) Rate of Returns ($\times 10^{-2}$)%



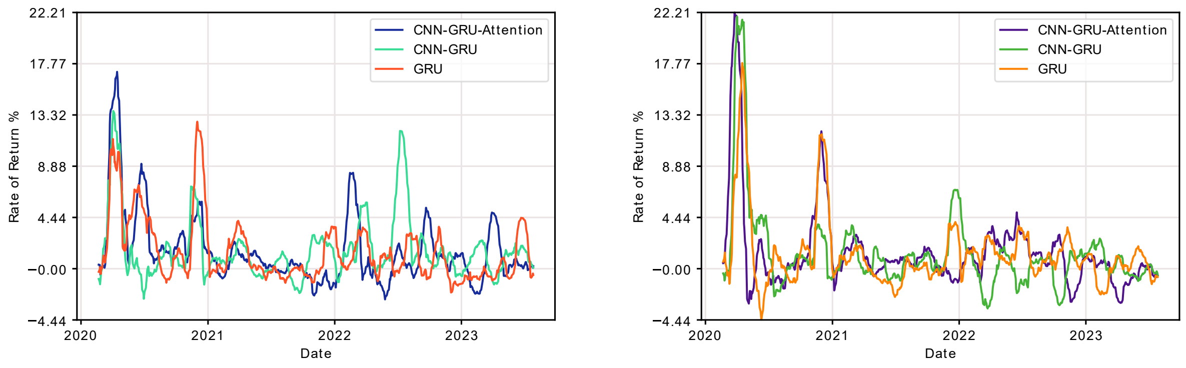
(b) Sortino Ratio

Figure 5: Performance comparison of RL models based on ROR ($\times 10^{-2}$)% and Sortino Ratio.

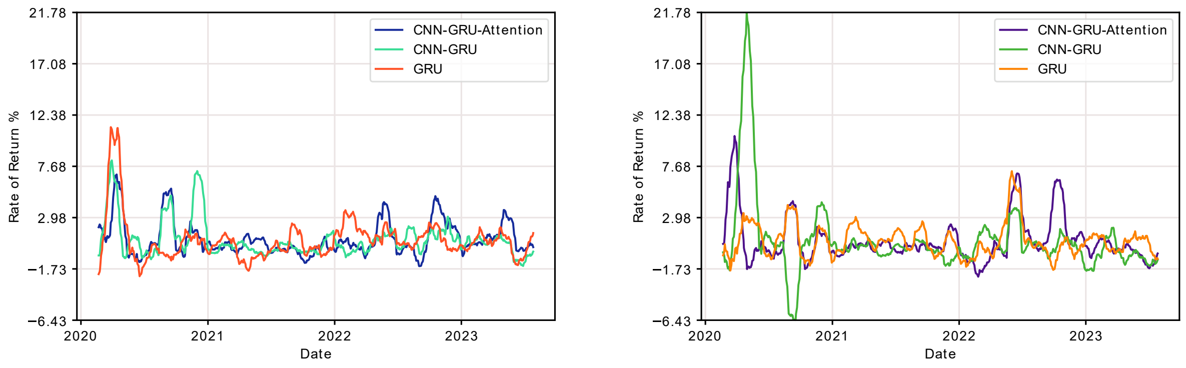
Table 5: Performance of GDQN and GDPG in the U.S. stock market.

Symbol	Model	GDQN		GDPG	
		SR	R(%)	SR	R(%)
AXP	A	0.464	12.555	0.344	10.601
	B	0.530	**13.825**	0.373	11.117
	C	0.480	13.362	0.424	12.091
CSCO	A	0.393	7.994	0.333	7.096
	B	0.508	8.069	0.276	7.301
	C	0.512	**8.870**	0.391	7.544
GE	A	0.493	14.022	0.704	18.456
	B	0.439	14.072	0.574	18.274
	C	0.494	14.591	0.508	**19.146**
IBM	A	0.368	7.044	0.377	8.139
	B	0.410	8.365	0.368	8.278
	C	0.339	**8.463**	0.369	8.410
MSFT	A	0.368	8.914	0.406	9.183
	B	0.402	9.551	0.431	9.532
	C	0.471	**9.612**	0.411	9.529

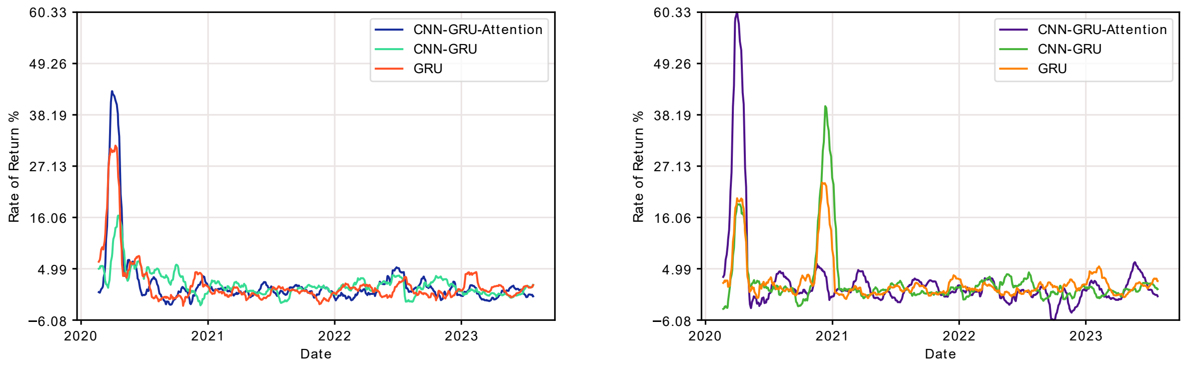
To provide a more comprehensive understanding of the outcomes achieved by utilizing CNN as a feature extractor and incorporating an attention mechanism to assign varying weights to different components, we further examine the performance results of GDQN and GDPG in Figs. 6 and 7. As an example, GDQN outperforms GDPG across all three models for AXP, demonstrating higher ROR and SR metrics. The optimal strategy combines GDQN with model B, as shown in Fig. 5(a). When AXP's stock price rises significantly, both strategies yield good returns, but GDPG excels with GE, where stock price changes are more gradual. GDPG also offers a more stable yield curve compared to GDQN. In cases of sharp stock price fluctuations like IBM, both GDQN and GDPG achieve modest profits. During downtrends, such as CSCO from 2022 onward, both strategies incur losses but still perform relatively well, with GDQN benefiting from attention mechanisms, as illustrated in Figs. 5 and 6(b). The experimental results clearly show that both strategies can be profitable, especially with stocks trending upwards like MSFT, seen in Fig. 7(b). Even in volatile markets like IBM, GDQN and GDPG strategies continue to generate returns. When combined with models B and C, the strategies often result in higher profit generation.



(a) AXP

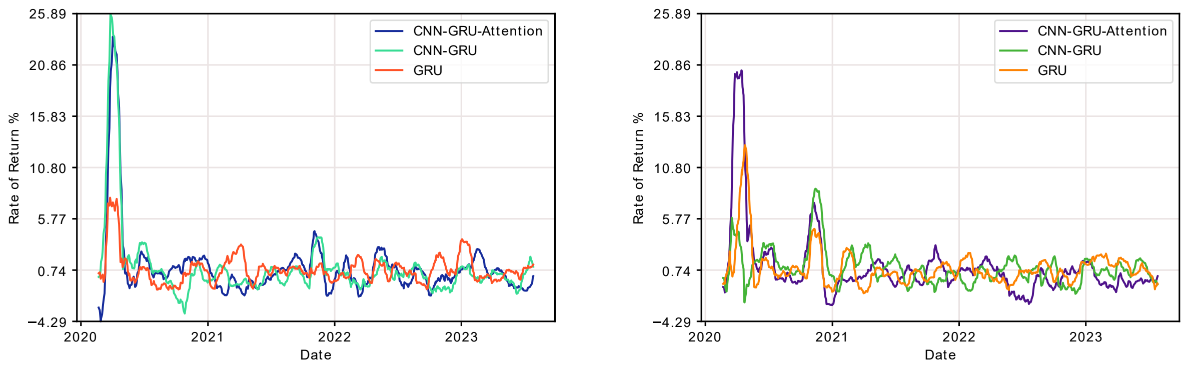


(b) CSCO

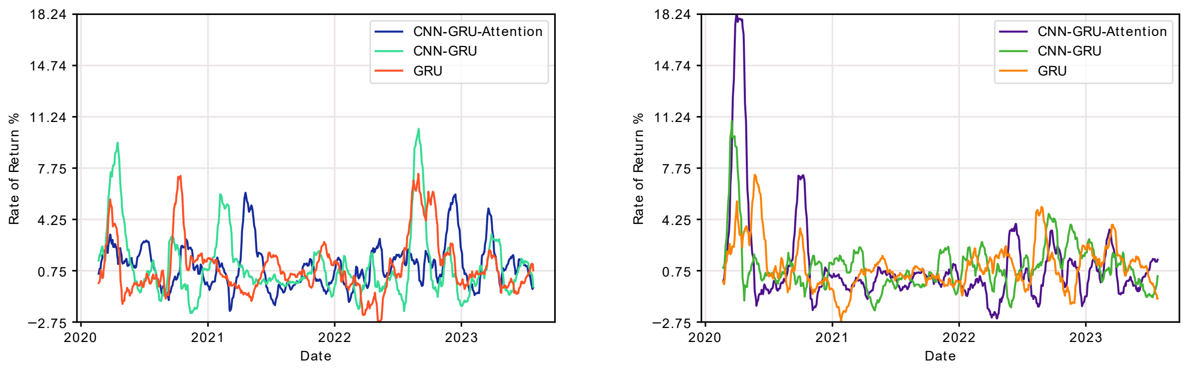


(c) GE

Figure 6: Comparison of GDQN and GDPG for the AXP, CSCO, and GE stocks. Each subfigure illustrates the relative outcome of GDQN and GDPG with a 25-day simple moving average based on the ROR%. See Fig. 7 for IBM and MSFT stocks.



(a) IBM



(b) MSFT

Figure 7: Comparison of GDQN and GDPG for the IBM and MSFT stocks. See Fig. 6 for AXP, CSCO, and GE stocks.

6. Conclusion

This paper utilized two DRL approaches, GDQN and GDPG, to develop profitable stock trading strategies. The proposed preprocessing network models integrate CNN and GRU to extract features and capture temporal dependencies in stock market data. Additionally, an attention mechanism is employed to focus on critical periods in the time series.

Extensive experiments conducted on historical stock data of five U.S. companies demonstrate the advantages of the proposed architectures over solely using GRU. The feature extraction and attention components enable the DRL agent to make better-informed trading decisions, resulting in higher returns and lower risk compared to the GRU baseline. Notably, the GDPG method achieved more satisfactory profits, generating over 19% returns in certain stocks, while the GDQN method achieved over 14% returns with acceptable risk-adjusted ratios.

The proposed models offer a promising approach to managing the dynamics of financial markets. By continuously refining trading policies based on environmental feedback, the agent can adapt to evolving market conditions, thereby overcoming the limitations of conventional predictive modeling.

The proposed models for stock trading have demonstrated profitable results, but it is essential to acknowledge their limitations. One area for future exploration is the extension of this approach to portfolio management, where the agent learns to make trading decisions for a basket of stocks simultaneously. This would involve incorporating diversification, risk management, and asset allocation strategies to optimize portfolio performance. By doing so, the model could provide more comprehensive and robust trading decisions. Furthermore, the current approach relies heavily on a limited feature set, primarily consisting of historical stock prices and technical indicators. However, stock prices are influenced by a multitude of factors, including company fundamentals, macroeconomic conditions, news events, and investor sentiment. To improve the model's predictive capabilities and decision-making process, it is essential to incorporate additional features, such as financial statements, economic indicators, and sentiment analysis data from news articles or social media. This would provide a more comprehensive understanding of the stock's behavior and enable the model to make more informed trading decisions.

The current implementation of the model assumes frictionless trading, neglecting important transaction costs such as brokerage fees, bid-ask spreads, and market impact costs. Furthermore, real-world trading often involves constraints like trading volume limits, order execution delays, and market regulations. To enhance the model's applicability and provide a more realistic assessment of its performance, it is crucial to incorporate these practical considerations into the trading environment. This would provide a more accurate representation of the trading process and enable the model to make more informed decisions. Additionally, it is essential to acknowledge that these findings are based on just five U.S. stock markets. Further research is needed to determine how well these models generalize to a broader range of markets, such as the U.K. or Chinese markets, and the different data preprocessing techniques they might require. To evaluate the robustness and generalizability of the model, it is essential to test it across different market conditions, such as periods of high volatility, market crashes, or bull/bear markets. This would provide valuable insights into the model's performance under varying market conditions and enable the identification of potential areas for improvement.

References

[1] F. G. D. C. Ferreira, A. H. Gandomi, and R. T. N. Cardoso, “Artificial intelligence applied to stock market trading: A review,” IEEE Access, vol. 9, pp. 30898–30917, 2021.

[2] M. R. Roostaee and A. A. Abin, “Forecasting financial signal for automated trading: An interpretable approach,” Expert Systems with Applications, vol. 211, pp. 118570–118583, 2023.

[3] A. M. Rahmani, B. Rezazadeh, M. Haghparast, W.-C. Chang, and S. G. Ting, “Applications of artificial intelligence in the economy, including applications in stock trading, market analysis, and risk management,” IEEE Access, vol. 11, pp. 80769–80793, 2023.

[4] M. M. Kumbure, C. Lohrmann, P. Luukka, and J. Porras, “Machine learning techniques and data for stock market forecasting: A literature review,” Expert Systems with Applications, vol. 197, pp. 116659–116700, 2022.

[5] H. Nasiri and M. M. Ebadzadeh, “Mfrfnn: Multi-functional recurrent fuzzy neural network for chaotic time series prediction,” Neurocomputing, vol. 507, pp. 292–310, 2022.

[6] H. Nasiri and M. M. Ebadzadeh, “Multi-step-ahead stock price prediction using recurrent fuzzy neural network and variational mode decomposition,” Applied Soft Computing, vol. 148, pp. 110867–110883, 2023.

[7] S. Sun, R. Wang, and B. An, “Reinforcement learning for quantitative trading,” ACM Trans. Intell. Syst. Technol., vol. 14, pp. 1–29, 3 2023.

[8] W. Lu, J. Li, Y. Li, A. Sun, and J. Wang, “A cnn-lstm-based model to forecast stock prices,” Complexity, vol. 2020, pp. 1–10, 2020.

[9] Y. Li, P. Ni, and V. Chang, “Application of deep reinforcement learning in stock trading strategies and stock forecasting,” Computing, vol. 102, no. 6, pp. 1305–1322, 2020.

[10] A. Mosavi, Y. Faghan, P. Ghamisi, P. Duan, S. F. Ardabili, E. Salwana, and S. S. Band, “Comprehensive review of deep reinforcement learning methods and applications in economics,” Mathematics, vol. 8, no. 10, pp. 1640–1682, 2020.

[11] M. R. Vargas, C. E. M. dos Anjos, G. L. G. Bichara, and A. G. Evsukoff, “Deep leaming for stock market prediction using technical indicators and financial news articles,” in 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8, 2018.

[12] Y. Hao and Q. Gao, “Predicting the trend of stock market index using the hybrid neural network based on multiple time scale feature learning,” Applied Sciences, vol. 10, no. 11, 2020.

[13] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, “Forecasting stock prices from the limit order book using convolutional neural networks,” in 2017 IEEE 19th Conference on Business Informatics (CBI), vol. 01, pp. 7–12, 2017.

[14] J. Patel, S. Shah, P. Thakkar, and K. Kotecha, “Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques,” Expert Systems with Applications, vol. 42, no. 1, pp. 259–268, 2015.

[15] M. L. De Prado, “The 10 reasons most machine learning funds fail,” The Journal of Portfolio Management, vol. 44, no. 6, pp. 120–133, 2018.

[16] T. Kabbani and E. Duman, “Deep reinforcement learning approach for trading automation in the stock market,” IEEE Access, vol. 10, pp. 93564–93574, 2022.

[17] T. L. Meng and M. Khushi, “Reinforcement learning in financial markets,” Data, vol. 4, no. 3, 2019.

[18] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” IEEE Signal Processing Magazine, vol. 34, no. 6, pp. 26–38, 2017.

[19] L. Hao, B. Wang, Z. Lu, and K. Hu, “Application of deep reinforcement learning in financial quantitative trading,” in 2022 4th International Conference on Communications, Information System and Computer Engineering (CISCE), pp. 466–471, IEEE, 2022.

[20] N. Kodurupaka, H. Basavadeepthi, S. T. Pecheti, and J. Amudha, “Deep reinforcement learning in stock trading: Evaluating ddpg and dqn strategies,” in 2024 International Conference on Emerging Smart Computing and Informatics (ESCI), pp. 1–7, IEEE, 2024.

[21] Y. Chen, “Comparison of deep q-learning network and double deep q-learning network for trading strategy,” in Proceedings of the 4th International Conference on Economic Management and Big Data Applications, ICEMBDA 2023, October 27–29, 2023, Tianjin, China, EAI, 1 2024.

[22] A. Brim, “Deep reinforcement learning pairs trading with a double deep q-network,” in 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), pp. 0222–0227, IEEE, 2020.

[23] H. Zhang, Z. Jiang, and J. Su, “A deep deterministic policy gradient-based strategy for stocks portfolio management,” in 2021 IEEE 6th International Conference on Big Data Analytics (ICBDA), pp. 230–238, IEEE, 2021.

[24] A. R. Azhikodan, A. G. K. Bhat, and M. V. Jadhav, “Stock trading bot using deep reinforcement learning,” in Innovations in Computer Science and Engineering (H. S. Saini, R. Sayal, A. Govardhan, and R. Buyya, eds.), (Singapore), pp. 41–49, Springer Singapore, 2019.

[25] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, “Adaptive stock trading strategies with deep reinforcement learning methods,” Information Sciences, vol. 538, pp. 142–158, 2020.

[26] X. Jiang, “Comparison of deep reinforcement learning algorithms for trading strategy,” in Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing (DAI 2023), pp. 4–14, Atlantis Press, 2024.

[27] T.-V. Pricope, “Deep reinforcement learning in quantitative algorithmic trading: A review,” arXiv preprint arXiv:2106.00123, 2021.

[28] B. Bebeshko, K. Khorolska, and A. Desiatko, “Analysis and modeling of price changes on the exchange market based on structural market data,” in 2021 IEEE 8th International Conference on Problems of Infocommunications, Science and Technology (PIC S&T), pp. 151–156, IEEE, 2021.

[29] A. Picasso, S. Merello, Y. Ma, L. Oneto, and E. Cambria, “Technical analysis and sentiment embeddings for market trend prediction,” Expert Systems with Applications, vol. 135, pp. 60–70, 2019.

[30] N. Rouf, M. B. Malik, T. Arif, S. Sharma, S. Singh, S. Aich, and H.-C. Kim, “Stock market prediction using machine learning techniques: A decade survey on methodologies, recent developments, and future directions,” Electronics, vol. 10, no. 21, 2021.

[31] A. S. Wafi, H. Hassan, and A. Mabrouk, “Fundamental analysis models in financial markets – review study,” Procedia Economics and Finance, vol. 30, pp. 939–947, 2015. IISES 3rd and 4th Economics and Finance Conference.

[32] J. R. Dahlquist and C. D. Kirkpatrick II, Technical analysis: the complete resource for financial market technicians. FT press, 2010.

[33] T. R. C. C. da Costa, R. T. Nazário, G. S. Z. Bergo, V. A. Sobreiro, and H. Kimura, “Trading system based on the use of technical analysis: A computational experiment,” Journal of Behavioral and Experimental Finance, vol. 6, pp. 42–55, 2015.

[34] C. Harrington, “Fundamental vs. technical analysis: Controversy between the two schools is still alive and well,” CFA Magazine. JAN-Feb, pp. 36–37, 2003.

[35] F. Larsen, “Automatic stock market trading based on technical analysis,” Master’s thesis, Institutt for datateknikk og informasjonsvitenskap, 2007.

[36] I. K. Nti, A. F. Adekoya, and B. A. Weyori, “A systematic review of fundamental and technical analysis of stock market predictions,” Artificial Intelligence Review, vol. 53, pp. 3007–3057, Apr 2020.

[37] N. Eiamkanitchat, T. Moontuy, and S. Ramingwong, “Fundamental analysis and technical analysis integrated system for stock filtration,” Cluster Computing, vol. 20, pp. 883–894, Mar 2017.

[38] P. Glabadanidis, Fundamental Versus Technical Analysis, pp. 1–3. New York: Palgrave Macmillan US, 2015.

[39] I. Contreras, J. I. Hidalgo, and L. Núñez-Letamendia, “A ga combining technical and fundamental analysis for trading the stock market,” in Applications of Evolutionary Computation (C. Di Chio, A. Agapitos, S. Cagnoni, C. Cotta, F. F. de Vega, G. A. Di Caro, R. Drechsler, A. Ekárt, A. I. Esparcia-Alcázar, M. Farooq, W. B. Langdon, J. J. Merelo-Guervós, M. Preuss, H. Richter, S. Silva, A. Simões, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, J. Togelius, N. Urquhart, A. Ş. Uyar, and G. N. Yannakakis, eds.), (Berlin, Heidelberg), pp. 174–183, Springer Berlin Heidelberg, 2012.

[40] C. Hargreaves and Y. Hao, “Does the use of technical & fundamental analysis improve stock choice?: A data mining approach applied to the australian stock market,” in 2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE), pp. 1–6, IEEE, 2012.

[41] W. Jiang, “Applications of deep learning in stock market prediction: Recent progress,” Expert Systems with Applications, vol. 184, pp. 115537–115559, 2021.

[42] S. Mokhtari, K. K. Yen, and J. Liu, “Effectiveness of artificial intelligence in stock market prediction based on machine learning,” arXiv preprint arXiv:2107.01031, 2021.

[43] A. Shah, M. Doshi, M. Parekh, N. Deliwala, P. Chawan, and M. Pramila, “Identifying trades using technical analysis and ml/dl models,” arXiv preprint arXiv:2304.09936, 2023.

[44] V. Polepally, N. S. N. Reddy, M. Sindhuja, N. Anjali, and K. J. Reddy, “A deep learning approach for prediction of stock price based on neural network models: Lstm and gru,” in 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 1–4, IEEE, 2021.

[45] G. Bathla, “Stock price prediction using lstm and svr,” in 2020 Sixth International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 211–214, IEEE, 2020.

[46] M. Salemi Mottaghi and M. Haghir Chehreghani, “A deep comprehensive model for stock price prediction,” Journal of Ambient Intelligence and Humanized Computing, vol. 14, pp. 11385–11395, Aug 2023.

[47] H. Haddadian, M. Baky Haskuee, and G. Zomorodian, “An algorithmic trading system based on machine learning in tehran stock exchange,” Advances in Mathematical Finance and Applications, vol. 6, no. 3, pp. 653–669, 2021.

[48] M. Nabipour, P. Nayyeri, H. Jabani, S. S., and A. Mosavi, “Predicting stock market trends using machine learning and deep learning algorithms via continuous and binary data; a comparative analysis,” IEEE Access, vol. 8, pp. 150199–150212, 2020.

[49] M. Nabipour, P. Nayyeri, H. Jabani, A. Mosavi, E. Salwana, and S. S., “Deep learning for stock market prediction,” Entropy, vol. 22, no. 8, 2020.

[50] D. Lien Minh, A. Sadeghi-Niaraki, H. D. Huy, K. Min, and H. Moon, “Deep learning approach for short-term stock trends prediction based on two-stream gated recurrent unit network,” IEEE Access, vol. 6, pp. 55392–55404, 2018.

[51] Q. Zhou, C. Zhou, and X. Wang, “Stock prediction based on bidirectional gated recurrent unit with convolutional neural network and feature selection,” PLOS ONE, vol. 17, pp. 1–20, 02 2022.

[52] H. Li, Y. Shen, and Y. Zhu, “Stock price prediction using attention-based multiinput lstm,” in Proceedings of The 10th Asian Conference on Machine Learning (J. Zhu and I. Takeuchi, eds.), vol. 95 of Proceedings of Machine Learning Research, pp. 454–469, PMLR, 14–16 Nov 2018.

[53] Y. Huang, X. Lu, C. Zhou, and Y. Song, “Dade-dqn: Dual action and dual environment deep q-network for enhancing stock trading strategy,” Mathematics, vol. 11, no. 17, 2023.

[54] N. Yousefi, “Deep reinforcement learning for tehran stock trading,” Journal of Novel Engineering Science and Technology, vol. 1, no. 02, pp. 37–42, 2022.

[55] C. Y. Huang, “Financial trading as a game: A deep reinforcement learning approach,” arXiv preprint arXiv:1807.02787, 2018.

[56] M. Taghian, A. Asadi, and R. Safabakhsh, “A reinforcement learning based encoder-decoder framework for learning stock trading rules,” arXiv preprint arXiv:2101.03867, 2021.
[57] S. Luo, X. Lin, and Z. Zheng, “A novel cnn-ddpg based ai-trader: Performance and roles in business operations,” Transportation Research Part E: Logistics and Transportation Review, vol. 131, pp. 68–79, 2019.
[58] W. Lu, J. Li, J. Wang, and L. Qin, “A cnn-bilstm-am method for stock price prediction,” Neural Computing and Applications, vol. 33, no. 10, pp. 4741–4753, 2021.
[59] M. Hausknecht and P. Stone, “Deep recurrent q-learning for partially observable mdps,” in *2015 AAAI fall symposium series*, 2015.
[60] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al., “Human-level control through deep reinforcement learning,” nature, vol. 518, no. 7540, pp. 529– 533, 2015.
[61] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double q-learning,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30, Mar. 2016.
[62] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, “Deterministic policy gradient algorithms,” in Proceedings of the 31st International Conference on Machine Learning (E. P. Xing and T. Jebara, eds.), vol. 32 of Proceedings of Machine Learning Research, (Bejing, China), pp. 387–395, PMLR, 6 2014.
[63] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1509.02971, 2015.
[64] Y. Yu, X. Si, C. Hu, and J. Zhang, “A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures,” Neural Computation, vol. 31, pp. 1235–1270, 07 2019.
[65] A. Thakkar and K. Chaudhari, “A comprehensive survey on deep neural networks for stock market: The need, challenges, and future directions,” Expert Systems with Applications, vol. 177, pp. 114800–114817, 2021.
[66] C. Chen, L. Xue, and W. Xing, “Research on improved gru-based stock price prediction method,” Applied Sciences, vol. 13, no. 15, 2023.
[67] M. Yang, X. Li, and Y. Liu, “Sequence to point learning based on an attention neural network for nonintrusive load decomposition,” Electronics, vol. 10, no. 14, 2021.
[68] E. Hoseinzade and S. Haratizadeh, “Cnnpred: Cnn-based stock market prediction using several data sources,” arXiv preprint arXiv:1810.08923, 2018.

Mahdi Shahbazi Khojasteh received his M.Sc. degree in Computer Engineering from Shahid Beheshti University, Tehran, Iran, in 2024. His primary research interests include machine learning, reinforcement learning, and robotics.

Mohammad Mahdi Setak received his B.Sc. in Software Engineering from Kharazmi University, Tehran, Iran. He is currently a master’s student specializing in Artificial Intelligence at Shahid Beheshti University (SBU), Tehran, Iran. His research interests encompass Computer Vision, Reinforcement Learning, and Deep Learning.

Armin Salimi-Badr *received the B.Sc., M.Sc. and PhD degrees in Computer Engineering, all from Amirkabir University of Technology, Tehran, Iran in 2010, 2012, and 2018 respectively. He also obtained a PhD degree in Neuroscience from University of Burgundy, Dijon, France in 2019, where he was researching on presenting a computational model of brain motor control in the Laboratory 1093 CAPS (Cognition, Action, et Plasticité Sensorimotrice) of the Institut National de la Santé et de la Recherche Médicale (INSERM). He was a Postdoctoral Research Fellow at Biocomputing lab of Amirkabir University of Technology from October 2019 to September 2020. Currently, he is an Assistant Professor at Faculty of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran and also the Head of Artificial Intelligence & Robotics & Cognitive Computing group in this faculty. He is also the founder and Chair of Robotics & Intelligent Autonomous Agents (RoIAA) Lab in Shahid Beheshti University. He is IEEE Senior Member and currently the Chair of Professional Activities Committee and a Board Member of Computer Society of IEEE Iran Section. His research interests include Computational Intelligence, Computational Neuroscience, and Robotics.*