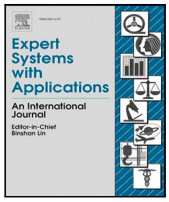
Expert Systems With Applications 237 (2024) 121502

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



A multi-agent reinforcement learning framework for optimizing financial trading strategies based on TimesNet

Yuling Huang, Chujin Zhou, Kai Cui, Xiaoping Lu *

School of Computer Science and Engineering, Macau University of Science and Technology, Taipa, Macao Special Administrative Region of China

ARTICLE INFO

Keywords:
Deep reinforcement learning
Multi-agent reinforcement learning
TimesNet
Multi-scale CNN
Algorithmic trading

ABSTRACT

An increasing number of studies have shown the effectiveness of using deep reinforcement learning to learn profitable trading strategies from financial market data. However, a single-agent model is not sufficient to handle complex financial scenarios. To address this problem, a novel approach called Multi-Agent Double Deep Q-Network (Later called MADDQN) is proposed in this study, which reasonably balances the pursuit of maximum revenue and the avoidance of risk under the multi-agent reinforcement learning framework by innovatively employing two different agents represented respectively by two time-series feature extraction networks, TimesNet, and the Multi-Scale Convolutional Neural Network. Furthermore, to achieve a more generalized model suitable for different underlying assets, a mixed dataset containing three major U.S. stock indexes is collected. And the proposed model has been pre-trained in this dataset and subsequently refined for the specified asset. The results from experiments on five different stock indices show that the proposed MADDQN has an average cumulative return of 23.08%, outperforming the other baseline methods. Besides, the multi-agent model demonstrates its advantage in balancing the risk and revenue, in comparison with the single-agent models. Additionally, The generalization experiments confirm that the proposed MADDQN method after pre-training in the proposed mixed dataset could be stably transferred to the other underlying assets with a refinement. These findings indicate that the proposed framework not only achieves good performance in complex financial market environments but also is able to generalize robustly across different scenarios in various markets.

1. Introduction

Algorithmic trading has received a lot of attention from researchers as the volume of financial data continues to increase. Algorithmic trading involves the use of mathematical models and statistical analysis to exploit market opportunities and make financial trading decisions. This approach differs from traditional trading strategies by collecting and processing more information, including price, volume, and volatility, and by using powerful computers to generate trading signals quickly and efficiently (Hendershott et al., 2011; Nuti et al., 2011; Treleaven et al., 2013). The primary benefit of algorithmic trading is its elimination of emotional decision-making during trading, along with the ability to optimize the utilization of backtest data. However, its major drawback is that the trading strategy's effectiveness could diminish if market conditions change. Consequently, developing a stable real-time trading strategy is highly valuable. Algorithmic trading methods are classified into two main categories: rule-based methods and machine learning-based methods. Rule-based algorithmic trading involves predefined rules created by humans to trade in financial markets. Rules of this nature can be formulated using conventional trading approaches or mathematical models, such as algorithms relying on price momentum, liquidity, and market sentiment. Conversely, algorithmic trading based on machine learning utilizes historical data for training and can autonomously engage in market transactions without direct human involvement. Techniques like Recurrent Reinforcement Learning (RRL) (Moody & Saffell, 1998, 2001) and Trading Deep Q-Network (TDQN) (Théate & Ernst, 2021) are employed in this context.

Deep learning-based algorithmic trading methods can automatically optimize investment decisions by analyzing historical data to extract patterns and knowledge (Wang & Yan, 2021). These methods can help investors make informed decisions and achieve better returns. With the improvement of deep learning algorithms and the development of high-performance computers, deep reinforcement learning algorithms have been applied in the field of algorithmic trading (Liu et al., 2020a), resulting in real-time automated trading strategies for single-asset trading. These algorithms can learn and improve strategies

* Corresponding author.
E-mail addresses: 2109853gia30003@student.must.edu.mo (Y. Huang), 3220002751@student.must.edu.mo (C. Zhou), 2109853nia30001@student.must.edu.mo (K. Cui), xplu@must.edu.mo (X. Lu).

https://doi.org/10.1016/j.eswa.2023.121502
Received 13 July 2023; Received in revised form 21 August 2023; Accepted 6 September 2023
Available online 11 September 2023

while interacting with a dynamic environment, with the goal of discovering profitable patterns and developing better strategies for long-term profitability. In deep reinforcement learning, the Q function represents the expected cumulative reward that an agent can obtain by taking a particular action in a given state and following a particular strategy thereafter.Q networks are designed to learn an agent's optimal action selection strategy by estimating the Q value (expected cumulative reward) for different state–action pairs. The Q-Network structure is a key part of deep reinforcement learning algorithms, which has an important impact on the final results of the model, in which combining different neural networks, such as Multilayer Perceptron (MLP) (Li et al., 2019a), Recurrent Neural Network (RNN) (Chen & Gao, 2019; Nan et al., 2022; Taghian et al., 2022) and Convolutional Neural Network (CNN) (Dang, 2020), to construct the network structure of the Q function has aroused great interest. Existing methods for financial time series feature extraction, such as linear models and traditional neural networks, may struggle to capture complex and nonlinear relationships in the data, leading to a poor performance on prediction tasks due to over-fitting or under-fitting issues. In contrast, TimesNet, a state-of-the-art time series feature extraction network, is specifically designed to process time series data (Wu et al., 2023). It uses a new multi-periodicity perspective to analyze time-series changes, decomposes them into different periods, and achieves unified modeling of intra-periodic and inter-periodic changes by transforming the original one-dimensional time series into two-dimensional space. In this paper, the TimesNet network is taken as a representation of agent.

Existing trading strategies often rely on single-agent models or algorithms that may be effective under certain market conditions but fail to perform well under changing market conditions or in the presence of unexpected events (Chen & Gao, 2019; Cheng et al., 2021; Dang, 2020; Gao, 2018; Nan et al., 2022; Si et al., 2017; Taghian et al., 2022). This can lead to suboptimal trading decisions and lost profits. On the other hand, multi-agent deep reinforcement learning is a powerful technique that combines multiple models or agents that can adapt to different market conditions and learn from each other. This can lead to more robust and effective trading strategies that can generate higher profits and minimize losses.

In order to tackle this issue, the problem of algorithmic trading is framed as a multi-agent system consisting of multiple agents capable of self-learning and interaction. More specifically, agents trained on distinct investment preference frameworks can acquire knowledge about the interplay among different frameworks by collaborating, ultimately leading to the development of more comprehensive investment strategies. This paper introduces a novel multi-agent deep reinforcement learning framework based on the TimesNet network for learning the interactions between different investment preferences through the collective intelligence of multiple agents, each representing a specific investment preference. As far as we know, the Multi-Agent Double Deep Q-Network (MADDQN) has not been previously studied in the literature. The study includes several steps: first, a deep reinforcement learning model based on TimesNet network for stock trading is proposed. Second, the Double DQN (DDQN) algorithm is introduced to train each agent independently. Third, a multi-agent deep reinforcement learning framework is proposed, and an algorithm is designed to train multiple agents to collaborate. Additionally, a multiple stock index dataset is created to train a generic TimesNet-based model and transfer it to other asset trading scenarios. Finally, a series of experiments is performed to assess and validate the effectiveness of the proposed framework in comparison to single-agent systems and other benchmark algorithms. These experiments serve to showcase the superiority of our framework in terms of performance and overall results.

There are four main contributions of this study:

- A novel deep reinforcement learning model based on CNN time-series feature extraction network TimesNet is proposed, which is applied in deep reinforcement learning for the first time and results in better performance.
- A framework is proposed, which combines multiple agents with different investment preferences to integrate strategies through a hierarchical structure to improve the performance of trading strategies. The experimental results on real datasets show that the trading strategies developed using the framework have demonstrated superiority over both single-agent approaches and other benchmark algorithms.
- For better transferring the trained model to other targets, a mixed dataset containing the three major U.S. stock indexes is created. By training the proposed model on this dataset, a generalized multi-agent reinforcement learning model is formed. This model can be transferred to the U.S. stock market individual stock trading scenario and has comparable performance to the traditional baseline model.

The structure of this paper is as follows. In Section 2, earlier studies on the topic is reviewed. Section 3 details the explanation of our proposed model. The experimental results are described and analyzed in Section 4. Finally, Section 5 concludes with a summary of our results and an overview of potential future research areas.

2. Related work

To more clearly discuss the similarities and differences in the literature covered by the proposed approach, two branches of the literature are reviewed: temporal feature extraction networks research in finance and deep reinforcement learning in algorithmic trading. By dividing the related work into different branches, it is possible to present research in different areas related to the topic in a focused and organized manner.

2.1. Temporal feature extraction networks in finance

Financial time series are challenging to analyze due to their high volatility and uncertainty. However, deep learning techniques, particularly deep neural networks, have been developed and applied to these data, enabling the extraction of various patterns. This has important implications for investment decisions, market analysis, financial engineering, and other fields. To extract temporal features from financial time series, researchers commonly use a combination of MLP, Long Short Term Memory networks (LSTM), and Convolutional Neural Network (CNN) (Ahmed et al., 2022; Sezer et al., 2019). For example, Wen et al. proposed a new approach that simplifies financial time series with noise by using frequent patterns and a CNN to capture the structure of the series (Wen et al., 2019). Wu et al. introduced the Stock Sequence Array Convolutional LSTM (SACLSTM) framework that combines CNN and LSTM to achieve more accurate stock price predictions (Wu et al., 2021a). Huang et al. improved the Variational Mode Decomposition (VMD) and LSTM-based financial data forecasting model by adding an input generation step to predict future data (Huang et al., 2021). Lin et al. used a mixture model of LSTM and Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) to forecast stock index prices such as the Standard & Poor's 500 index (S&P500) and China Securities 300 Index (CSI300) (Lin et al., 2021), while Liang et al. used candlestick similarity analysis to predict price trends and achieved favorable results (Liang et al., 2022).

To achieve more precise description of financial time-series, some studies have focused on processing characteristic variables. For instance, Yang et al. introduced a new neural network called Attention enhanced Compound Neural Network (AComNN) which includes Artificial Neural Network, LSTM, and self-attention to capture the time-zone-dependent context of financial time-series data from various regions through multiple filters (Yang et al., 2021). Chen used an attention mechanism to fuse variables from different series, such as market macrodynamic and multiseed series, across time and sequence features (Chen, 2022). Kirisci et al. proposed a CNN-based forecasting model to prevent loss of information during image transformation and

improve forecasting results (Kirisci & Cagcag Yolcu, 2022). Additionally, some models such as multi-scale CNNs, multi-scale local cues, and hierarchical attention-based LSTM have been developed to capture underlying patterns and maximize profit in stock investment. Teng et al. developed the Multi-scale Local Cues and hierarchical Attention-based LSTM (MLCA-LSTM) model to capture underlying price trend patterns and achieve maximum profit in stock investment (Teng et al., 2022).

Recently, several studies aim to develop versatile and modular deep learning models for time series prediction. Wu et al. proposed two architectures, Autoformer (Wu et al., 2021b) and TimesNet (Wu et al., 2023), both of which have modular structures and are capable of handling complex time series data. Autoformer uses a decomposition approach with an auto-correlation mechanism, while TimesNet employs TimesBlock, an inception module that can discover multi-periodicity adaptively and extract complex temporal variations from transformed 2D tensors. Cao developed Wavelet Decomposition Transformer (Dwtformer), which combines the auto-correlation mechanism from Autoformer with a Transformer architecture and a 2D feature learning module based on wavelet decomposition. The 2D module captures complex period variations, while the Transformer captures long-term historical details (Cao & Zhao, 2023). These models have demonstrated good results in predicting various time series and can be useful tools in finance, economics, and engineering for better understanding and predicting time series data.

2.2. Deep reinforcement learning in algorithmic trading

Reinforcement learning begins to be applied to algorithmic trading systems since Moody's first application of recurrent reinforcement learning to financial asset in 1999 (Moody & Saffell, 1998, 2001). With the development of deep reinforcement learning, more researchers are using it for algorithmic trading. Trading models have been classified into three types: value-based, policy-based, or actor-critic-based, depending on the type of trading strategy used. Value-based methods are the most popular and have combined various neural networks, such as MLP, RNN, and CNN, with different algorithms like DQN (Bajpai, 2021; Chakraborty, 2019; Chen & Gao, 2019; Cheng et al., 2021; Dang, 2020; Gao, 2018; Jeong & Kim, 2019; Lei et al., 2019; Li et al., 2019a; Nan et al., 2022; Si et al., 2017; Taghian et al., 2022), DDQN (Brim & Flann, 2022; Chakole & Kurhekar, 2020; Dang, 2020; Li et al., 2022, 2020, 2019a; Liu et al., 2023; Shi et al., 2021; Théate & Ernst, 2021), Deep Recurrent Q-Network (DRQN) (Chen & Gao, 2019; Huang, 2018; Lei et al., 2019; Ma et al., 2021; Zhou & Tang, 2021), Dueling DQN (Bajpai, 2021; Dang, 2020; Li et al., 2022, 2020, 2019a; Shin et al., 2019), and DQN-Hindsight Experience Replay (DQN-HER) (Cornalba et al., 2022) to achieve optimal trading policies, as well as generate trading signals and positions. These studies have shown that value-based deep reinforcement learning methods have potential for creating automated investment models, enhancing stock investment strategies, and implementing artificial intelligence in the financial investment domain.

Policy-based reinforcement learning methods are another approach for developing trading agents. Chen et al. used historical trading data to create an agent that learns from the strategies of expert traders to make profitable decisions (Chen et al., 2018). Lele et al. applied On-policy Reinforcement Learning Algorithms to trade stocks for three companies and concluded that Trust Region Policy Optimization (TRPO) was the best algorithm (Lele et al., 2020). Liu et al. proposed a framework that employs Proximal Policy Optimization (PPO) algorithms and deep neural networks to learn optimal bitcoin trading policies (Liu et al., 2021). Wang et al. introduced a parallel-network continuous quantitative trading model with Generalized AutoRegressive Conditional Heteroskedasticity (GARCH) and PPO that performed better than other models in terms of profit rate while reducing transaction risk (Wang et al., 2021). Sagiraju et al. trained reinforcement learning agents using four algorithms and used the Sharpe ratio to select the best performing agent for different market conditions (Sagiraju & Shashi, 2021). Li et al. presented a PPO Enhancement Strategy to improve the performance of a base stock timing strategy (Li & Chen, 2021). Mahayana et al. proposed a deep reinforcement learning model for algorithmic trading of cryptocurrencies (Mahayana et al., 2022). Tsai et al. proposed an explainable approach called "financial vision" that combines candlestick pattern detection with trading signals to develop investment strategies (Tsai et al., 2022). Xiao investigated the relationship between bitcoin and gold and developed a quantitative investment strategy using deep reinforcement learning with PPO and LSTM (Xiao, 2023).

In addition, there are various actor-critic reinforcement learning techniques that are applied to single asset trading. A few of the popular ones include Advantage Actor Critic (A2C), Asynchronous Advantage Actor-Critic (A3C), Deep Deterministic Policy Gradient (DDPG), Soft Actor-Critic (SAC), and Twin Delayed DDPG (TD3). Ponomarev et al. used an asynchronous advantage actor-critic method with neural network architectures for trading a fixed volume of a financial instrument on the stock exchange (Ponomarev et al., 2019). Li et al. proposed a novel framework for algorithmic trading that uses SDAEs to tackle noisy financial data and incorporates LSTM units to extend DQN and A3C (Li et al., 2019b). Liu et al. presented a deep reinforcement learning library for training and evaluating deep reinforcement learning models using historical financial data (Liu et al., 2020b), while Wu et al. proposed Gated DQN (GDQN) and Gated Deterministic Policy Gradient (GDPG) methods for quantitative stock trading (Wu et al., 2020). Yuan et al. introduced Data Augmentation based Reinforcement Learning (DARL) to address insufficient training data in financial deep reinforcement learning (Yuan et al., 2020), and Lima et al. proposed Sentiment-Aware RL (SentARL) to identify market sentiment momentum (Lima Paiva et al., 2021). Vishal et al. used Actor-Critic deep reinforcement learning algorithms like PPO, DDPG, A2C, and TD3 to create trading agents for the Indian Stock Market (Vishal et al., 2021), while Ge et al. explored the use of deep reinforcement learning methods to automate trading of a single stock using the Shanghai Composite Index as a case study (Ge et al., 2022). Lastly, Liu et al. utilized deep reinforcement learning to increase trading profits by combining financial data, news sentiment data, and technical indicator data (Liu et al., 2023).

In contrast to the single-agent approaches, there have been efforts by researchers to extend reinforcement learning to multi-agent scenarios. Lee et al. introduced a framework that utilizes multiple cooperative agents to combine global trend prediction with local trading strategies. This approach aims to enhance trading performance by enabling the agents to share training episodes and learned policies. Through their framework, Lee et al. demonstrated improved trading performance compared to single-agent models by leveraging the collective intelligence of multiple agents working together (Lee & O, 2002). Carta et al. introduced a multi-layer and multi-ensemble stock trader that uses deep neural networks to pre-process data and a reward-based classifier as a meta-learner to generate stock signals, with multiple trading agents making the final decision (Carta, 2021). Shavandi et al. presented a hierarchical and robust multi-agent deep reinforcement learning framework that can handle noise in financial time series data by training each agent in a specific time frame using the DQN algorithm (Shavandi & Khedmati, 2022). He et al. proposed a Multi-agent Virtual Market (MVMM) model that includes multiple Generative Adversarial Networks (GANs) cooperating with each other to reproduce market price changes, and allows the action of the trading agent to be superimposed on the current state to generate an action-dependent next state (He et al., 2023).

3. Method

In algorithmic trading, the selection and execution process of trading strategies can be regarded as a reinforcement learning problem.

The agent, which is a computational model with decision-making and learning abilities, learns the best trading strategy and execution method by trying various actions in the trading environment. The state of the environment is described by factors such as price and volume, and the agent can take actions defined in a set of discrete options, $A = \{-1, 0, 1\} = \{\text{Sell, Hold, Buy}\}$, and each action is rewarded or punished accordingly. This study develops reward functions for agents with different preferences, including the Sharpe ratio reward function that considers both risk and return for Risk Agent, the mid-term return reward function for Return Agent, and the profit reward function for Final Agent. The expressions of these three reward functions are as follows:

$$\text{Risk Agent Reward}_t = \text{POS}_t \cdot \frac{\text{mean}(R_t^n)}{\text{std}(R_t^n)},$$

$$\text{Return Agent Reward}_t = \text{POS}_t \cdot \frac{p_{t+n} - p_t}{p_t} \cdot 100,$$

$$\text{Final Agent Reward}_t = \text{POS}_t \cdot \frac{p_{t+1} - p_t}{p_t} \cdot 100.$$

where $R_t^n = \left[\frac{p_{t+1}-p_t}{p_t}, \frac{p_{t+2}-p_t}{p_t}, \ldots, \frac{p_{t+n}-p_t}{p_t}\right]$ represents the rate of returns for the next n days starting from day t; p_t is the closing price of the asset on the tth day; POS_t denotes the position information for the tth day; n represents the number of days in the future.

Although this paper thoroughly discusses the stock trading decision problem, some reasonable assumptions need to be made to draw reasonable conclusions. First, it is necessary to assume that the market is perfectly efficient, i.e., the market price reflects all available information independent of any factors outside the market. Second, trading orders from agents should not be sufficient to affect market prices. In addition, it is important to assume that there is no slippage in the execution of orders, i.e., that all market assets are sufficiently liquid to complete the transaction at the final price. Only when all these assumptions are met can reasonable conclusions and results be drawn.

3.1. Overview of the proposed trading system

In this section, a brief overview of the proposed single asset trading system, which consists of the multi-agent deep reinforcement learning framework and its associated policy network, is presented. Multi-agent reinforcement learning is a specialized implementation of reinforcement learning within the context of multi-agent systems. In this framework, numerous agents engage in the process of learning and decision-making within an environment to optimize their cumulative rewards. Each individual agent employs reinforcement learning methodologies such as Q-learning, policy gradients, or actor-critic methods to acquire strategies that are either optimal or very close to being optimal for their respective decision-making tasks. The nature of this interaction can vary based on factors such as the specific environment and the goals of the agents, encompassing cooperation, competition, or a mixture of both (Busoniu et al., 2006). Fig. 1 presents the structure of MADDQN.

First, the time series data of a given asset is preprocessed into a vector of window length × feature dimension. The goal of a reinforcement learning agent is to select the action that yields the highest benefit over time based on a set of observations that describe the current state of the environment.

As presented in Fig. 1, in the proposed framework, the framework consists of multiple independent agents trained to improve financial trading performance. These agents work together to make trading decisions based on their individual learned preferences, with the aim of achieving the best overall performance.

In terms of the sub-agent perspective of the proposed framework, the agent has a structure consisting of two different investment preference, namely Risk Agent (considering both risk and return) and Return Agent (considering mid-term returns). The agent is trained using the double DQN, and responsible for accurately perceiving the corresponding state representation according to the sub-agent policy network and generating Q-values for the three actions. DQN, as one of the reinforcement learning algorithms, operates as an off-policy value-based technique (Mnih et al., 2013). It addresses the challenge of unstable convergence often associated with approximating learning state–action value functions. This is achieved through the amalgamation of Q-learning and deep learning, leveraging two crucial methodologies: experience replay and target networks. Experience replay reduces the correlation between training data by randomly sampling batches of data points from a stored buffer of the most recent N data points. This randomness aids in training data diversity, enhancing learning stability. The algorithm stores a limited number of the latest data points and selects a batch of these points for model fitting during each step or iteration.

The three actions of Q-values from each sub-agent and the current state are then used to construct the global state, which is the information already extracted by sub-agents in the state representation and is the input of the final agent. The final agent is responsible for constructing the investment decision according to final agent policy network by accurately perceiving the global state.

In the financial trading context, each agent is trained to make trading decisions based on its specific investment preference, and the framework combines the individual investment preferences of the agents to achieve the best overall performance.

3.2. MADDQN for decision-making

The proposed approach is a multi-agent deep reinforcement learning framework designed to improve financial trading performance. The framework is composed of several independent agents, each of which is trained using the Double DQN to maximize a specific investment preference. The final agent takes in the Q values from the sub-agents and combines them to make trading decisions. The agents work together to make trading decisions based on their individual learned preferences, with the goal of achieving the best overall performance. Fig. 2 presents how a trading strategy is created using deep reinforcement learning.

It leverages the collective intelligence of multiple agents by training them independently to optimize specific investment preferences. The framework is composed of several independent agents, each of which is trained to maximize a specific investment preference. Agents consist of two types: sub-agents and final agent. Sub-agents have no difference with agents in single-agent deep reinforcement learning, which take the state as input and generate actions by computing a Q function. The final agent, however, has a slight difference. It not only takes the state as input but also takes Q-values from sub-agents.

Assuming that there are Q-values from each sub-agent $Q_1, Q_2, \ldots, Q_n$, the Q-function of the final agent A_F can be represented as

$$\begin{aligned} &Q_F(s, Q_1, Q_2, \ldots, Q_n, a_F; \theta_F) = \\ &\mathbb{E}[G|S = \text{concat}(s, Q_1, Q_2, \ldots, Q_n), A = a_F], \end{aligned} \tag{1}$$

where G is the cumulative reward, a_F is the action taken by the final agent, and θ_F is the neural network of the final agent.

The agents are trained independently and utilize the double Deep Q Network (double DQN) framework to generate their optimal policy (Hasselt et al., 2015). Double DQN is a variant of the Deep Q-Network (DQN) algorithm used in deep reinforcement learning. The Double DQN algorithm improves upon the original DQN algorithm by addressing the problem of overestimation of action values. In the DQN algorithm, the agent uses the same neural network to both estimate the value of the current state–action pair and to select the next action. This can lead to overestimation of the action values, as the agent may assign a high value to an action that is actually not optimal.

To address this issue, the Double DQN algorithm uses two neural networks: a primary network θ and a target network θ^-. Every time the agent takes an action a, it formalizes a transition (s, a, r, s'), where s is the current state, r is the reward computed by the task-specific reward function, and s' is the next state. With the transition, the primary

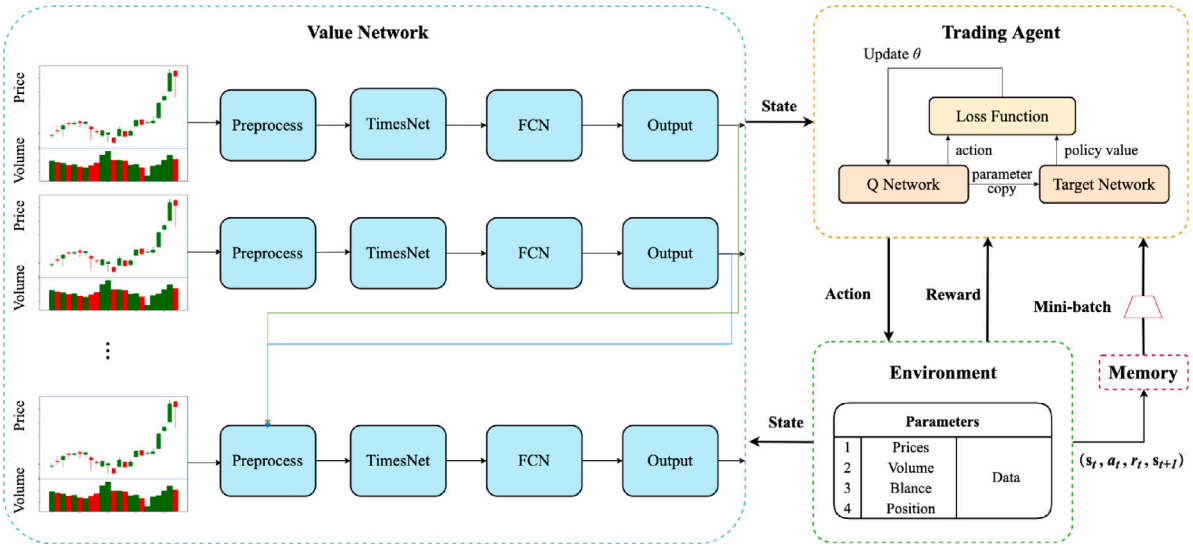


Fig. 1. The structure of MADDQN.

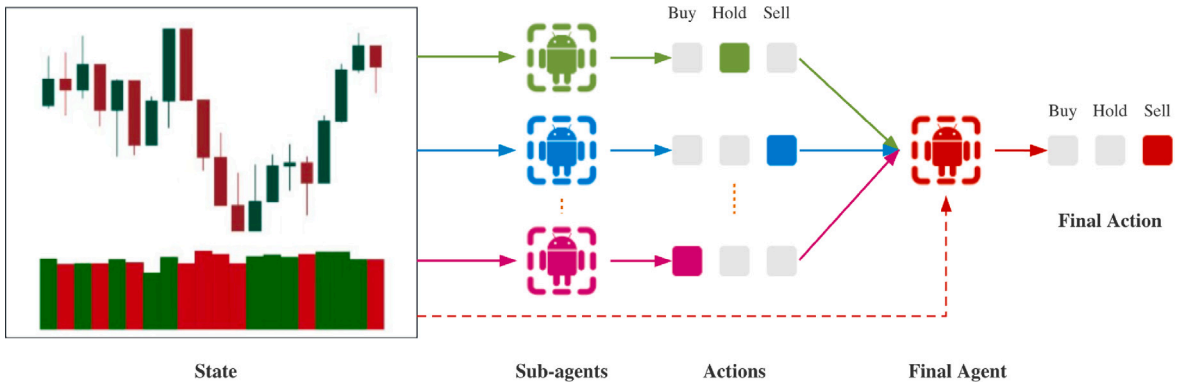


Fig. 2. The trading strategy based on the proposed MADDQN.

network θ selects the next action $a_{\max}$. The formula can be represented as:

$$a_{\max}(s';\theta) = \text{argmax}_{a'} Q(s', a'; \theta), \tag{2}$$

Target network θ^-, on the other hand, estimates the Q value with the formula $Q(s', a_{\max}; \theta^-)$. With the estimation from the target network, the double DQN algorithm can calculate the temporal difference target. This target signifies an estimation of the anticipated cumulative reward that the agent employs to refine its value function during updates:

$$y = r + \gamma Q(s', a_{\max}(s';\theta); \theta^-), \tag{3}$$

where γ is a discounted factor. Finally, the primary network performs gradient descent with loss

$$L = \|y - Q(s, a; \theta)\|^2, \tag{4}$$

The target network is periodically updated to match the weights of the primary network, which helps to stabilize the learning process and reduce overestimation of action values.

In the context of financial trading, each agent is trained to make trading decisions based on its specific investment preference. By combining the individual investment preferences of the agents, the framework can make trading decisions that balance risk and reward to achieve the best overall performance. This can be especially useful in markets where volatility and uncertainty are common, as the framework can adapt to changing market conditions and optimize trading strategies accordingly. This approach has the potential to help traders optimize their investment strategies by leveraging the collective intelligence of multiple agents with different investment preferences. By combining the strengths of different agents, the framework can lead to more robust and effective trading decisions, ultimately improving financial trading performance.

3.3. Policy network structure for agents

This section provides a detailed explanation of the policy network structure of the sub-agent and final agent of the proposed MADDQN.

3.3.1. TimesNet for sub-agent

The proposed policy network designed for the sub-agent is responsible for accurately perceiving the corresponding state representation and generating Q-values for the three actions. The detailed architecture of the proposed network is presented in Fig. 3. The TimesNet network is a temporal feature extraction network that is designed for use in stock trading tasks. The network is made up of four main blocks: Input Block, Embedding Block, TimesBlock, and Action Block.

Input Block: The Input Block of the network takes in the input features, which include the open price, high price, low price, closing price, and trading volume. These features are represented as a tensor with shape $[B, T, N]$, where B is the batch size, T is the window length, and N is the number of features.

Embedding Block: The Embedding Block is responsible for converting the input data into a format that can be processed by the network.

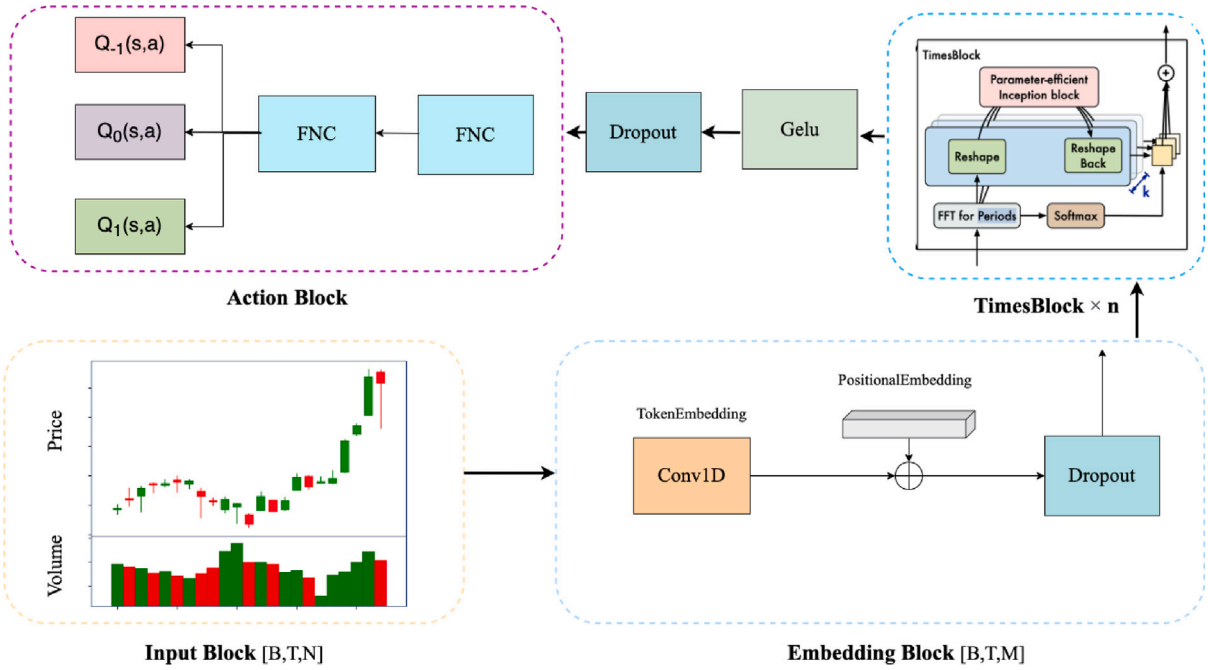


Fig. 3. The structure of TimesNet.

This is achieved by using a 1-D convolution to change the channel of the input data. A positional embedding parameter is then added to the vector, and a dropout layer is used to prevent overfitting. In the proposed algorithm, the number of channels of the input vector is 5, and if the number of channels is insufficient, the feature information will be too little. Therefore, the embedding layer is used to increase the channel of the input vector. When the vector length is very long, it is difficult for distant features to interact, so position encoding is added to increase the position information of features. The position encoding method described in this paper uses a sine and cosine function to map the pixel values onto a plane and determines the position of each pixel value by the extreme value points of the sine and cosine function. This approach increases the long-distance correlation between vectors and calculates the position by the adjacent values of the sine function, thus improving the generalization ability and robustness of the model.

$$\hat{X}_{\text{channel}=M} = \text{Embed}(X_{\text{channel}=N}), \quad M > N \tag{5}$$

TimesBlock: The TimesBlock is the backbone feature extraction part of the network. It takes the input X_{1D} and transforms it into the frequency domain using Fourier transform. The top k frequency components with the largest amplitude in the frequency domain data are then selected, and the periods corresponding to these frequencies are taken out as T_i ($i \in [0, k]$). X_{1D} is then sequentially folded according to T_i into a 2-D vector, which is processed by a 2-D vision backbone, such as ResNet, ConvNeXt, or MobileNet. The output of the backbone processing is reshaped into a 1-D vector with the same shape as X_{1D}. These 1-D vectors are combined into a vector through the Weighted Summation Method, where the weight is the top k amplitudes obtained by the Fourier transform. The output is defined as

$$\text{output} = \hat{X}_{1D} + X_{1D}, \tag{6}$$

where the residual structure is used to ensure that the network depth can be increased.

Action Block: The Action Block is used to convert the features into the final required output format. In the stock trading task, the unique output value is obtained through two fully connected layers to obtain Q values for the three actions. This enables the network to predict the optimal action to take based on the input features.

Overall, the TimesNet network is a powerful tool for extracting temporal features from stock trading data. Its four main blocks work together seamlessly to enable accurate predictions of stock market trends and optimal actions to take based on these trends.

3.3.2. *Multi-scale CNN for final agent*

For final agent, it takes a more global perspective of the current state and the policies of the sub-agents. In this paper, a convolutional neural network with local connectivity and weight sharing is used to not only reduce the overfitting phenomenon but also to optimize the network by reducing the amount of weights using shared weights. Therefore, the MS-CNN is utilized to approximate the state–action function of the final agent. The MS-CNN comprises four main blocks: Input Block, Multi scale Block, Backbone block, and Action Block. Fig. 4 shows the Multi-scale feature extraction network structure.

Input Block: The Input Block includes the daily open price, highest price, lowest price, close price, and trading volume, along with the Q values of the sub-agent.

Multi-scale Block: The Multi-Scale Block has three feature extraction modules at different scales, namely single-scale, medium-scale, and global-scale feature extraction. The single-scale module uses a 1×3 filter, while the medium-scale and global-scale modules use 3×3 and 5×5 filters, respectively. The resulting features from these modules are combined into an 8-channel 2-D feature layer. In stock trading, people commonly analyze stock trends using fixed time scales, particularly with 3-day and 5-day lines as important reference indicators.

Backbone Block: The Backbone block uses a deep convolutional layer for feature extraction, with traditional 3×3 convolution, batch normalization (BN), activation function, pooling layer, and attention modules added to prevent overfitting.

Action Block: The Action Block of the MS-CNN takes the feature maps generated by the Multi scale Block and transforms them into the final output. To achieve this, the network uses two fully connected layers to convert the feature maps into three parameters.

Thus, the main objective of this network is to calculate the action value function, represented as $Q(s, a; \theta)$, for the final agent in the market environment. The set of weights and biases of the network are represented by θ, and the function approximates the value of taking a particular action in a given state.

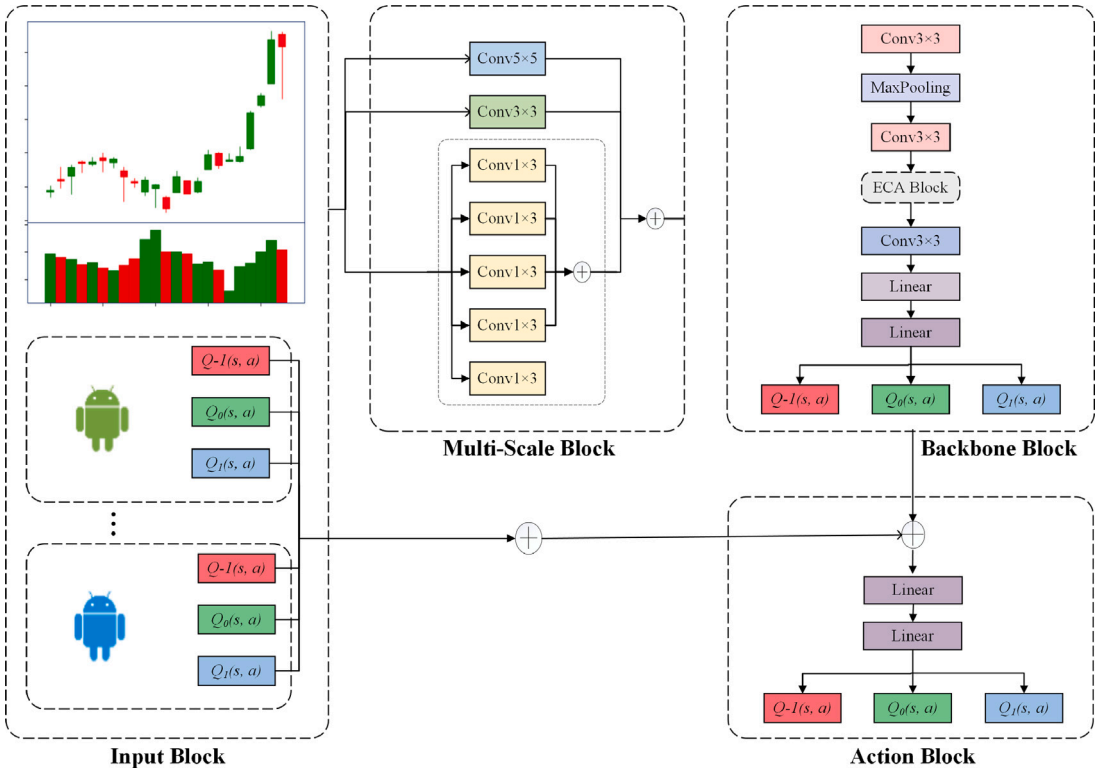


Fig. 4. The structure of multi-scale feature extraction network.

3.4. MADDQN training

The proposed MADDQN involves multiple agents with specific investment preferences working together to assist the final agent in generating the ultimate trading strategy. The primary objective of this framework is to enhance financial trading performance by utilizing the collective intelligence of several agents, which can be more effective than individual agents working alone. The set of sub-agents is represented by $A = \{\text{agent}_1, \ldots, \text{agent}_J\}$, comprising agents specializing in different investment preferences. As the number of agents J increases, the trading strategy becomes more robust and efficient. The framework also uses a hierarchical feedback transfer mechanism to transfer knowledge and experience from agents with specific investment preferences to agents at the global perspective, creating a super-intelligent trading agent, final agent, to generate the final trading signal. It is worth to notice that the proposed framework does not rely on a particular type or number of sub-agents, which means, the reward functions, network structures and numbers of sub-agents can be changed according to the need of investors.

The training procedure for this framework is presented in Algorithm 1. Initially, a sequence of market features that matches the window length is randomly sampled from real data to create an initial condition for a trading event. Sub-agents with specific investment preferences take actions and receive rewards based on the state, while the final agent takes actions and receives corresponding rewards based on the sub-agent's strategy and the current environment state. At the end of the trading process, the collected market states and trading behaviors are used to optimize the trading strategy by updating the parameters of the respective strategy network.

4. Experiments

In this section, the proposed MADDQN is evaluated with several performance metrics, such as return-based and risk-based. On one hand, the proposed MADDQN is compared on a single independent agent with the proposed MADDQN framework in five stock index datasets in numerical experiments, and also with some of the more popular trading strategies. On the other hand, the proposed model is trained on a multi-sample dataset based on the U.S. stock indices and then transferred to a U.S. single stock trading scenario for the evaluation of relevant metrics and comparison with popular trading strategies.

4.1. Dataset and evaluation method

Five stock indices from different countries and regions and four stocks from the United States were used in the experiment. All dataset we used in this experiment are selected from different regions and economic contexts that are widely recognized and utilized in the financial field. These indices and stocks offer insights into different sectors and regions, allowing us to capture a broader range of market dynamics and help our proposed model to be more generalizable. These country/region stock indices include Hang Seng (HSI) in Hong Kong, CAC40 (FCHI) in France, Nasdaq Composite (IXIC), S&P 500 (SP500) and Dow Jones Industrial Average (DJI) in the US. On the other hand, stocks that used in the experiment include Apple(AAPL), Amazon(AMZN), Google(GOOGL) and Microsoft(MSFT).[1] This datasets contain information from January 1, 2007 to December 31, 2022, which provides a wide range of data for model training.

The complete dataset is divided into training and testing sets, consisting of five-dimensional time series data for each index, including daily opening, low, high, closing prices, and trading volume. The training set spans from January 1, 2007, to December 31, 2020, and is used to train the model parameters. The testing set covers the period from January 1, 2021, to December 31, 2022, and is used to assess the models' performance. Fig. 5 shows the close price curve of each stock indices. Fig. 6 shows the close price curve of each stocks from United States. In all close price curve, orange part represents the training set and blue part represents the test set.

[1] http://finance.yahoo.com

Algorithm 1: MADDQN algorithm.

Input: Number of episodes M, learning rate λ, and number of sub-agents J;
Set final agent as F;
for $j = 1$ *to* J *for each sub-agent* j **do**
  Initialize the policy network (Q_j) with random initial weights θ_j for each sub-agent j;
  Initialize the target network (Q_j^-) with weights $\theta_j^- = \theta_j$ for each sub-agent j;
  Initialize replay buffer B_j for each sub-agent j;
  Initialize the policy network (Q_F) with random initial weights θ_F for final agent F;
  Initialize the target network (Q_F^-) with weights $\theta_F^- = \theta_F$ for final agent F;
  Initialize replay buffer B_F for final agent F;
end
for *episode = 1 to M* **do**
  Initialize sequence $s_1 = \{x_1\}$ and preprocessed state $\phi_1 = \phi(s_1)$;
  for *t=1 to T* **do**
    for *j=1 to J* **do**
      Select $a_{j,t}$ with ϵ-greedy method;
      Otherwise, calculate $Q_j = Q(\phi(s_t), a_j; \theta_j)$;
      Select $a_{j,t} = \text{argmax}_{a_j} Q_j$;
      Sub-agent executes action $a_{j,t}$ in the environment;
      Sub-agent gets reward $r_{j,t}$ and observes next state $s_{t+1}, \phi_{t+1} = \phi(s_{t+1})$;
      Store the transition $(\phi_t, a_{j,t}, r_{j,t}, \phi_{t+1})$ in B_j;
    end
    Select $a_{F,t}$ with ϵ-greedy method;
    Otherwise, select $a_{F,t} = \text{argmax}_{a_F} Q(\phi(s_t), Q_1, Q_2, \cdots, Q_j, a_F; \theta_F)$;
    Final agent executes action $a_{F,t}$ in the environment;
    Final agent gets reward r_t and observes next state $s_{t+1}, \phi_{t+1} = \phi(s_{t+1})$;
    for *j=1 to J for each sub-agent* j **do**
      Sample random mini-batch of N randomly selected transitions of $(\phi_i, a_{j,i}, r_{j,i}, \phi_{i+1})$ from B_j;
      Set:

$$y_{j,i} = \begin{cases} r_{j,i}, & \text{if episode terminates at step } i+1, \\ r_{j,i} + \gamma Q_j^-(\phi_{i+1}, a_{j,i}; \theta_j^-), & \text{otherwise.} \end{cases}$$

      Update the sub-agent j by minimizing the loss: $L = E[(y_{j,i} - Q_j(\phi_i, a_{j,i}; \theta_j))^2]$;
      Update the parameters of the target network $\theta_j^- = \theta_j$ every C steps;
    end
    Store the transition $(\phi_t, a_{F,t}, r_{F,t}, Q_{1,t}, Q_{2,t}, \cdots, Q_{j,t}, \phi_{t+1})$ in B_F;
    Sample random mini-batch of N transitions of $(\phi_i, a_{F,i}, r_{F,i}, Q_{1,i}, Q_{2,i}, \cdots, Q_{j,i}, \phi_{i+1})$ from B_F;
    Set:

$$y_{F,i} = \begin{cases} r_{F,i}, & \text{if episode terminates at step } i+1, \\ r_{F,i} + \gamma Q_F^-(\phi_{i+1}, Q_{1,i}, Q_{2,i}, \cdots, Q_{j,i}, a_{F,i}; \theta_F^-), & \text{otherwise.} \end{cases}$$

    Update the final agent F by minimizing the loss: $L = E[(y_{F,i} - Q_F(\phi_i, Q_{1,i}, Q_{2,i}, \cdots, Q_{j,i}, a_{F,i}; \theta_F))^2]$;
    Update the parameters of the target network $\theta_F^- = \theta_F$ every C steps;
    Set $s_t \leftarrow s_{t+1}$;
  end
end

To ensure the stability of the datasets and create a robust model, the original data, including both the training and testing sets are normalized. The normalization process is conducted as follows:

$$\hat{x}_t = \frac{x_t - \mu}{\sigma}, \tag{7}$$

where $\hat{x}_t$ represents the normalized price at time t, while μ and σ represent the sample mean and sample standard deviation of the training set, respectively.

Additionally, the proposed MADDQN strategies are compared to other strategies such as buy and hold (B&H), sell and hold (S&H), moving average trend following (TF), moving average mean reversion (MR), TDQN, DQN-vanilla, MLP-Vanilla, etc. to evaluate their performance. Lastly, four metrics are presented to assess the stock trading performance directly. These metrics include the cumulative return, annualized return, maximum drawdown ratio, and Sharpe ratio.

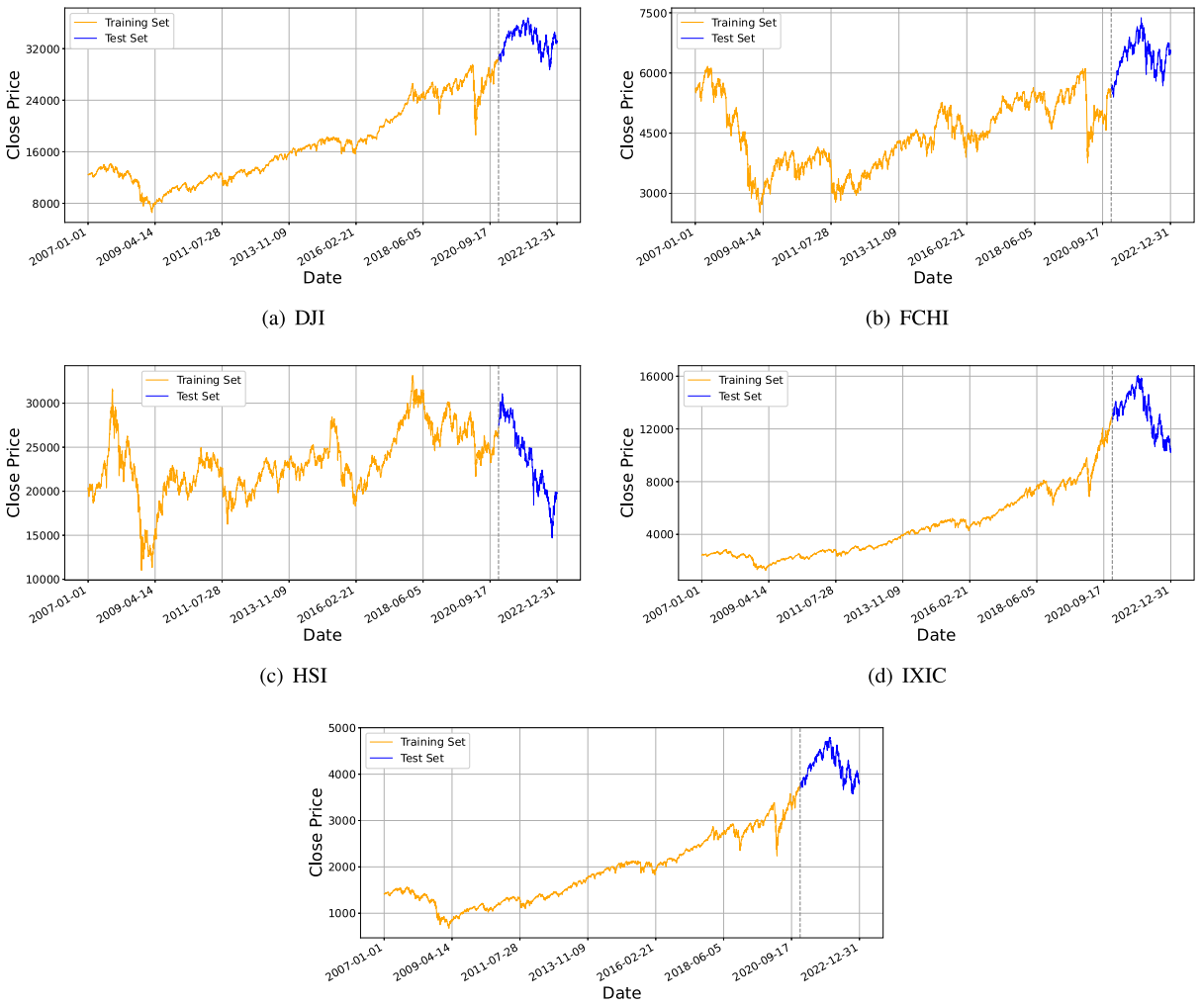
4.2. Comparison on single financial asset dataset

4.2.1. Experimental setup

Individual training of the models are conducted on five different stock index datasets corresponding to different countries.[2] Following the training phase, each model is assessed using a distinct test set associated with its respective stock index dataset. This evaluation process allows us to measure the model's accuracy in predicting diverse behaviors within various stock markets by training it on multiple datasets. The tuned hyperparameters for each independent agent in this experiment are presented in Table 1.

Due to the fact that the results of deep reinforcement learning are often highly stochastic, each agent are trained for 100 episodes to

[2] http://finance.yahoo.com



(a) DJI (b) FCHI

(c) HSI (d) IXIC

(e) SP500

Fig. 5. Close price curve of five stock indices.

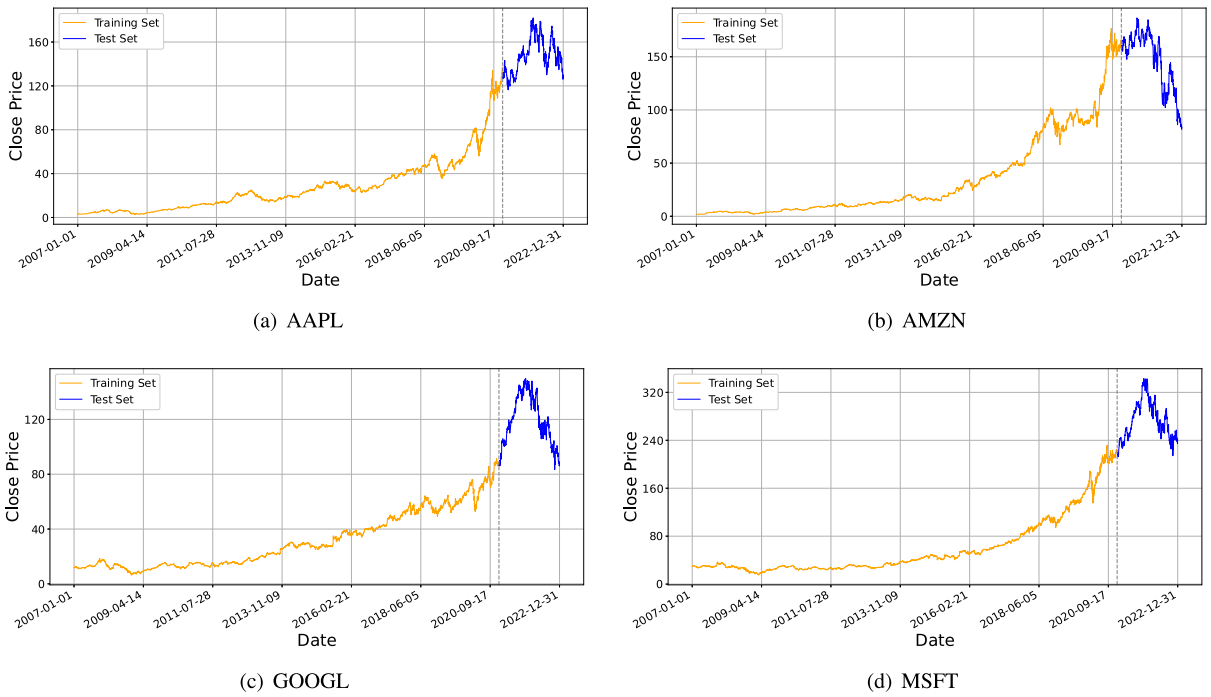
Table 1
The tuned hyperparameters.

Hyperparameter	Risk agent	Return agent	Final agent
Number of CNN layers	13	13	12
Number of Linear layers	2	2	6
Activation function	GELU	GELU	Tanh, ReLU, Sigmoid
Learning rate (α)	0.001	0.001	0.001
Replay memory size	1000	1000	1000
Discount factor	0.9	0.9	0.9
Decay rate of ϵ	0.9 to 0.05	0.9 to 0.05	0.9 to 0.05
Target Update Frequency	10	10	10
Window size	10	10	10

ensure that the training can converge. Also, the accumulated reward is used to measure the agent's training situation. The accumulated reward is the sum of the rewards that the agent gets from a complete trajectory (from begin to end). When the accumulated reward maintains a stable interval in multiple consecutive episodes, and there is no sudden drop or rise in the subsequent episodes, the training has converged. Fig. 7 shows the accumulated reward of three agents(risk agent, return agent and final agent) in 5 datasets during training. It can be clearly seen from the figure that there are no sudden drop or rise of accumulated reward for each agent after 10 or 15 episodes.

4.2.2. Comparison with baselines

To evaluate the performance of the proposed method, the testing results of agents are compared with baselines mentioned earlier. Figs. 8 to 12 shows the cumulative return (%) of baselines and the proposed method in five datasets (DJI, FCHI, HSI, IXIC, S&P500). Also, Table 2 lists metrics, including cumulative return (CR), annualized return (AR), maximum drawdown ratio (MDD), and Sharpe ratio (SR), that can evaluate the performance of all methods mentioned above. It is worth to notice that final agent, which is corresponding to the profit reward function, outperform each single agent(risk agent and return agent) in terms of cumulative return. Additionally, when compared with the baseline, final agent can get the highest cumulative return in almost all datasets. Apart from cumulative return, maximum drawdown ratio (MDD) is also worth to discuss. Risk agent uses Sharpe ratio as reward function, in this case, it considers both risk and return. Similarly, return agent consider more about return due to its reward function. Therefore, risk agent should have a smaller MDD than return agent theoretically. This assumption has been well demonstrated by Table 2. It can be seen from the table that, among all datasets, risk agent always have a smaller MDD compared to return agent. Also, due to the fact that final agent takes the Q value of risk agent and return agent as input, its MDD is always smaller than both agents or sit in the middle. Thus, the proposed method effectively learns information related to the optimal action during the training period and obtains superior performance from it.



(a) AAPL (b) AMZN

(c) GOOGL (d) MSFT

Fig. 6. Close price curve of four U.S. stocks.

Table 2
The performance of various trading approaches on five datasets.

Datasets	Metrics	Buy and hold	Sell and hold	MR	TF	TDQN	MLP-vanilla	DQN-vanilla	Risk agent	Return agent	Final agent
DJI	CR	7.90%	−8.49%	−12.50%	−23.49%	−26.14%	35.51%	34.51%	35.71%	17.09%	**41.53%**
	AR	7.73%	−4.14%	−8.69%	−19.42%	−15.12%	23.06%	22.73%	24.03%	13.25%	**27.01%**
	SR	0.32	−0.15	−0.39	−0.86	−0.88	1.24	1.14	1.16	0.65	**1.31**
	MDD	21.74%	21.31%	21.77%	32.99%	32.21%	**7.64%**	11.73%	9.58%	19.27%	15.44%
FCHI	CR	16.75%	−17.34%	−29.65%	1.71%	6.25%	37.97%	17.19%	31.16%	37.98%	**47.68%**
	AR	14.41%	−10.20%	−25.92%	3.54%	4.71%	25.29%	14.35%	22.58%	26.43%	**31.02%**
	SR	0.53	−0.27	−0.92	0.19	0.26	1.15	0.53	0.92	1.04	**1.34**
	MDD	22.26%	32.64%	33.64%	25.60%	28.11%	**8.84%**	23.44%	22.90%	20.07%	11.81%
HSI	CR	−32.52%	31.93%	−17.67%	−29.26%	−31.17%	56.43%	67.76%	36.95%	55.93%	**76.67%**
	AR	−26.66%	23.25%	−10.26%	−23.69%	−16.76%	35.31%	39.95%	27.5%	37.44%	**45.93%**
	SR	−0.65	0.92	−0.26	−0.65	−0.59	1.24	1.47	0.83	1.07	**1.39**
	MDD	51.51%	**12.07%**	28.99%	37.31%	51.98%	17.62%	12.45%	16.09%	22.94%	23.15%
IXIC	CR	−17.69%	17.1%	−1.53%	−36.3%	−28.62%	32.08%	19.25%	33.93%	61.71%	**65.64%**
	AR	−10.10%	16.53%	3.07%	−32.06%	−14.77%	23.43%	16.48%	25.49%	38.14%	**39.97%**
	SR	−0.26	0.44	0.087	−0.87	−0.53	0.76	0.51	0.76	1.25	**1.28**
	MDD	35.46%	25.53%	27.95%	43.83%	39.90%	25.94%	24.36%	14.11%	**11.65%**	12.91%
SP500	CR	2.61%	−3.21%	17.14%	−35.67%	−5.83%	0.32%	46.81%	52.42%	26.52%	**69.36%**
	AR	4.75%	1.69%	14.21%	−33.17%	−1.17%	2.99%	29.18%	33.14%	19.45%	**40.18%**
	SR	0.17	0.05	0.54	−1.18	−0.06	0.11	1.31	1.26	1.25	**1.65**
	MDD	25.38%	28.25%	17.00%	39.12%	27.75%	25.43%	13.05%	**10.02%**	15.38%	12.91%
Average	CR	−4.59%	4.00%	−8.84%	−24.60%	−17.10%	32.46%	37.10%	38.03%	39.85%	**60.18%**
	AR	−1.97%	5.43%	−5.52%	−20.96%	−8.62%	22.02%	24.54%	26.55%	26.94%	**36.82%**
	SR	0.02	0.20	−0.19	−0.68	−0.36	0.90	1.00	0.99	1.05	**1.39**
	MDD	31.27%	23.96%	25.87%	35.77%	34.71%	17.09%	17.01%	**14.54%**	17.86%	15.24%

[1] **CR**: Cumulative return refers to the total amount of return on an investment over a period of time;
[2] **AR**: Annualized return refers to the average rate of return earned on an investment over a period of time, expressed as a percentage per year;
[3] **SR**: The Sharpe ratio used to evaluate the risk-adjusted return of an investment or investment portfolio. It measures the excess return earned over a risk-free rate per unit of volatility or total risk;
[4] **MDD**: Maximum drawdown indicates the potential downside risk of an investment or portfolio, measuring the largest percentage drop from a peak to a trough in the value of an investment or investment portfolio over a specific period of time.

4.2.3. Analysis of single agent and multi-agent

The performance of the risk agent and the final agent are compared on five different datasets and their cumulative return trends are presented in Fig. 13. It can be seen that the final agent consistently outperformed the risk agent in all test results. When observing the risk agent, one can noticed that its cumulative return had a relatively smooth growth pattern. Although there were significant increases and decreases in the FCHI dataset, it exhibited a stable upward trend in the remaining datasets. This aligned with the theoretical behavior of the risk agent's reward function, which prioritized risk avoidance.

The final agent, which took into account the trading decisions of the risk agent, was able to better mitigate risks. From Fig. 13, it was evident that the final agent could simultaneously avoid risks and achieve higher returns. This characteristic was also reflected in Table 2, where the

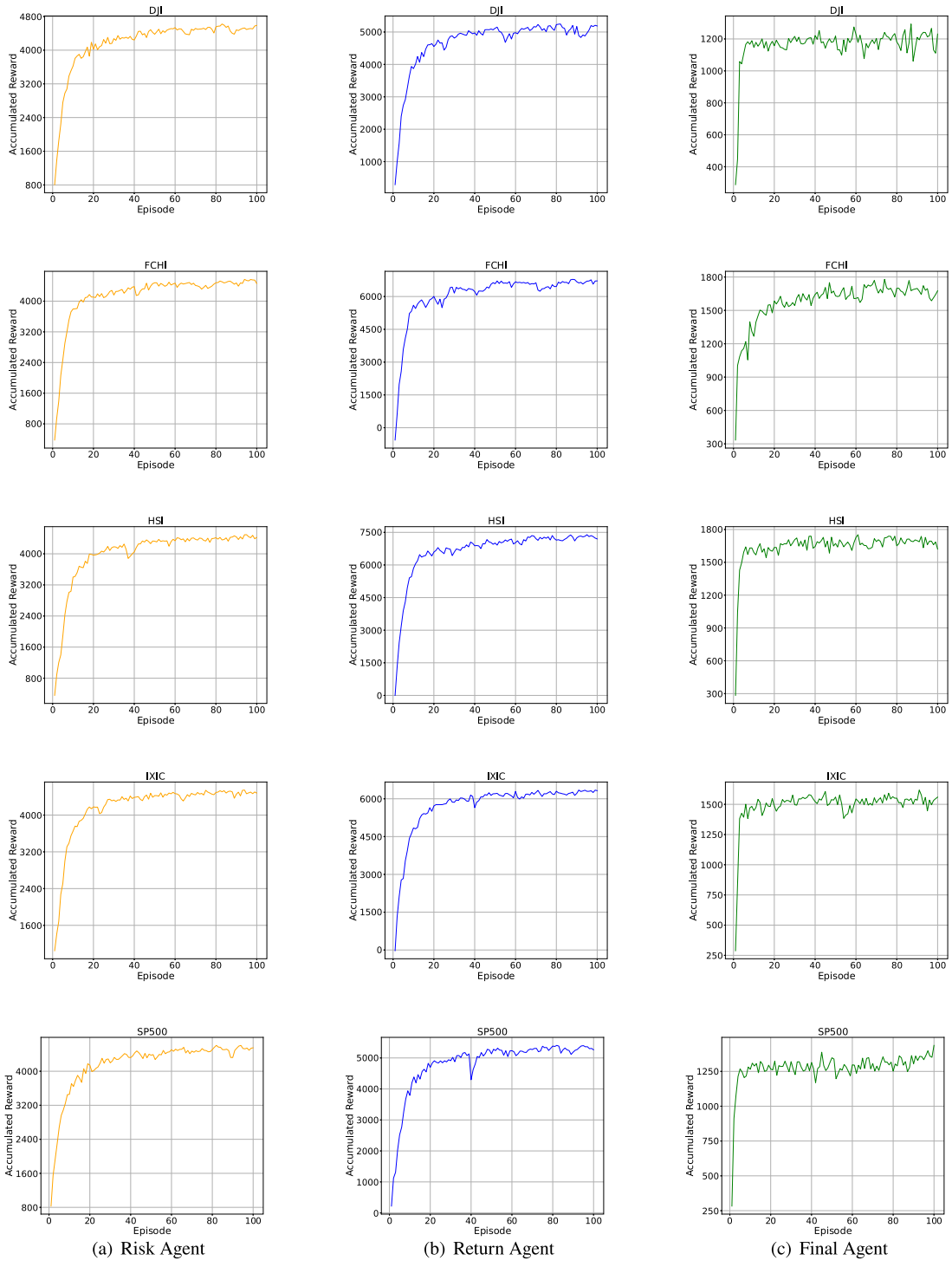


Fig. 7. Accumulated reward of three agents in five datasets.

final agent consistently achieved lower or similar Maximum Drawdown Ratio (MDD) compared to the risk agent across all datasets.

The performance of the return agent and the final agent are also compared on five different datasets and their cumulative return trends are presented in Fig. 13. Similar to the previous comparison with the risk agent, the final agent consistently outperformed the return agent in all test results.

When observing the return agent, one can noticed that the cumulative return trends exhibited significant declines or increases in all datasets. Taking the HSI dataset as an example, the curve experienced rapid growth around the 300th day, reaching a cumulative return of around 50%. However, after the 400th day, the curve started a sharp decline, and the return rate dropped to 20% before stabilizing. This situation occurred because the return agent's reward function focused on long-term returns without considering risks.

The final agent overcame the shortcomings of the return agent and could prioritize both returns and risk mitigation. From Fig. 13, it was evident that the final agent could increase returns at a faster pace while avoiding significant declines. The Maximum Drawdown Ratio (MDD) section in Table 2 also supported this observation, as the return agent consistently exhibited higher or similar MDD values compared to the final agent across all datasets.

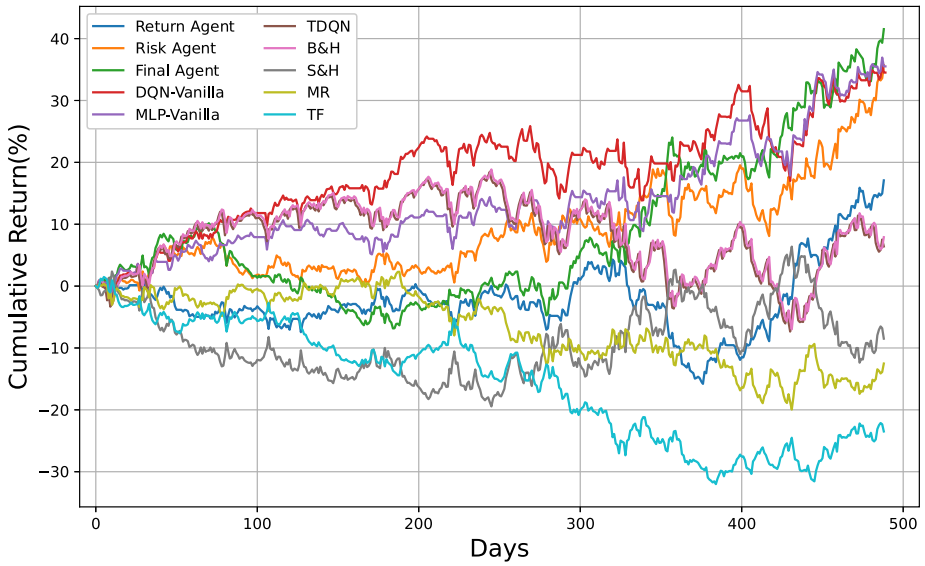


Fig. 8. Cumulative return(%) of proposed method and baselines in DJI.

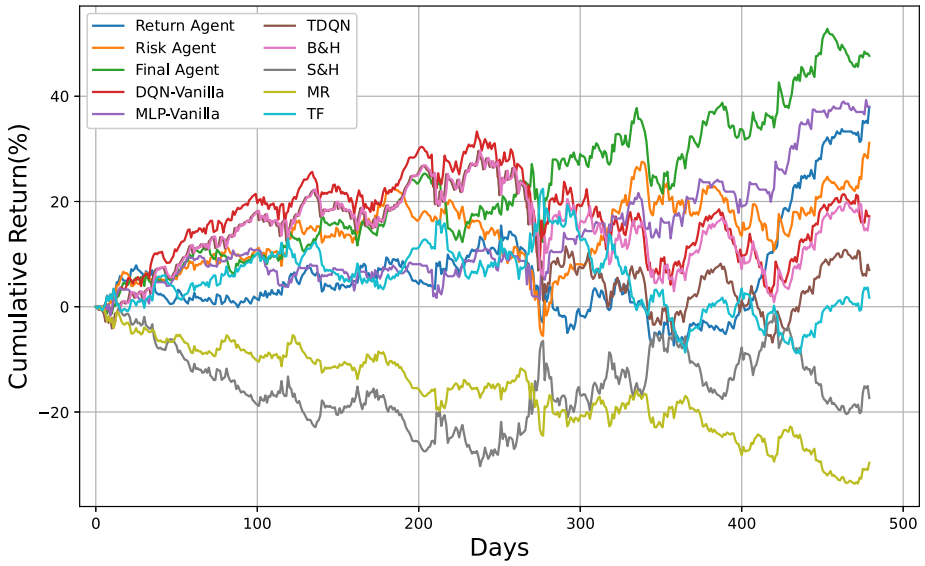


Fig. 9. Cumulative return(%) of proposed method and baselines in FCHI.

4.2.4. Analysis of strategy

After analyzing the differences and advantages among the risk agent, return agent, and final agent based on the cumulative return trend, further analysis was still required. Actions taken by each agent across all datasets need to be observed. Also, whether these actions aligned with the expected theoretical performance of their reward functions should be carefully analyzed. Figs. 14 to 18 display the actions performed by the three agents in five different datasets. The actions were marked on the stock price trend charts to facilitate understanding of how the agents made trading decisions based on price movements.

Firstly, from the zoomed-in sections of the five figures, one can observe that the risk agent tended to take more actions within the same number of trading days. This aligned with the characteristics of the risk agent's reward function, the Sharpe Ratio, which encouraged the agent to mitigate risks. Consequently, the agent executed more trading decisions to avoid potential losses. On the other hand, the return agent, as it focused more on long-term gains, tended to make fewer trading decisions. The final agent, as a combination of both agents' strategies, amalgamates the return agent's emphasis on long-term returns and the risk agent's focus on risk mitigation. This amalgamation leads to the final agent exhibiting a higher number of actions compared to the return agent, but a lower number than the risk agent. This approach ensures a balance between capitalizing on potential gains and safeguarding against undue losses. Therefore, the final agent is designed to execute a greater number of trading decisions than the return agent, yet a lesser number than the risk agent.

Secondly, the final agent incorporated the "advice" provided by the two sub-agents and selected the superior option for making trading decisions. Taking the zoomed-in section of Fig. 14 as an example, around the 432nd day, the stock price reached a low point. The return agent chose to execute a "long" decision on this day, while the risk agent chose not to trade. The final agent, after considering both "advice" sources, deemed the return agent's choice as superior and executed a "long" decision on the same day. Several days later, the stock price reached a high point, and both the risk agent and return agent chose to execute "short" decisions during this period. From a macro perspective, both decisions could be profitable, but it became evident that the risk agent's decision was wiser. Therefore, the final agent followed the "advice" of the risk agent in this trade. Similar situations occurred in

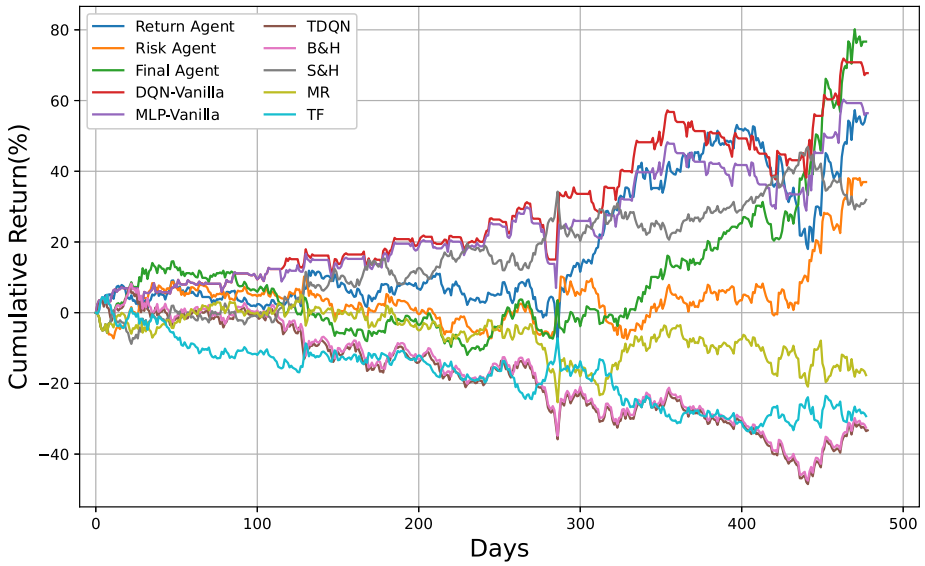


Fig. 10. Cumulative return(%) of proposed method and baselines in HSI.

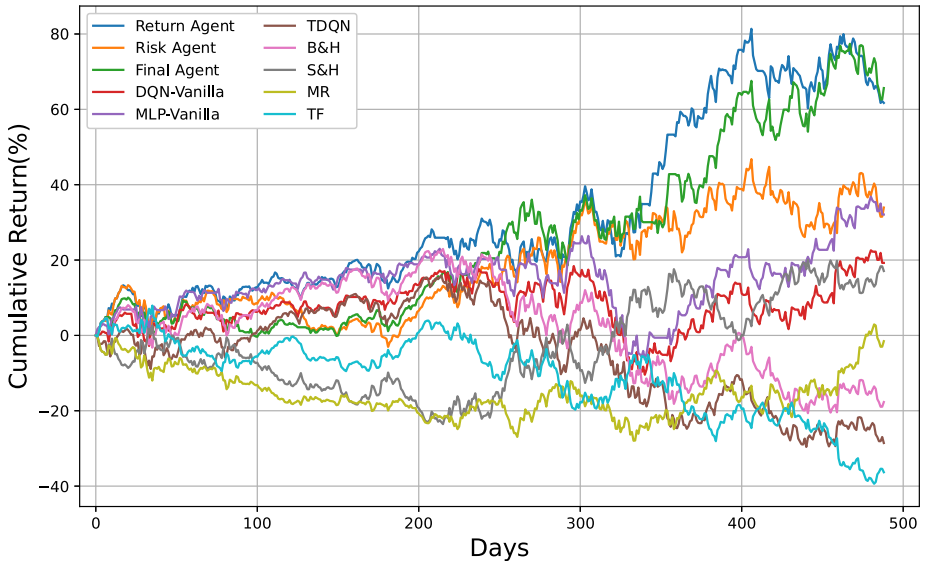


Fig. 11. Cumulative return(%) of proposed method and baselines in IXIC.

Fig. 17, where the final agent followed the risk agent's "advice" around the 250th day and achieved higher returns.

Thirdly, the final agent did not always strictly adhere to the "advice" of the two agents. Sometimes it made trading decisions based on its own judgment. In Fig. 15, around the 136th day, the stock price reached a high point. Both the risk agent and return agent chose to stay put, as they had already executed "short" decisions in the preceding days. From a macro perspective, neither the risk agent's nor the return agent's choices could be considered astute, as initiating a "short" position too early would inevitably result in losses. The final agent, with acute awareness of the price trend, diverged from the "advice" of the two sub-agents and executed a "short" decision based on its own judgment. Similar situations were also evident in Figs. 16 and 18. In Fig. 18, around the 397th day, the final agent executed a "short" decision, while in Fig. 16, it did the same around the 161st day. These spontaneous trading decisions differed from those of the risk agent and return agent at the same time. Although the final agent's decisions may not have always been optimal from a macro perspective, they at least generated higher returns compared to the two sub-agents.

4.2.5. Conclusion of single financial asset dataset

In this experiment, the proposed method is analyzed from three dimensions and the risk agent, return agent, and final agent are compared with several baselines. Firstly, from the perspective of cumulative return, the final agent achieved higher returns compared to the risk agent and return agent. Additionally, the cumulative return curve of the final agent was more stable, without significant declines. Secondly, in terms of trading decisions, the final agent effectively incorporated the "advice" from the risk agent and return agent and selected the superior option. When both agents' decisions were suboptimal, the final agent leveraged its own judgment to make better trading decisions based on the current situation. Thirdly, the action execution of the three agents well reflected the characteristics of their respective reward functions, and the number of actions taken within the same number of trading days closely aligned with the theoretical performance of the reward functions. Fourthly, compared to the baselines, the final agent achieved the best performance across all datasets.

In conclusion, the proposed method effectively learned the price trends in the stock market. Leveraging the framework of multi-agent reinforcement learning, the final agent efficiently absorbed the "advice"

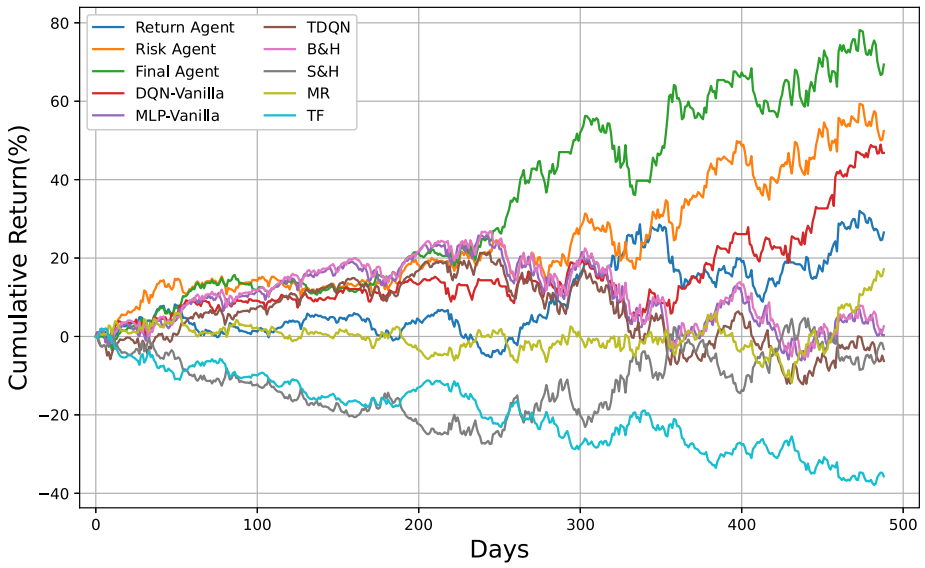


Fig. 12. Cumulative return(%) of proposed method and baselines in SP500.

from sub-agents, leading to smarter trading decisions. Furthermore, the ability of the final agent to learn from "advice" implies that the number of sub-agents, reward functions, and other parameters are not fixed. They can be modified according to specific circumstances to enhance the performance of the final agent.

4.3. Comparison of the mixed dataset of multiple financial assets

A dataset consisting of multiple assets from three stock indices(DJI, IXIC, SP500) of United States is created. Each sample was created by slicing the selected stock indices according to a fixed look-back window length, resulting in a data pool. The models were then trained using a random selection of samples from this data pool, with the aim of creating a generalized model that could be applied to other underlying assets. This approach is critical in developing a model that can handle different market conditions and assets, which is necessary for real-world applications in financial markets. By training on a diverse set of stock index data, the model becomes more adaptable and better equipped to handle varying financial data and provide accurate predictions for future trading strategies. This strategy allows for the creation of a robust and reliable model that can handle different market conditions and assets, providing a practical and effective tool for financial decision-makers.

4.3.1. Experimental setup

In line with previous experiments, the multi-agent framework are trained and tested on a dataset. However, the training set was no longer based on a single stock index but on a mixed dataset consisting of three stock indexes as mentioned in the previous section. Similarly, the testing set underwent changes. Instead of testing the results on DJI, IXIC, or SP500, agents are on four US stocks (AAPL, AMZN, GOOGL, MSFT) in order to validate the generalizability of the proposed framework. The close price curve of the four stocks is shown in Fig. 6. The experimental hyperparameters remained largely consistent with those used previously, with the only difference being an increase in the size of the replay buffer from 1000 to 10 000.

The training of the model followed the same procedure as before. Each agent are trained for 100 iterations and selected the best-performing model from the converged models for subsequent testing. The training progress is illustrated in Fig. 19.

4.3.2. Comparison with baselines

To evaluate the performance of the proposed method, the testing results of agents are compared with baselines mentioned earlier. Figs. 20 to 23 show the cumulative return(%) of baselines and the proposed method in four datasets. Also, Table 3 lists metrics, including cumulative return (CR), annualized return (AR), maximum drawdown ratio(MDD), and Sharpe ratio (SR), that can evaluate the performance of all methods mentioned above.

By combining Figs. 20 to 23 and Table 3, it can be observed that in the AAPL and MSFT datasets, the cumulative return of the final agent significantly surpassed all the baselines. In the GOOGL dataset, final agent achieved a marginal advantage over DQN-Vanilla. Although the final agent did not outperform all the baselines in the AMZN dataset, it only slightly lagged behind sell and hold and DQN-Vanilla. Apart from the cumulative return, another important aspect to discuss is the Maximum Drawdown (MDD). In previous experiments, the characteristics of the final agent, which could maximize returns while mitigating risks has been detailedly demonstrated. Observing Table 3, it can be seen that the MDD of the final agent was the lowest in half of the cases. Moreover, in the average section, it can be observed that the final agent's average MDD was only slightly lower than MLP-Vanilla, while its average return was higher than all the baselines. This indicated that the proposed method exhibited good generalizability and satisfied the theoretical properties established in the previous experiments.

4.3.3. Analysis of strategy

The trading decisions made by the final agent during testing on each dataset still needed to be analyzed. Figs. 24 to 27 displayed the actions taken by the final agent in the four datasets. The actions were marked on the stock price trend charts to facilitate understanding of how the agent made trading decisions based on price movements.

First, let us examine the zoomed-in sections of the four figures, where the final agent demonstrated numerous astute trading decisions. Taking Fig. 25 as an example, around the 347th day, the stock price reached a peak. At this point, the final agent anticipated the upcoming downward trend and promptly chose to execute a "short" trading decision. By the 355th day, the stock price plummeted to nearly a low point, and the final agent chose to execute a "long" trading decision. From a macro perspective, the final agent's choices proved to be correct as the subsequent stock prices experienced a prolonged upward trend.

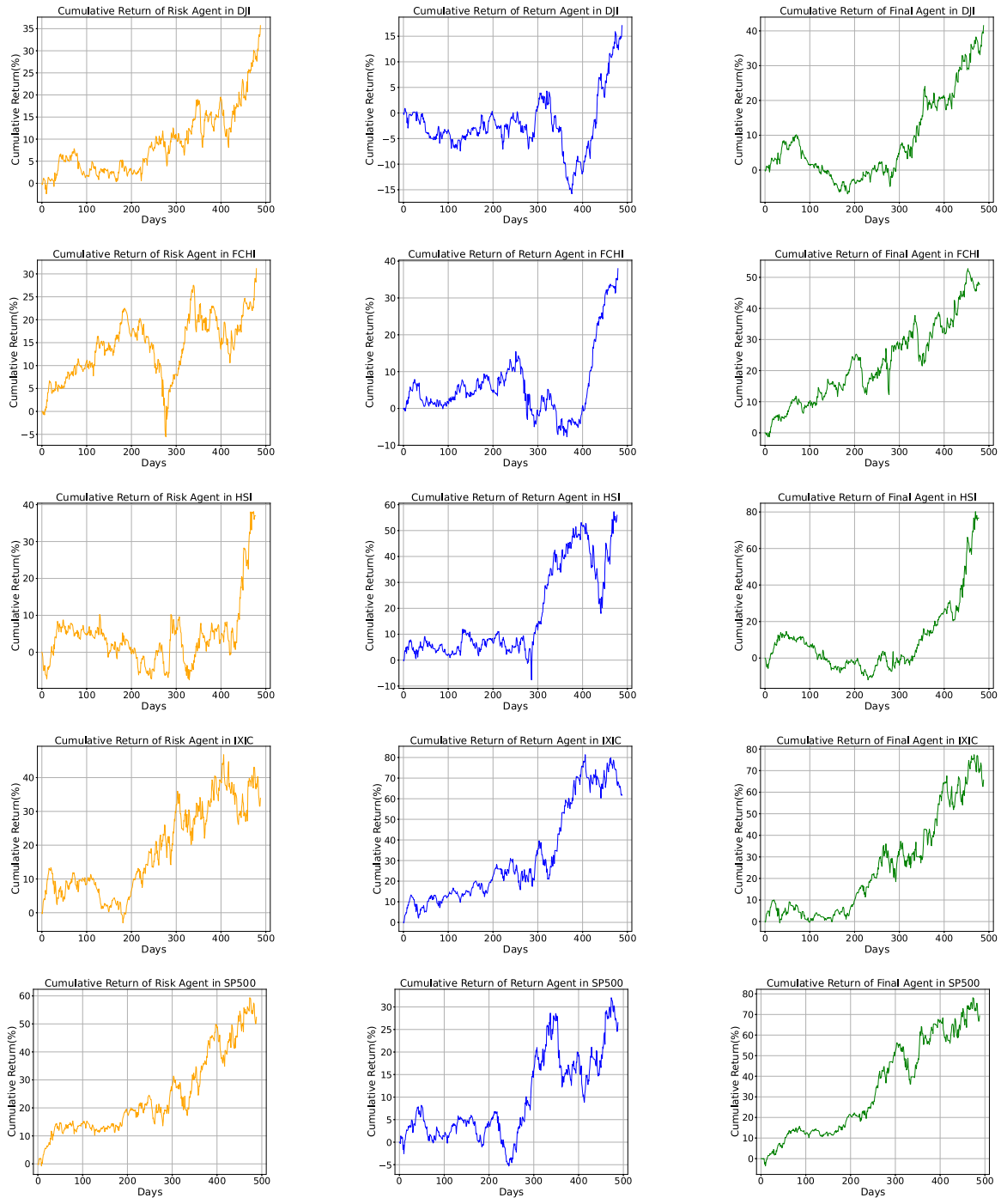


Fig. 13. Cumulative return(%) of Risk Agent, Return Agent and Final Agent in five datasets.

Similar examples can be found in Fig. 26, where the final agent selected a "long" trading decision near the low point around the 258th day and a "short" trading decision near the high point around the 265th day. These decisions consistently yielded profits.

Moreover, the final agent exhibited the ability to accurately observe forthcoming price trends. Fig. 27 served as an example, where from around the 180th day to the 200th day, the stock price was about to undergo a rapid rise. The final agent successfully anticipated this trend and chose a "long" trading decision around the 182nd day. It refrained from executing any other decisions for the next 20 days, which further demonstrated its precise anticipation of price trends.

4.3.4. Conclusion of multiple financial asset dataset

In this experiment, the proposed method has been analyzed from two perspectives and the final agent of the framework has been compared with several baselines. Firstly, in terms of cumulative return, the final agent outperformed the baselines in most of the datasets, and it also exhibited lower Maximum Drawdown (MDD), indicating its ability to mitigate risks while maximizing returns, which aligns with the theoretical properties established in previous experiments. Secondly, in terms of trading decisions, the final agent demonstrated the ability to anticipate future stock price trends and execute relatively favorable trading decisions. It is worth noting that all the testing results in this experiment were based on the same set of model parameters. The final agent achieved outstanding performance across multiple datasets

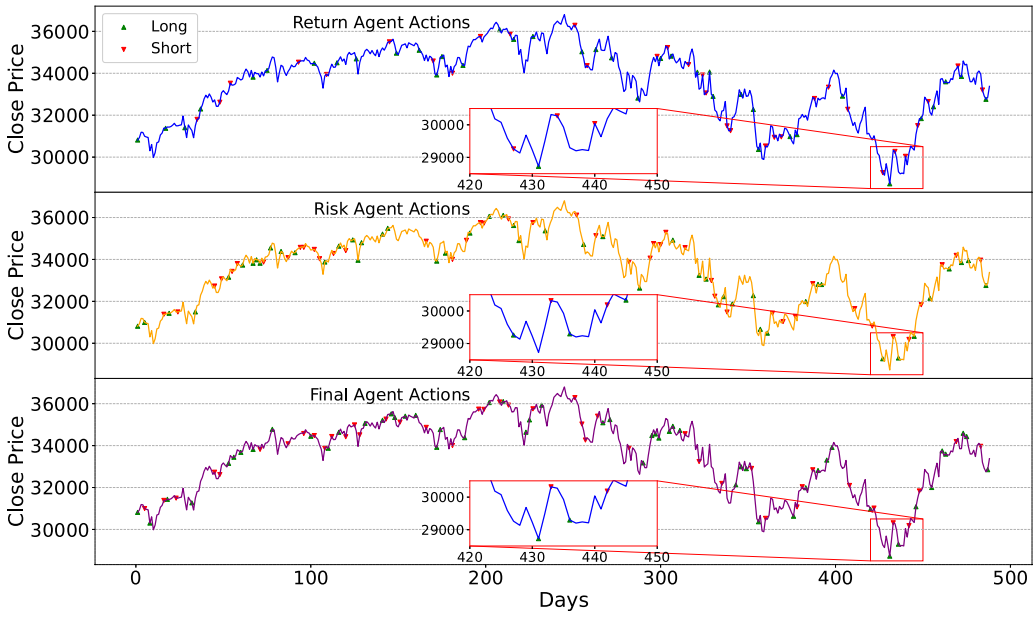


Fig. 14. Actions of Risk Agent, Return Agent and Final Agent in DJI.

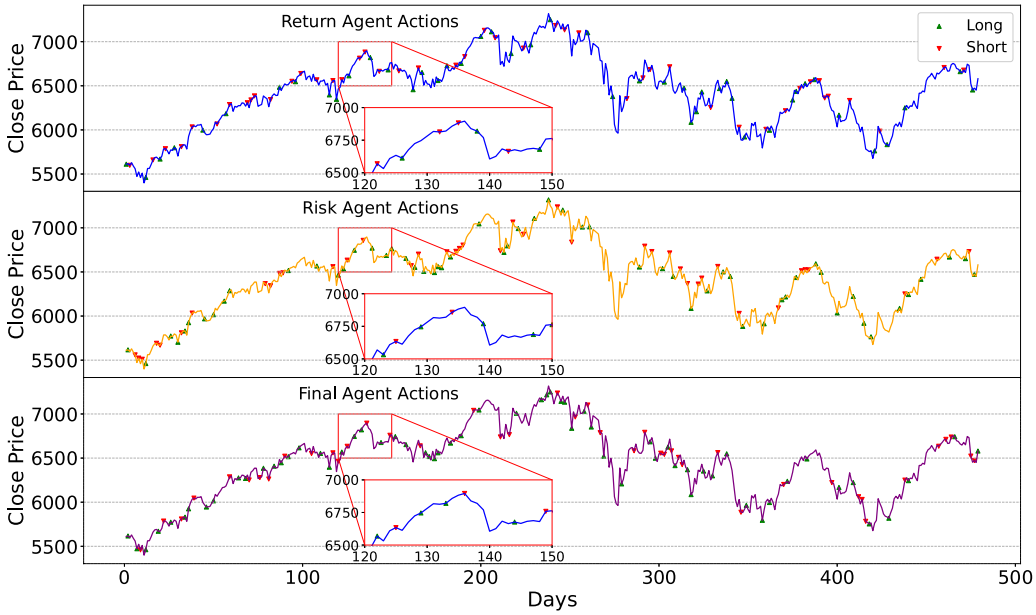


Fig. 15. Actions of Risk Agent, Return Agent and Final Agent in FCHI.

without specific training for each dataset, indicating that the proposed method exhibits remarkable generalizability.

5. Conclusion

In this paper, a novel financial trading framework that leverages the principles of multi-agent reinforcement learning has been presented, aiming to address the challenges associated with financial markets, such as high volatility, non-linearity, and complex dynamics. By employing multiple intelligent agents to learn and interact with the environment, the decision-making process and overall trading performance has been ehanced. The main contributions are as follows:

1. A novel financial trading framework based on MADDQN is proposed, which successfully balances the pursuit of maximum revenue and the avoidance of risk by innovatively employing two different agents with different trading preferences. The final agent leverages the trading "advice" provided by each sub-agent to make informed trading decisions that are suitable for the current market conditions. Importantly, the framework is not limited to a specific reward function and does not impose restrictions on the number of sub-agents. Researchers have the flexibility to choose appropriate reward functions and the number of sub-agents based on their specific requirements.

2. The proposed framework pioneeringly combine the state-of-the-art time series model, TimesNet. By utilizing this model, the proposed framework achieves accurate predictions of stock trends, which represents a significant breakthrough in the field of MADDQN.

3. The proposed framework is also been trained in a mixed dataset, which is based on major US stock market indices (DJI, IXIC, SP500). This dataset captures the features of the US stock market during a specific time period. Through training on this dataset, agents effectively learn and grasp the changing trends of multiple stock prices in the US

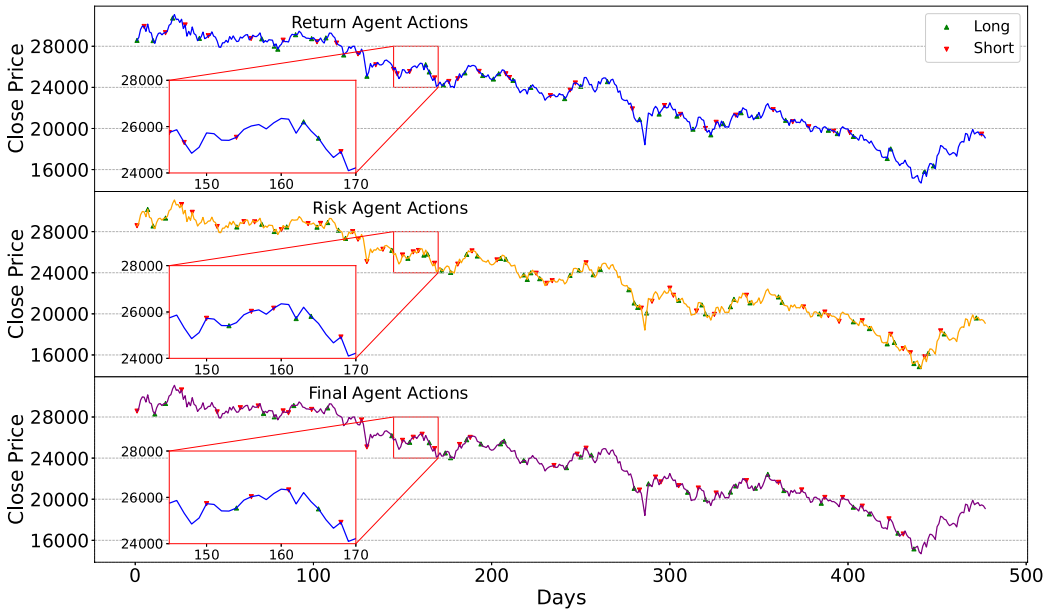


Fig. 16. Actions of Risk Agent, Return Agent and Final Agent in HSI.

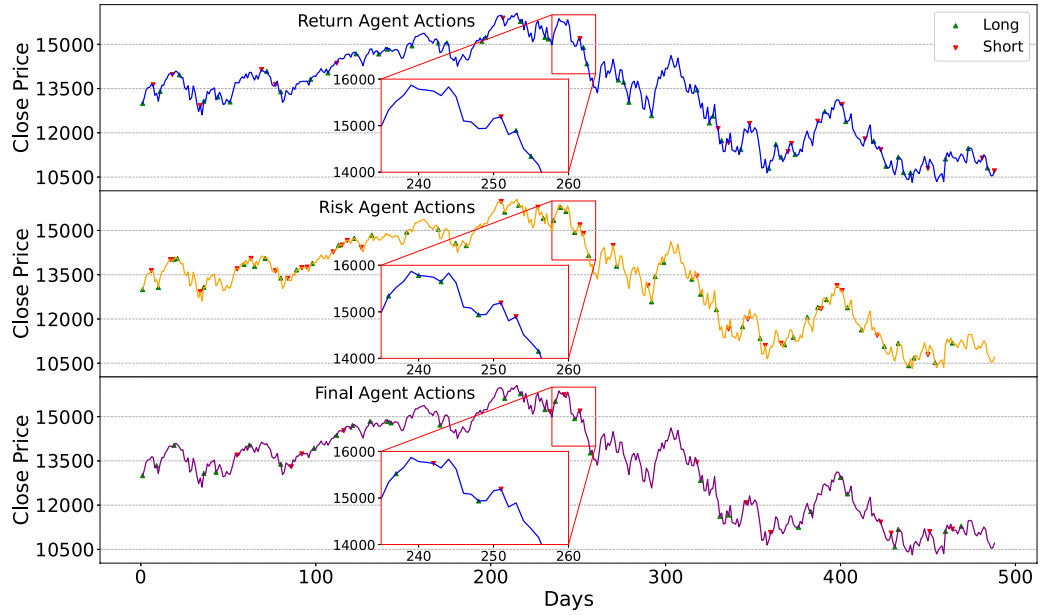


Fig. 17. Actions of Risk Agent, Return Agent and Final Agent in IXIC.

stock market, which indicates a strong generalization of the proposed framework.

Overall, the research contributions encompass the development of a flexible MADDQN framework for financial trading. Agents based on advanced time series modeling techniques are implemented and a comprehensive dataset is created. Agents of the proposed framework can make informed decisions and capture the dynamics of the stock market more accurately.

Through the experiments, following observations have been made:

1. Agents of proposed framework demonstrate remarkable convergence during the training process, indicating that the proposed framework effectively overcomes the noise and uncertainties that commonly appeared in financial data. This convergence indicates the robustness of the framework in handling complex financial dynamics.

2. The performance of proposed framework, as compared to the baseline mentioned earlier, shows significant improvements in both training and testing phases across five different stock indices. Moreover, all agents within the framework exhibit performance that aligns with their respective reward functions. This indicates that the proposed approach not only enables agents to achieve outstanding performance in financial trading but also ensures that their trading decisions are grounded in the theoretical foundations of their reward functions.

3. The proposed framework demonstrates strong generalization capabilities. Through training on the mixed dataset, the final agent exhibits excellent performance across multiple stocks. This highlights the ability of the framework to generalize its learnings and make informed decisions across other underlying assets with a refinement.

Overall, the experimental findings provide substantial evidence of the effectiveness of the proposed framework. The agents exhibit strong convergence, outperform baseline methods, adhere to their respective reward functions, and indicate robust generalization capabilities across multiple stocks.

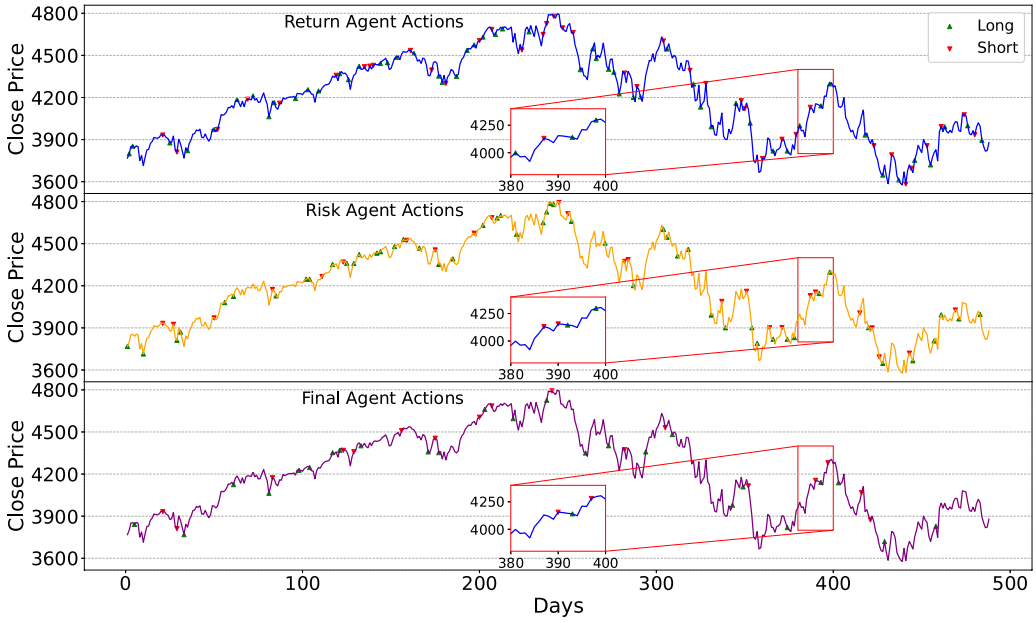
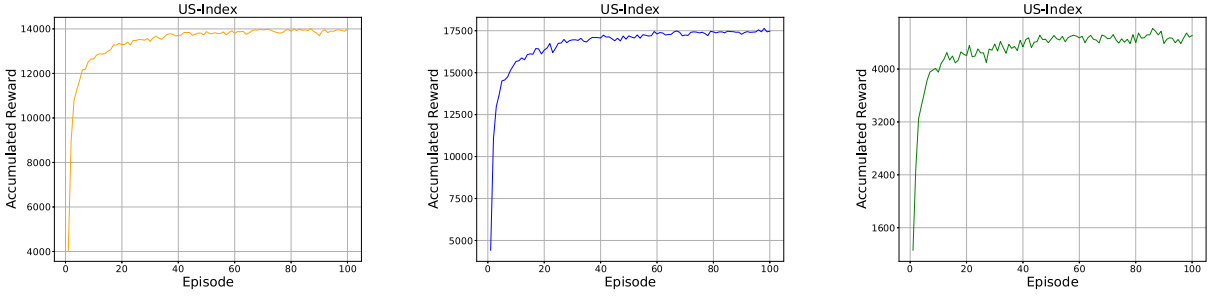


Fig. 18. Actions of Risk Agent, Return Agent and Final Agent in SP500.



(a) Risk Agent (b) Return Agent (c) Final Agent

Fig. 19. Accumulated reward of three agents in US-Index data.

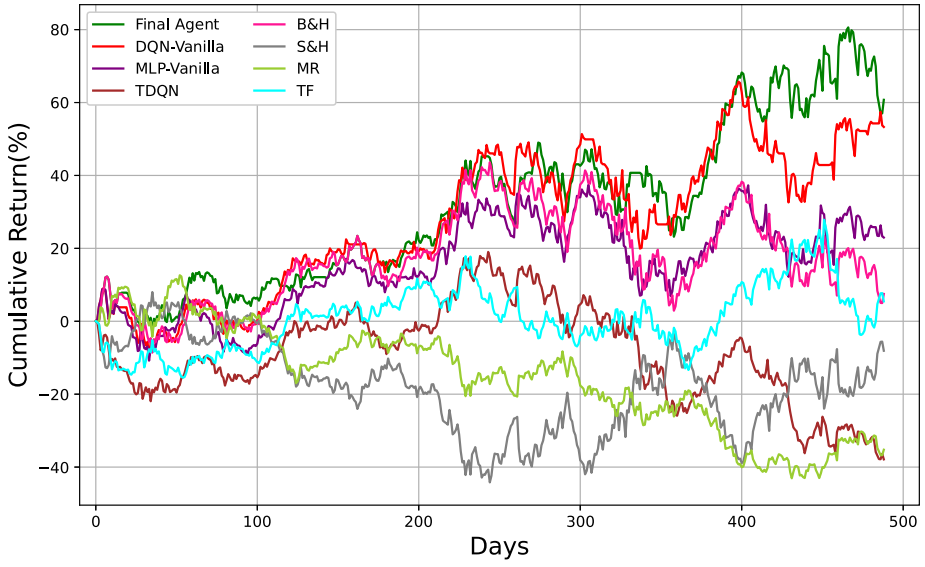


Fig. 20. Cumulative return(%) of proposed method and baselines in AAPL.

Table 3
The performance of various trading approaches on four single stock datasets.

Datasets	Metrics	buy and hold	sell and hold	MR	TF	TDQN	MLP-Vanilla	DQN-Vanilla	Final agent
AAPL	CR	7.49%	−8.10%	−35.24%	5.46%	−39.62%	22.97%	53.30%	**60.75%**
	AR	12.26%	9.53%	−27.92%	9.88%	−23.43%	20.74%	35.47%	**39.96%**
	SR	0.27	0.14	−0.60	0.24	−0.68	0.51	0.93	**1.60**
	MDD	28.34%	48.34%	49.45%	24.74%	50.25%	22.17%	20.00%	**17.34%**
AMZN	CR	−44.25%	43.65%	−16.07%	−55.66%	−62.32%	26.92%	**46.03%**	33.37%
	AR	−35.14%	33.89%	−3.66%	−59.49%	−58.11%	21.78%	**34.06%**	29.20%
	SR	−0.57	0.73	−0.07	−1.02	−1.06	0.60	**0.76**	0.60
	MDD	54.47%	**20.40%**	45.76%	64.44%	69.21%	24.01%	31.79%	37.81%
GOOGL	CR	3.39%	−3.99%	38.33%	−61.79%	−35.43%	55.56%	74.75%	**75.61%**
	AR	10.19%	46.82%	30.58%	−80.09%	−18.53%	35.86%	44.88%	**46.22%**
	SR	0.22	0.40	0.69	−1.56	−0.75	1.00	1.12	**1.14**
	MDD	44.32%	73.53%	22.13%	64.20%	57.28%	21.76%	25.58%	**21.52%**
MSFT	CR	16.64%	−17.24%	2.11%	−38.91%	−27.11%	12.14%	21.02%	**42.37%**
	AR	17.59%	10.07%	7.14%	−34.09%	−12.50%	14.44%	19.34%	**30.94%**
	SR	0.42	0.12	0.18	−0.80	−0.40	0.35	0.49	**0.81**
	MDD	37.15%	62.81%	33.98%	53.61%	55.50%	37.56%	31.19%	**30.17%**
Average	CR	−4.18%	3.58%	−2.72%	−37.73%	−41.12%	29.40%	48.78%	**53.03%**
	AR	1.23%	25.08%	1.54%	−40.95%	−28.14%	23.21%	33.44%	**36.58%**
	SR	0.08	0.35	0.05	−0.79	−0.72	0.62	0.83	**1.11**
	MDD	41.07%	51.27%	37.83%	51.75%	58.06%	26.38%	27.14%	**26.71%**

[1]**CR:** Cumulative return refers to the total amount of return on an investment over a period of time;
[2]**AR:** Annualized return refers to the average rate of return earned on an investment over a period of time, expressed as a percentage per year;
[3]**SR:** The Sharpe ratio used to evaluate the risk-adjusted return of an investment or investment portfolio. It measures the excess return earned over a risk-free rate per unit of volatility or total risk;
[4]**MDD:** Maximum drawdown indicates the potential downside risk of an investment or portfolio, measuring the largest percentage drop from a peak to a trough in the value of an investment or investment portfolio over a specific period of time.

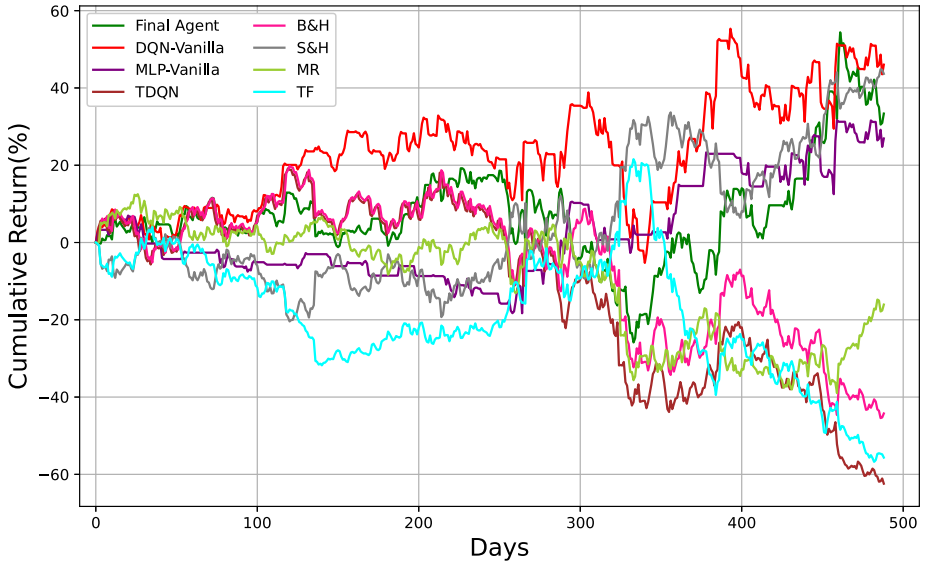


Fig. 21. Cumulative return(%) of proposed method and baselines in AMZN.

Indeed, there exist some limitations within the proposed framework. The experimental results involving the mixed dataset indicate that although the proposed framework exhibits strong generalization, it is not always guaranteed that the trading decisions made by the final agent are optimal. Various factors contribute to this particular outcome. Firstly, the hyperparameters, such as the replay buffer size, learning rate, and discount factor, may be further optimized to achieve better performance. Secondly, in the current work, two sub-agents with one focusing on risk and the other on mid-term returns, are implemented to offer "advice" to the final agent. In the future work, more agents with different trading preferences will be adopted to the current framework. By incorporating more diverse sub-agents that consider a wider range of

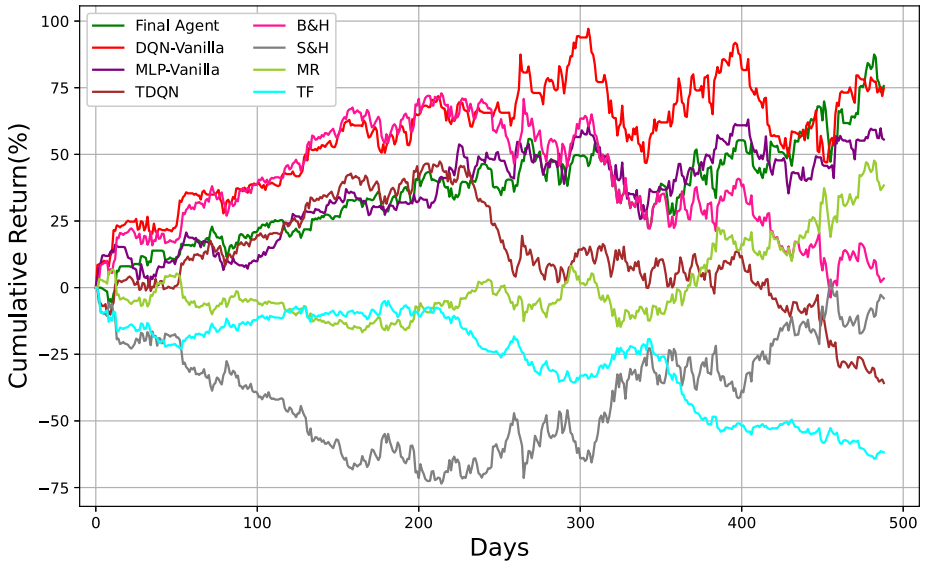


Fig. 22. Cumulative return(%) of proposed method and baselines in GOOGL.

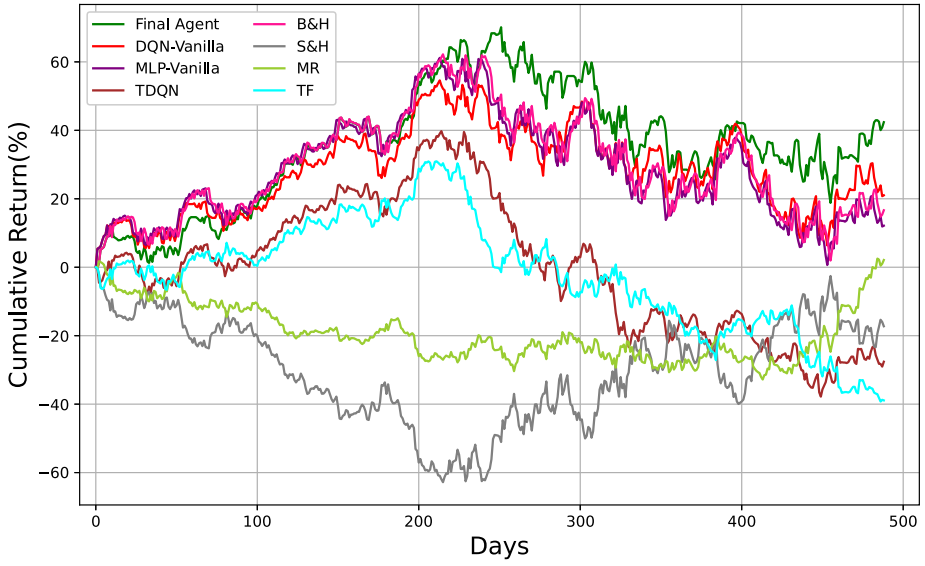


Fig. 23. Cumulative return(%) of proposed method and baselines in MSFT.

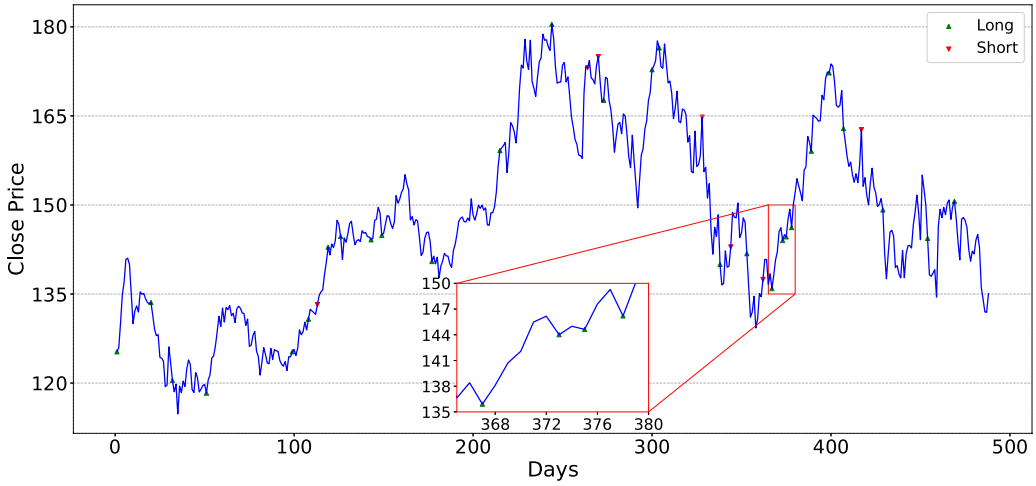


Fig. 24. Actions of Final Agent in AAPL.

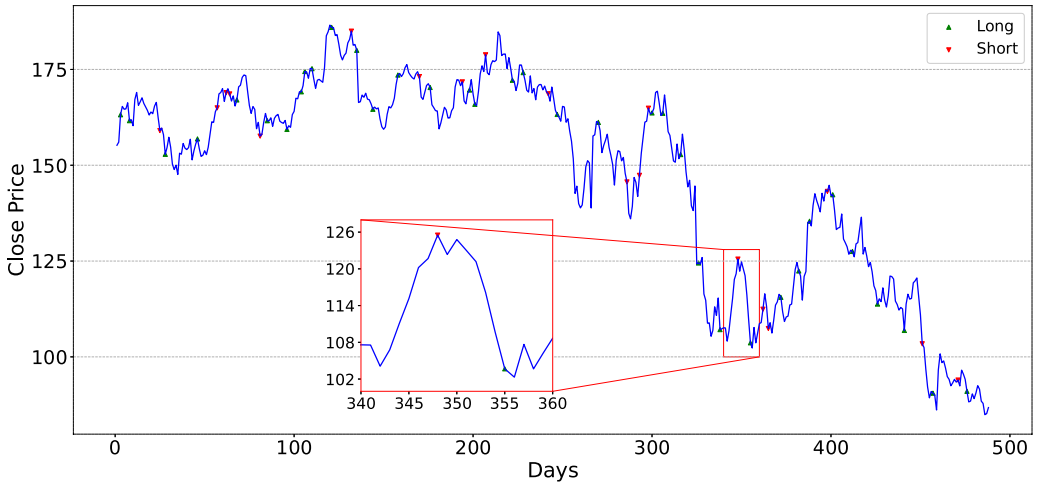


Fig. 25. Actions of Final Agent in AMZN.

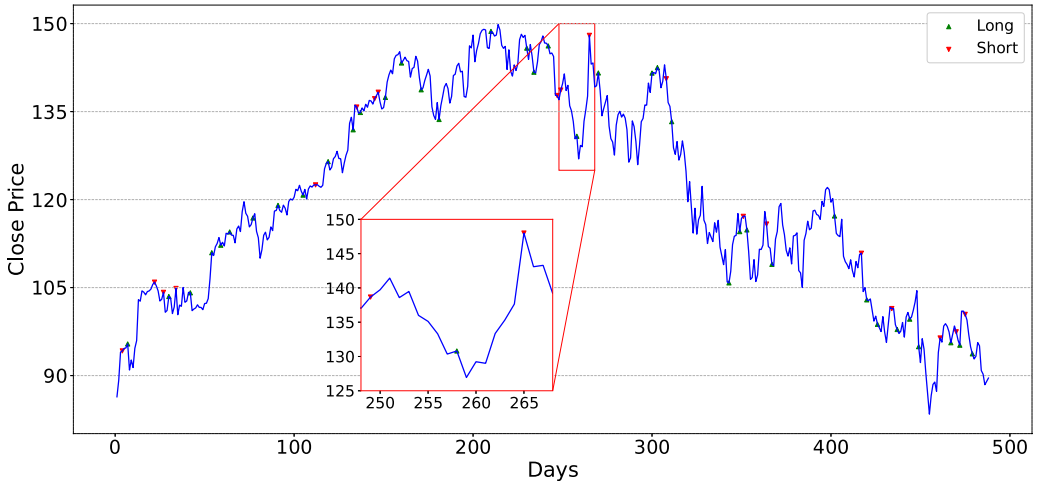


Fig. 26. Actions of Final Agent in GOOGL.

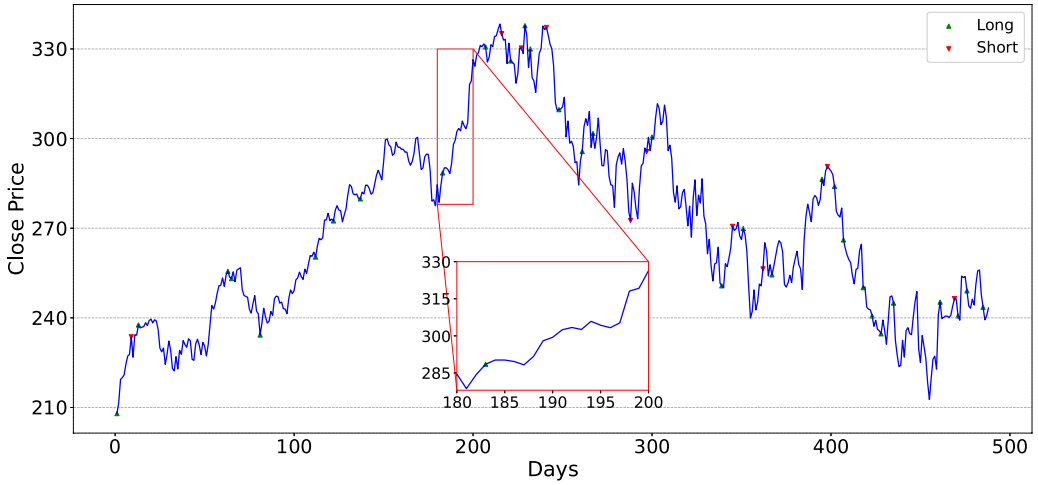


Fig. 27. Actions of Final Agent in MSFT.

trading scenarios, the final agent's decisions could potentially become more optimal. These assumptions will be addressed in future work to gradually refine the framework and address these limitations.

CRediT authorship contribution statement

Yuling Huang: Methodology, Conceptualization, Investigation, Writing – original draft. **Chujin Zhou:** Methodology, Software, Validation, Visualization, Writing – original draft. **Kai Cui:** Methodology, Software, Validation. **Xiaoping Lu:** Supervision, Writing – review & editing, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported in part by the Faculty Research Grants Macau University of Science and Technology (Project no. FRG-22-001-INT), and the Science and Technology Development Fund, Macau SAR (File no. 0096/2022/A).

Appendix

Abbreviations	Meanings
DQN	Deep Q-Network
DDQN	Double Deep Q-Network
TDQN	Trading Deep Q-Network
MADDQN	Multi-Agent Double Deep Q-Network
RNN	Recurrent Neural Network
CNN	Convolutional Neural Network
MLP	Multilayer Perceptron
LSTM	Long Short Term Memory
RRL	Recurrent Reinforcement Learning
SACLSTM	Stock Sequence Array Convolutional LSTM
VMD	Variational Mode Decomposition
CEEMDAN	Complete Ensemble Empirical Mode Decomposition with Adaptive Noise
MLCA-LSTM	Multi-scale Local Cues and hierarchical Attention-based LSTM
AComNN	Attention enhanced Compound Neural Network
Dwtformer	Wavelet Decomposition Transformer
DRQN	Deep Recurrent Q-Network
DQN-HER	DQN-Hindsight Experience Replay
TRPO	Trust Region Policy Optimization
PPO	Proximal Policy Optimization
GARCH	Generalized AutoRegressive Conditional Heteroskedasticity
A2C	Advantage Actor Critic
A3C	Asynchronous Advantage Actor Critic
DDPG	Deep Deterministic Policy Gradient
SAC	Soft Actor-Critic
TD3	Twin Delayed DDPG
GDQN	Gated DQN
GDPG	Gated Deterministic Policy Gradient
DARL	Data Augmentation based Reinforcement Learning
SentARL	Sentiment-Aware RL

References

Ahmed, D. M., Hassan, M. M., & Mstafa, R. J. (2022). A review on deep sequential models for forecasting time series data. *Applied Computational Intelligence and Soft Computing*, *2022*.

Bajpai, S. (2021). Application of deep reinforcement learning for Indian stock trading automation. arXiv preprint arXiv:2106.16088.

Brim, A., & Flann, N. S. (2022). Deep reinforcement learning stock market trading, utilizing a CNN with candlestick images. *PLoS ONE*, *17*(2), Article e0263181.

Busoniu, L., Babuska, R., & De Schutter, B. (2006). Multi-agent reinforcement learning: A survey. In *2006 9th international conference on control, automation, robotics and vision* (pp. 1–6). IEEE.

Cao, Y., & Zhao, X. (2023). Dwtformer: Wavelet decomposition transformer with 2D variation for long-term series forecasting. *Vol. 6*, In *2023 IEEE 6th information technology,networking,electronic and automation control conference (ITNEC)* (pp. 1548–1558).

Carta, S. (2021). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *51*(2).

Chakole, J., & Kurhekar, M. (2020). Trend following deep Q-learning strategy for stock trading. *Expert Systems*, *37*(4), Article e12514.

Chakraborty, S. (2019). Capturing financial markets to apply deep reinforcement learning. arXiv preprint arXiv:1907.04373.

Chen, F. (2022). Deep neural network model forecasting for financial and economic market. *Journal of Mathematics*, *2022*.

Chen, C.-T., Chen, A.-P., & Huang, S.-H. (2018). Cloning strategies from trading records using agent-based reinforcement learning algorithm. In *2018 IEEE international conference on agents (ICA)* (pp. 34–37).

Chen, L., & Gao, Q. (2019). Application of deep reinforcement learning on automated stock trading. In *2019 IEEE 10th international conference on software engineering and service science (ICSESS)* (pp. 29–33).

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), 3094.

Cornalba, F., Disselkamp, C., Scassola, D., & Helf, C. (2022). Multi-objective reward generalization: Improving performance of deep reinforcement learning for selected applications in stock and cryptocurrency trading. arXiv:2203.04579.

Dang, Q.-V. (2020). Reinforcement learning in stock trading. In *Advanced computational methods for knowledge engineering: proceedings of the 6th international conference on computer science, applied mathematics and applications, ICCSAMA 2019 6* (pp. 311–322).

Gao, X. (2018). Deep reinforcement learning for time series: playing idealized trading games. arXiv preprint arXiv:1803.03916.

Ge, J., Qin, Y., Li, Y., Huang, y., & Hu, H. (2022). Single stock trading with deep reinforcement learning: A comparative study. In *2022 14th international conference on machine learning and computing (ICMLC)* (pp. 34–43).

Hasselt, H. V., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double Q-learning. arXiv:1509.06461.

He, F.-F., Chen, C.-T., & Huang, S.-H. (2023). A multi-agent virtual market model for generalization in reinforcement learning based trading strategies. *Applied Soft Computing*, *134*, Article 109985.

Hendershott, T., Jones, C. M., & Menkveld, A. J. (2011). Does algorithmic trading improve liquidity? *The Journal of Finance*, *66*(1), 1–33.

Huang, C. Y. (2018). Financial trading as a game: A deep reinforcement learning approach. arXiv preprint arXiv:1807.02787.

Huang, Y., Gao, Y., Gan, Y., & Ye, M. (2021). A new financial data forecasting model using genetic algorithm and long short-term memory network. *Neurocomputing*, *425*, 207–218.

Jeong, G. H., & Kim, H. Y. (2019). Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning. *Expert Systems with Applications*, *117*, 125–138.

Kirisci, M., & Cagcag Yolcu, O. (2022). A new CNN-based model for financial time series: TAIEX and FTSE stocks forecasting. *Neural Processing Letters*, *54*(4), 3357–3374.

Lee, J. W., & O, J. (2002). A multi-agent Q-learning framework for optimizing stock trading systems. In *Database and expert systems applications: 13th international conference, DEXA 2002 Aix-En-Provence, France, September 2–6, 2002 proceedings 13* (pp. 153–162). Springer.

Lei, K., Zhang, B., Li, Y., Yang, M., & Shen, Y. (2019). Time-driven feature-aware jointly deep reinforcement learning for financial signal representation and algorithmic trading. *Expert Systems with Applications*, *140*, Article 112872.

Lele, S., Gangar, K., Daftary, H., & Dharkar, D. (2020). Stock market trading agent using on-policy reinforcement learning algorithms. Available at SSRN 3582014.

Li, Y., & Chen, Y. (2021). Enhancing a stock timing strategy by reinforcement learning. *IAENG International Journal of Computer Science*, *48*, 1–10.

Li, Y., Liu, P., & Wang, Z. (2022). Stock trading strategies based on deep reinforcement learning. *Scientific Programming*, *2022*, Article e4698656. http://dx.doi.org/10.1155/2022/4698656.

Li, Y., Ni, P., & Chang, V. (2020). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*, *102*(5).

Li, Y., Zheng, W., & Zheng, Z. (2019). Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access*, *7*, 108014–108022. http://dx.doi.org/10.1109/ACCESS.2019.2932789.

Li, Y., Zheng, W., & Zheng, Z. (2019). Deep robust reinforcement learning for practical algorithmic trading. *IEEE Access*, *7*, 108014–108022.

Liang, M., Wu, S., Wang, X., & Chen, Q. (2022). A stock time series forecasting approach incorporating candlestick patterns and sequence similarity. *Expert Systems with Applications*, *205*, Article 117595.

Lima Paiva, F. C., Felizardo, L. K., Bianchi, R. A. d. C., & Costa, A. H. R. (2021). Intelligent trading systems: a sentiment-aware reinforcement learning approach. In *Proceedings of the second ACM international conference on AI in finance* (pp. 1–9).

Lin, Y., Yan, Y., Xu, J., Liao, Y., & Ma, F. (2021). Forecasting stock index price using the CEEMDAN-LSTM model. *The North American Journal of Economics and Finance*, *57*, Article 101421.

Liu, F., Li, Y., Li, B., Li, J., & Xie, H. (2021). Bitcoin transaction strategy construction based on deep reinforcement learning. *Applied Soft Computing*, *113*, Article 107952.

Liu, Y., Liu, Q., Zhao, H., Pan, Z., & Liu, C. (2020). Adaptive quantitative trading: An imitative deep reinforcement learning approach. *Vol. 34*, In *Proceedings of the AAAI conference on artificial intelligence* (02), (pp. 2128–2135).

Liu, X.-Y., Yang, H., Chen, Q., Zhang, R., Yang, L., Xiao, B., & Wang, C. D. (2020). Finrl: A deep reinforcement learning library for automated stock trading in quantitative finance. arXiv preprint arXiv:2011.09607.

Liu, P., Zhang, Y., Bao, F., Yao, X., & Zhang, C. (2023). Multi-type data fusion framework based on deep reinforcement learning for algorithmic trading. *Applied Intelligence*, *53*(2), 1683–1706.

Ma, C., Zhang, J., Liu, J., Ji, L., & Gao, F. (2021). A parallel multi-module deep reinforcement learning algorithm for stock trading. *Neurocomputing*.

Mahayana, D., Shan, E., & Fadhl'Abbas, M. (2022). Deep reinforcement learning to automate cryptocurrency trading. In *2022 12th international conference on system engineering and technology (ICSET)* (pp. 36–41). IEEE.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602.

Moody, J. E., & Saffell, M. J. (1998). Reinforcement learning for trading. *Advances in Neural Information Processing Systems*, *17*(5–6), 917–923.

Moody, J., & Saffell, M. (2001). Learning to trade via direct reinforcement. *IEEE Transactions on Neural Networks*, *12*(4), 875–889.

Nan, A., Perumal, A., & Zaiane, O. R. (2022). Sentiment and knowledge based algorithmic trading with deep reinforcement learning. In *Database and expert systems applications: 33rd international conference, DEXA 2022, Vienna, Austria, August 22–24, 2022, Proceedings, Part I* (pp. 167–180). Springer.

Nuti, G., Mirghaemi, M., Treleaven, P., & Yingsaeree, C. (2011). Algorithmic trading. *Computer*, *44*(11), 61–69.

Ponomarev, E., Oseledets, I. V., & Cichocki, A. (2019). Using reinforcement learning in the algorithmic trading problem. *Journal of Communications Technology and Electronics*, *64*, 1450–1457.

Sagiraju, K., & Shashi, M. (2021). Reinforcement learning algorithms for automated stock trading. *Advances in Dynamical Systems and Applications (ADSA)*, *16*(2), 1021–1032.

Sezer, O. B., Gudelek, M. U., & Ozbayoglu, A. M. (2019). Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. arXiv:1911.13288.

Shavandi, A., & Khedmati, M. (2022). A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets. *Expert Systems with Applications*, *208*, Article 118124.

Shi, Y., Li, W., Zhu, L., Guo, K., & Cambria, E. (2021). Stock trading rule discovery with double deep Q-network. *Applied Soft Computing*, *107*, Article 107320.

Shin, H.-G., Ra, I., & Choi, Y.-H. (2019). A deep multimodal reinforcement learning system combined with CNN and LSTM for stock trading. In *2019 international conference on information and communication technology convergence (ICTC)* (pp. 7–11). IEEE.

Si, W., Li, J., Ding, P., & Rao, R. (2017). A multi-objective deep reinforcement learning approach for stock index future's intraday trading. *Vol. 2*, In *2017 10th international symposium on computational intelligence and design (ISCID)* (pp. 431–436).

Taghian, M., Asadi, A., & Safabakhsh, R. (2022). Learning financial asset-specific trading rules via deep reinforcement learning. *Expert Systems with Applications*, Article 116523.

Teng, X., Zhang, X., & Luo, Z. (2022). Multi-scale local cues and hierarchical attention-based LSTM for stock price trend prediction. *Neurocomputing*, *505*, 92–100.

Théate, T., & Ernst, D. (2021). An application of deep reinforcement learning to algorithmic trading. *Expert Systems with Applications*, *173*, Article 114632.

Treleaven, P., Galas, M., & Lalchand, V. (2013). Algorithmic trading review. *Communications of the ACM*, *56*(11), 76–85.

Tsai, Y.-C., Szu, F.-M., Chen, J.-H., & Chen, S. Y.-C. (2022). Financial vision-based reinforcement learning trading strategy. *Analytics*, *1*(1), 35–53.

Vishal, M., Satija, Y., & Babu, B. S. (2021). Trading agent for the Indian stock market scenario using actor-critic based reinforcement learning. In *2021 IEEE international conference on computation system and information technology for sustainable solutions (CSITSS)* (pp. 1–5).

Wang, Z., Lu, W., Zhang, K., Li, T., & Zhao, Z. (2021). A parallel-network continuous quantitative trading model with GARCH and PPO. arXiv preprint arXiv:2105.03625.

Wang, Y., & Yan, G. (2021). Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*.

Wen, M., Li, P., Zhang, L., & Chen, Y. (2019). Stock market trend prediction using high-order information of time series. *Ieee Access*, *7*, 28299–28308.

Wu, X., Chen, H., Wang, J., Troiano, L., & Fujita, H. (2020). Adaptive stock trading strategies with deep reinforcement learning methods. *Information Sciences*, *538*.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., & Long, M. (2023). TimesNet: Temporal 2D-variation modeling for general time series analysis. In *The eleventh international conference on learning representations*.

Wu, J. M.-T., Li, Z., Herencsar, N., Vo, B., & Lin, J. C.-W. (2021). A graph-based CNN-LSTM stock price prediction algorithm with leading indicators. *Multimedia Systems*, 1–20.

Wu, H., Xu, J., Wang, J., & Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Neural Information Processing Systems*.

Xiao, X. (2023). Quantitative investment decision model based on PPO algorithm. *Highlights in Science, Engineering and Technology*, *34*, 16–24.

Yang, Z., Keung, J. W., Kabir, M. A., Yu, X., Tang, Y., Zhang, M., & Feng, S. (2021). AComNN: Attention enhanced compound neural network for financial time-series forecasting with cross-regional features. *Applied Soft Computing*, *111*, Article 107649.

Yuan, Y., Wen, W., & Yang, J. (2020). Using data augmentation based reinforcement learning for daily stock trading. *Electronics*, *9*(9), 1384.

Zhou, P., & Tang, J. (2021). Improved method of stock trading under reinforcement learning based on DRQN and sentiment indicators ARBR. arXiv preprint arXiv:2111.15356.