Expert Systems With Applications 208 (2022) 118124

Contents lists available at ScienceDirect

Expert Systems With Applications

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

A multi-agent deep reinforcement learning framework for algorithmic trading in financial markets

Ali Shavandi [a], Majid Khedmati [b,*]

[a] Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran
[b] Department of Industrial Engineering, Sharif University of Technology, Azadi Ave., Tehran 1458889694, Iran

ARTICLE INFO

Keywords:
Reinforcement learning
Multi-agent
Algorithmic trading
Multi-timeframe
Deep Q-learning

ABSTRACT

Algorithmic trading based on machine learning is a developing and promising field of research. Financial markets have a complex, uncertain, and dynamic nature, making them challenging for trading. Some financial theories, such as the fractal market hypothesis, believe that the markets behave based on the collective psychology of investors who trade with different investment horizons and interpretations of information. Accordingly, a multi-agent deep reinforcement learning framework is proposed in this paper to trade on the collective intelligence of multiple agents, each of which is an expert trader on a specific timeframe. The proposed framework works in a hierarchical structure in which the flow of knowledge is from the agents trading at higher timeframes to the agents trading at lower timeframes, making them highly robust to the noise in financial time series. The Deep Q-learning algorithm is utilized for training the agents in the framework. The performance of the proposed framework is evaluated through extensive numerical experiments conducted on a historical dataset of the EUR/USD currency pair. The results demonstrate that the proposed multi-agent framework, based on several return-based and risk-based performance measures, outperforms single independent agents and several benchmark trading strategies in all investigated trading timeframes. The robust performance of the multi-agent framework throughout the trading period makes it suitable for algorithmic trading in financial markets.

1. Introduction

Financial trading is a research area that has received significant attention from researchers; accordingly, various methods have been developed for trading in financial markets. The existing approaches for financial trading can be categorized into traditional and modern methods. The traditional trading methods can be further divided into the technical analysis (Murphy, 1999) and fundamental analysis (Abarbanell & Bushee, 1997). Similarly, the modern trading methods can be further divided into algorithmic trading (Chaboud, Chiquoine, Hjalmarsson, & Vega, 2014) and machine learning (ML) approaches (Moody, Wu, Liao, & Saffell, 1998). The traditional financial trading methods are often implemented manually by humans, while the modern methods are executed automatically by computers. Algorithmic trading and ML methods have recently attracted growing interest from hedge funds, individual investors, and other stakeholders of financial markets. The increasing complexity of markets, abundance of data, growing number of market stakeholders, and obsolescence of the traditional trading methods have prompted the need for modern trading methods, such as algorithmic trading, more than ever.

Algorithmic trading can be performed by either rule-based methods or ML-based approaches. In rule-based algorithmic trading, computers trade in financial markets through predetermined rules specified by humans. These rules can be defined based on the traditional trading methods, mathematical models, or human-made strategies. On the other hand, in ML-based algorithmic trading, computers are trained on historical data to trade in markets without any direct human interventions. Algorithmic trading has many advantages over the traditional trading methods, including:

- Speed: Computers are very fast in executing trading decisions.
- Precision: Manual trading can be prone to human errors, while computers are more precise in executing trading decisions than humans.
- Rationality: Humans might make wrong trading decisions regarding their emotional and psychological factors, while computers are not negatively affected by these factors in making the trading decisions.

* Corresponding author.
E-mail addresses: Ali.shavandi@ie.sharif.edu (A. Shavandi), khedmati@sharif.edu (M. Khedmati).

https://doi.org/10.1016/j.eswa.2022.118124
Received 22 December 2021; Received in revised form 10 June 2022; Accepted 7 July 2022
Available online 11 July 2022
0957-4174/© 2022 Elsevier Ltd. All rights reserved.

- Processing ability: Computers are far more capable than human minds of processing a tremendous amount of information in real-time.
- Vigilance: Computers can monitor the markets permanently. They do not get tired or sleep and have immediate reactions when unexpected events occur.

Moreover, algorithmic trading using ML-based approaches has additional advantages over rule-based algorithmic trading. To name a few, i) ML-based algorithms can extract patterns, relationships, and knowledge from the historical data without the need for predetermined guidelines or strategies by domain experts, and ii) ML-based algorithmic trading approaches can discover profitable insights unknown to humans.

ML has received significant attention from researchers. Numerous supervised, unsupervised, and semi-supervised algorithms have been developed to extract knowledge from various types of data so far. In the early stages, most ML algorithms were limited to only making predictions. After that, methods were developed that could make decisions by interacting with an environment and improving a policy via self-learning. These methods were known as adaptive optimal control and eventually led to the emergence of reinforcement learning (RL). Most of the research efforts in applying ML-based methods for financial trading, such as McNally, Roche, and Caton (2018) and Yoo, Kim, and Jan (2005), attempt to predict the future prices by training supervised learning models on financial time series historical data. However, due to the noise, uncertainty, and volatility in financial time series, the accuracy of their predictive models might not be satisfactory for trading in markets. Furthermore, in addition to the prediction accuracy, a comprehensive trading strategy should consider a variety of financial facets, including risk management, quick response to unexpected events, and other essential factors related to trading strategies. Therefore, financial trading is a multi-faceted and complex decision-making problem. ML algorithms such as deep reinforcement learning (DRL) can autonomously make optimal decisions in complex environments (e.g., financial markets). This ability makes ML an interesting research topic in financial trading. Recently, DRL has achieved remarkable successes in solving complex sequential decision-making problems (Mnih et al., 2015; Silver et al., 2016). The intrinsic advantage of DRL is the ability to learn and improve a policy while interacting with a dynamic environment, which makes it suitable for algorithmic trading.

RL problems can be formulated as multi-agent systems consisting of several agents that can discover solutions through self-learning and interaction. Multi-agent systems are applied in various domains such as economics, robotics, distributed control, and telecommunications. The research studies in the field of RL have mainly focused on single agents. Fewer studies in this field have investigated multi-agent RL (Busoniu, Babuska, & De Schutter, 2006), especially in the financial trading domain (Lee & Park, 2007; Lussange, Lazarevich, Bourgeois-Gironde, Palminteri, & Gutkin, 2020).

According to the efficient market hypothesis (EMH) (Fama, 1970), it is impossible to predict prices in an efficient market based on historical data. The fractal market hypothesis (FMH), proposed by Peters (1994), is an alternative hypothesis to the widely known efficient market hypothesis. The FMH speculates that the behaviour of price in the markets does not resemble a random walk, as believed in the EMH, but in fact, the price has a fractal property with a similar structure at different time intervals. The FMH explains market behaviour by fractals, chaos, crises, and crashes in the market. Like the FMH, Elliott wave theory (Frost, Prechter, & Collins, 1999) speculates that the price has a recurrent behaviour at different timeframes. This theory claims that the markets behave on the collective psychology of traders in repetitive cycles. Accordingly, a set of sub-waves at a lower timeframe form a corresponding major wave at a higher timeframe. The major wave itself belongs to a set of sub-waves at a lower timeframe that creates a corresponding major wave at a higher timeframe. The behaviour of major waves at the higher timeframes is similar to that of the sub-waves at the lower timeframes and vice versa, as shown in Fig. 1.

Some research studies speculate that the fractal property of the markets might be caused by interactions of agents (traders) with different investment horizons and interpretations of information (Anderson & Noss, 2013). Therefore, in each specific timeframe, financial trading has particular characteristics. In the real world, every trader is often an expert for trading on a certain timeframe. Accordingly, it requires that for each specific timeframe, an expert agent be trained concerning the particular attributes of that timeframe. Furthermore, according to FMH, there might be interrelations in different timeframes due to different interpretations of information in each timeframe. Thus, the expert trading agents trained on different timeframes can learn the interactions between various timeframes by cooperation. This paper aims to design a multi-agent DRL trading framework for learning the interactions between different timeframes where each agent is an expert trader on a specific timeframe. The main question of this research is whether the proposed framework outperforms the single RL agents trained independently on different timeframes.

It should be noted that the research studies performed on RL-based algorithmic trading, in the literature, mainly apply a single timeframe's historical data to train a single agent. The main contribution of this paper is to introduce a new framework for learning the interactions between various timeframes by collective intelligence of multiple agents, each of which is an expert trader on a specific timeframe. To the best of the authors' knowledge, the multi-agent DRL trading framework proposed in this paper has not been investigated previously in the literature. The steps of this research are as follows: first, a general RL model with its assumptions for financial trading is presented. The DQN algorithm is then introduced to train the single independent agents. After that, the proposed framework is presented, and the algorithm for training multiple cooperative agents is designed. Finally, some numerical experiments are conducted to evaluate and demonstrate the superiority of the proposed framework over single independent agents and other benchmark algorithms.

The remainder of this paper is organized as follows. In section 2, a brief overview of the ML-based methods and a comprehensive review of the RL-based methods for financial trading is presented. Section 3 describes the trading problem, assumptions, algorithms, and the multi-agent trading framework. The performance of the proposed framework is evaluated and validated in section 4. Finally, the concluding remarks and recommendations for future research are provided in section 5.

2. Literature review

In recent years, numerous research studies have been conducted in the field of financial trading, and researchers have become increasingly

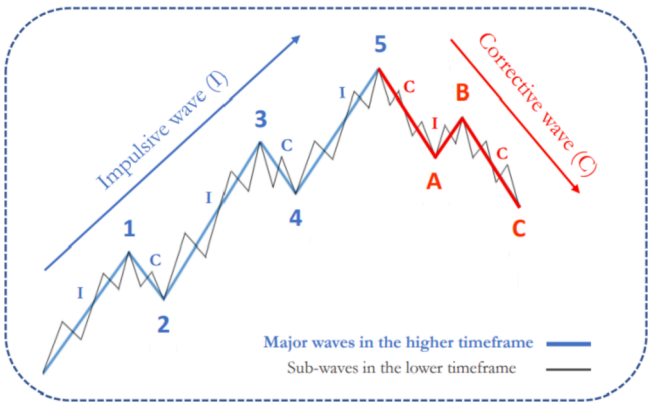

Fig. 1. Recurrent behaviour of price in Elliot wave theory.

interested in algorithmic and ML-based financial trading methods. Algorithmic trading is an automated trading approach that makes trading decisions based on predefined rules. These predefined rules can be created by human experts or discovered by ML methods. The significant advantage of ML-based financial trading methods is that they can discover profitable trading knowledge unknown to humans. This section provides a brief overview of ML-based approaches and a detailed review of RL-based approaches in financial trading.

2.1. Machine learning in financial trading

Most of the research studies in the area of ML-based financial trading have focused on predicting future prices using supervised learning models. In this regard, Lee (2009) applied a support vector machine (SVM) with a hybrid feature selection method to predict the stock trends. Guresen, Kayakutlu, and Daim (2011) used artificial neural networks for stock market index prediction. Due to the sequential nature of financial time series, recurrent neural networks (RNNs) and long short-term memory networks (LSTMs) are among the most investigated ML methods in the literature for financial time series prediction (Selvin, Vinayakumar, Gopalakrishnan, Menon, & Soman, 2017). Applications of ensemble ML have also been widely studied in the literature. For instance, Carta, Corriga, Ferreira, Recupero, and Saia (2019) proposed an auto-configurable ensemble ML strategy for financial trading. In their proposed ensemble, two sets of parameters, including time-series and classifier parameters, were tuned for each market. Moreover, Barbosa and Belo (2010) proposed a multi-agent trading architecture to trade currency pairs based on an ensemble of classification and regression models. Text mining and sentiment analysis are other ML methods investigated in the literature to predict markets (Nguyen, Shirai, & Velcin, 2015).

Although the supervised ML methods have shown a potential ability to predict financial markets such as the foreign exchange (Forex) market, their prediction accuracy might not be enough to make them applicable for algorithmic trading in real markets due to the noisy, volatile, and uncertain nature of financial time series. In addition to prediction accuracy, other essential factors such as risk management and adaptive decision-making should be considered to create a successful algorithmic trading strategy. Compared to supervised ML methods, few research studies have been conducted on applications of RL in financial trading. However, research interest in RL-based approaches for financial trading is increasing after the advent of deep learning and recent developments in DRL (Mnih et al., 2015).

2.2. Reinforcement learning in financial trading

After the emergence of supervised and unsupervised ML methods in financial trading, RL-based approaches were introduced for automated trading in this field of research. Initial explorations of RL-based financial trading appeared in the 1990s using recurrent RL (RRL) to optimize trading (Moody et al., 1998). Then, the value-based and policy-based RL algorithms were introduced to perform trading in various financial markets. After the advent of DRL, researchers started to explore its application in financial markets (Almahdi & Yang, 2017; Deng, Bao, Kong, Ren, & Dai, 2017), and numerous research studies have been conducted on applications of DRL in financial trading since then (Carapuço, Neves, & Horta, 2018; Li et al., 2019b; Tsantekidis et al., 2021b).

The applications of RL in financial trading were first investigated by Moody et al. (1998). They employed RRL to optimize utility functions directly and measure the performance of financial investments. Their approach created a new ML-based research trend that differed from the trend of supervised ML methods in financial trading. After that, researchers began to explore various RL-based approaches for financial trading. For instance, Almahdi and Yang (2017) proposed an RRL method to solve an optimal asset allocation problem. They optimized the allocation of assets in a portfolio concerning the expected maximum drawdown. The applications of RRL have also been investigated by Gold (2003) for high-frequency trading (HFT) in the Forex market. Moreover, Gabrielsson and Johansson (2015) implemented RRL to perform HFT in the futures market. They used Japanese candlestick chart features, which are more informative than close-price series, to design their proposed model's state space. Dempster and Leemans (2006) applied adaptive RL to create an automated trading system. Their system operated based on a layered structure to automate the trading in the Forex market. The layered structure included a layer of RRL algorithm, a layer of risk management mechanism, and a layer of dynamic utility optimization. After the advent of DRL in 2015, one of the initial explorations of it in the field of financial trading was proposed by Deng et al. (2017). They combined deep learning with RRL to design a model that could trade and process financial signals directly. In addition to a deep neural network, they used a fuzzy network to reduce the noise and learn features in the price time-series data.

After exploring the applications of RRL, researchers started to investigate those of value-based RL in financial trading. Gao and Chan (2000) introduced one of these applications by optimizing the Sharp ratio with Q-Learning. They proposed a combined model of RL and genetic algorithm in which the genetic algorithm was used to select the most profitable strategy. In their research, the technical indicators that composed the rules of trading strategies were used as input for the Q-Learning algorithm that optimized the strategies learned through experience from an unknown environment. Pendharkar and Cusatis (2018) introduced another trading application of value-based RL to manage a retirement investment portfolio with two assets. In this regard, they applied SARSA and Q-Learning as on-policy and off-policy algorithms, respectively. A majority of RL-based financial trading research studies have focused on signaling. Only a few studies have addressed the execution of the trading decisions. One of these studies belongs to Nevmyvaka, Feng, and Kearns (2006), proposing a value-based RL algorithm to execute the trading orders. The input data for their proposed algorithm was taken from NASDAQ order books, which contained millisecond order data for one and a half years. Regarding their achievements, the RL-based trading systems can potentially be developed to implement the whole trading process in real-world conditions.

After the advent of DRL, explorations of value-based DRL in financial trading began. In this regard, Carapuço et al. (2018) investigated the performance of the DQN algorithm in the Forex market. They considered small granularities of price time-series and adapted them to various time windows to create the state space of their proposed model. Sornmayura (2019) used the DQN algorithm to create a robust Forex trading system trained on EUR/USD and USD/JPY currency pairs. The algorithm was significantly superior to the buy and hold strategy and outperformed an experienced trader's performance. Li et al. (2019a), in addition to the DQN algorithm, investigated two other value-based RL algorithms, namely double DQN and dueling DQN. Zarkias, Passalis, Tsantekidis, and Tefas (2019) exploited DQN to implement a price trend tracking mechanism. To prevent over-fitting and reduce the sensitivity of algorithms to the noise in price time-series, they introduced a price trailing approach in which the agent was motivated to follow the price trend instead of trying to predict future prices. The price trailing mechanism turns the algorithmic trading into a control problem in which the agent aims to control its position in the market. Ensemble value-based DRL methods have also been investigated in the literature for financial trading. In this regard, Leem and Kim (2020) proposed an action-specialized ensemble learning approach based on DQN for automated trading. They conducted a specialized training process for each of the agent's actions. Carta, Corriga, Ferreira, Podda, and Recupero (2021) employed DRL to create a multi-layer and multi-ensemble intraday stock trader. They fused the decisions of several trading agents to create a robust trading strategy. To cope with the high noise and uncertainty of the market environment, Fengqian and Chao (2020) applied DRL with decomposing candlestick features to trade financial markets. They used candlesticks as a generalization of price movements over a time period

and decomposed them to create the input of a DRL model. The multi-objective systems have also been investigated by using value-based DRL in financial trading. For instance, Bisht and Kumar (2020) proposed a multi-objective DRL model to maximize profit by adjusting risk. The value-based DRL research studies mainly used time-series data as input for DRL models. However, in the study of Suhail et al. (2022), the outputs of sentiment analysis on daily market news were used as the inputs of a trading DRL model.

Policy-based RL has some potential advantages over value-based RL in financial trading, especially when dealing with continuous action spaces. The research studies discussed in the following include the policy-based RL methods in financial trading. García-Galicia, Carsteanu, and Clempner (2019) proposed a continuous-time RL model to solve the portfolio management problem. They used an actor-critic structure to implement RL and achieved the optimal policy with a proximal optimization method that included a time penalty on commissions and rewards. Their study used RL to estimate the transition and reward matrices. Jiang and Liang (2017) implemented DRL to solve the investment portfolio management problem in the cryptocurrency market. They employed a deterministic policy gradient algorithm with a convolutional neural network (CNN) as a policy network for a trading agent. Li et al. (2019b) used DQN and Asynchronous Advantage Actor-Critic (A3C) algorithms with LSTM networks and anti-noise autoencoders for function approximation to perform financial trading. Zhang, Zohren, and Roberts (2020) applied DRL to design trading strategies for future contracts. They considered discrete and continuous action spaces and utilized three different RL algorithms including DQN, Policy Gradient, and Advantage Actor-Critic (A2C). Tsantekidis et al. (2021b) employed double DQN value-based, and Proximal Policy Optimization (PPO) policy-based, with RNN networks to implement a price trailing mechanism. Ensemble strategies that use policy-based DRL have also been developed for financial trading. In this regard, Yang, Liu, Zhong, and Walid (2020) created an ensemble trading strategy using three policy-based RL algorithms: PPO, A2C, and deep deterministic policy gradient. Training reliability of RL agents is crucial, especially when dealing with the environments of financial markets. Tsantekidis, Passalis, and Tefas (2021a) employed a PPO algorithm with a neural network distillation approach to stabilize the training process of trading agents in the noisy environments of financial markets. The main contribution of their research was to use distillation from diversified ensembles of teacher agents, which traded in different currencies, to guide the training process of student agents.

Up to this point, few research studies have been conducted on applications of multi-agent RL in financial trading. One of these studies belongs to Lee and Park (2007), in which they used Q-Learning to train a multi-agent RL framework. To determine the best time and price for ordering, they divided the trading process into two parts: scheduling and pricing. Accordingly, the segmentation of the trading process created two different types of agents: signaling and ordering agents. They trained a buy signal agent and a sell signal agent as signaling agents. Similarly, a buy order agent and a sell order agent were trained as ordering agents. The ordering agents chose the best price on a trading day to execute the signals issued by the signaling agents. Another study of multi-agent DRL belongs to AbdelKawy, Abdelmoez, and Shoukry (2021), proposing a multi-stock trading model based on synchronous multi-agent DRL. The applications of multi-agent RL have also been investigated in liquidation strategy analysis (Bao & Liu, 2019) and modeling stock markets (Lussange et al., 2020). A brief review of the research papers in the RL-based financial trading literature is provided in Table 1.

To the best of authors' knowledge, none of the previous research studies in RL-based financial trading literature have investigated the multi-agent framework proposed in this paper to train multiple expert agents on multiple distinct timeframes to perform trading based on the collective intelligence of the agents.

3. Material and methods

In this section, the concept of RL is introduced first. Then, the trading problem is formulated as a Markov decision process (MDP). After that, the value-based DQN algorithm is presented to train each trading agent on its specific timeframe. Finally, a multi-agent DRL framework is proposed to perform financial trading based on the collective intelligence of multiple expert agents.

Table 1
A brief review of research studies in the literature of RL-based financial trading.

Author(s)	Recurrent	Value-based	Policy-based	Deep	Multi-agent or ensemble	Multi-timeframe
Moody et al. (1998)	✓					
Gold (2003)	✓					
Gao and Chan (2000)		✓				
Dempster and Leemans (2006)	✓					
Nevmyvaka et al. (2006)		✓				
Lee and Park (2007)		✓			✓	
Gabrielsson and Johansson (2015)	✓					
Almahdi and Yang (2017)	✓					
Deng et al. (2017)	✓			✓		
Jiang and Liang (2017)			✓	✓		
Pendharkar and Cusatis (2018)		✓				
Carapuço et al. (2018)		✓		✓		
Sornmayura (2019)		✓		✓		
Li et al. (2019a)		✓		✓		
Zarkias et al. (2019)		✓		✓		
García-Galicia et al. (2019)		✓	✓			
Li et al. (2019b)		✓	✓	✓		
Leem and Kim (2020)		✓		✓	✓	
Zhang et al. (2020)		✓	✓	✓		
Yang et al. (2020)		✓	✓	✓	✓	
Fengqian and Chao (2020)		✓		✓		
Bisht and Kumar (2020)		✓		✓		
Carta et al. (2021)		✓		✓	✓	
Tsantekidis et al. (2021a)			✓	✓	✓	
Tsantekidis et al. (2021b)		✓	✓	✓		
AbdelKawy et al. (2021)		✓	✓	✓	✓	
Suhail et al. (2022)		✓		✓		
Current research		✓		✓	✓	✓

3.1. Reinforcement learning concept

Reinforcement learning is an ML approach in which an agent interacts with an environment and tries to improve its decision-making by receiving feedback from the consequences of the decisions in the environment (Sutton & Barto, 1998). The main framework of RL consists of two components: the agent and the environment. As a result of the interaction between the agent and the environment in a recurring cycle, a chain of states, actions, and rewards is created (Fig. 2). According to this framework, an intelligent agent performs an action (A_t) concerning the current state at time t (S_t) and receives a reward (R_{t+1}) as a result of the performed action. Then, the agent observes a new state (S_{t+1}) and performs the next action based on the new state. This iterative framework continues until the agent converges to an optimal policy by maximizing a notion of cumulative reward.

It should be noted that the framework of multi-agent RL is similar to that of single-agent RL, except that in multi-agent RL, instead of one agent, several agents interact with one environment to achieve their desired goals (Fig. 3). The interaction of these agents can be cooperative, competitive, or a combination of cooperative and competitive (Busoniu et al., 2006).

3.2. Problem formulation and assumptions

Financial trading can be formulated as an RL problem in which an agent interacts iteratively with an unknown environment (market) and tries to converge to an optimal policy. A trading problem can be formulated as an MDP with a (S, A, P, R, γ) tuple where S is a set of states $\{s_1, s_2, \cdots, s_m\}$, A is a set of available actions $\{a_1, a_2, \cdots, a_n\}$ at each state, P is a transition probability matrix that transfers the environment from s_t to s_{t+1} based on the action at time t, R is a set of rewards that depends on the action taken at each state, and γ is a discount factor that takes a value between [0, 1]. The discount factor makes a trade-off between short-term and long-term rewards. In an RL problem, at each time t, the agent performs $a_t \in A$ in $s_t \in S$ and receives $r_t \in R$. Then, the environment state transfers from s_t to s_{t+1} according to transition probabilities P. The agent's goal is to maximize a cumulative reward $G_t = \sum_{i=0}^{T} \gamma^i r_{t+i+1}$ where γ is the discount factor. The formulation of RL problems is vitally important; even the best algorithms cannot achieve desired results without appropriate models. An MDP formulation of financial trading by an agent on a specific timeframe is provided in this section. Every single independent agent and the existing agents in the proposed multi-agent framework will trade based on this formulation.

3.2.1. State space

In the MDP formulation of financial trading, the state space (S) includes the agent's observation of the market at each time step. The environment of financial markets is very complex and vast. In real-world conditions, a comprehensive observation of the market environment is unlikely, and hence, a partial perception of the market environment, such as observing historical prices or social sentiment, is typically considered as the state space. Accordingly, we consider *OHLCV* (open, high, low, close, and volume) data for the state space as a standard representation of price movements at different granularities. At each time step t, the *OHLCV* is represented as $(OHLCV)_t$. Depending on the trading timeframe, each agent observes a sequence of n previous *OHLCV*s for its specific timeframe as s_t.

$$s_t = \left((OHLCV)_t, (OHLCV)_{t-1}, \cdots, (OHLCV)_{t-(n-1)} \right) \quad (1)$$

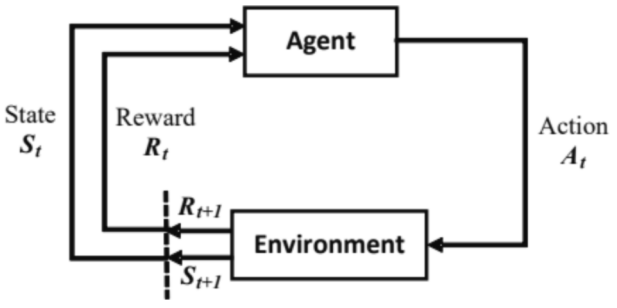

Fig. 2. Reinforcement learning framework.

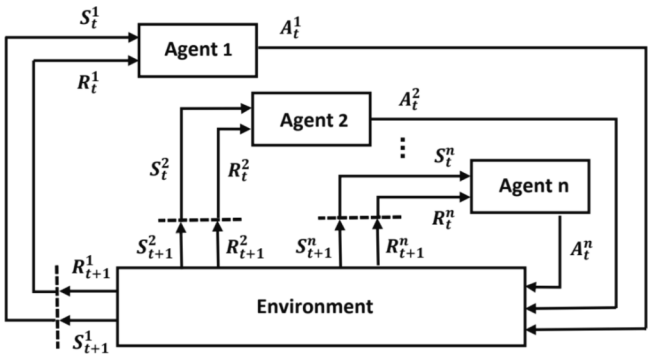

Fig. 3. Multi-agent reinforcement learning framework.

3.2.2. Action space

In the proposed framework, the agents trade a single asset in a two-sided market at the same time. In the RL-based financial trading literature, the simplest action space for trading a single asset in a two-sided market consists of three actions: long, wait, and short. According to the results of some experiments, this action space has some limitations when a trading agent performs actions in consecutive time steps, especially when there is no notion of commission cost in the structure of the reward function. Suppose the agent performs actions in consecutive time steps. In that case, it does not learn to control and hold its position in the market; as a result, the agent constantly changes its trading position without getting adequate feedback on its actions in the training process. To stabilize the training process, we introduce a mechanism to freeze the agent's trading position for a fixed time window (*tw*) after it performs an action at a time step. This time window should also be considered in the structure of the reward function, as the action space and reward function are closely related to each other.

The logic behind freezing the agent's action for a fixed time window is consistent with trading in real-world conditions. The best time for closing or opening a trading position depends on the trading strategy. There are various strategies for financial trading, some of which are short-term, some medium-term, and some long-term. However, one thing is common among the various strategies; when a trading position opens at a certain timeframe, the closing of that position usually occurs several time steps ahead in the same timeframe, rather than closing at exactly the next time step in that timeframe. For example, an intraday trader opens a position on a day and closes the position before the market closes on the same day. The intraday trader does not look at a daily timeframe chart for opening or closing trading positions but at the lower timeframe charts. The typical timeframes for identifying opportunities for opening and closing positions in intraday trading are usually between 30 min and four hours. Thus, if an intraday trader looks at a 1-hour timeframe chart for decision making, closing an opened position usually does not occur at exactly the next time step, one hour after opening. Accordingly, the proposed mechanism in this research freezes the actions for a fixed time window after being performed at each time step before performing the next action (Fig. 4). The time window should be tuned for each trading timeframe, separately.

The action space of the trading agents is formulated as Equation (2).

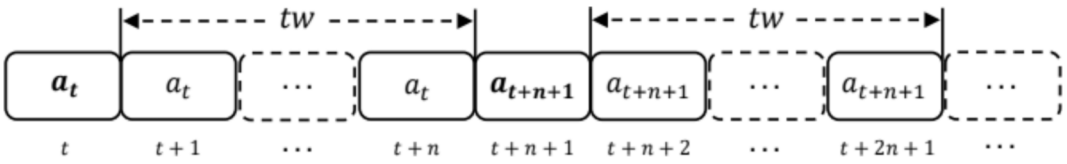

Fig. 4. Action stabilization mechanism.

$$a_t = \begin{Bmatrix} 1: Close\ the\ previous\ position\ (if\ any)\ and\ open\ a\ long\ position\ for\ tw, \\ 0: Close\ the\ previous\ position\ (if\ any)\ and\ do\ nothing, \\ -1: Close\ the\ previous\ position\ (if\ any)\ and\ open\ a\ short\ position\ for\ tw \end{Bmatrix} \tag{2}$$

3.2.3. Reward function

Another critical component of a reinforcement learning model is the reward function. The reward function directly affects the agent's learning process by providing feedback for its actions. The feedback for a trading agent can be the profit or loss received from a trading position. According to the action stabilization mechanism, the agent stays in that position for a fixed time window after opening a position. Therefore, the feedback received from each trading position should be commensurate with the time window for that position. If we consider the close price at time t as p_t, the realized reward at time t can be calculated based on Equation (3).

$$r_t = \frac{(p_{t+\mathrm{tw}} - p_t)}{p_t} \times a_t \tag{3}$$

According to our observations, when the value of *tw* is greater than one, the agent has a better performance in trading, so it can be concluded that the feedback of the reward function usually contains more noise as the value of *tw* gets closer to one.

3.2.4. Transition probabilities and discount factor

The other main components of a reinforcement learning model are the transition probabilities and discount factor. Transition probabilities are the market dynamics that transfer the environment from s_t to s_{t+1}. In model-free RL problems, transition probabilities are not required in advance as the agents learn them by interacting with the environment over time. The discount factor is a scalar value between [0, 1] that discounts the future rewards over time. The optimal value of this parameter can be achieved by parameter tuning methods.

3.2.5. Assumptions

Financial markets are inherently complex, and creating a comprehensive trading system requires numerous considerations. Suppose financial trading is a process; various issues must be considered from the beginning, when the data is received and preprocessed, to the end of the process, when the trading signals are executed in the market. This research proposes and evaluates a multi-agent DRL trading framework for generating trading signals in the financial trading process. Accordingly, several assumptions have been considered to simplify the trading process in this research, as follows:

- There is no market gap or slippage. Therefore, the trading signals are executed immediately at the exact prices determined by the agents.
- The commission fee is zero; hence the transaction costs are not involved in the structure of the reward function.
- Order size is 100% of the available balance for opening a trading position.
- The agents trade only with their balance; hence no leverage.
- The agents' actions have a negligible impact on the market transitions; accordingly, the market impact is considered to be zero.

3.3. DQN algorithm

Q-Learning is a value-based RL algorithm developed by Watkins and Dayan (1992). In this algorithm, the agent learns a value function $Q_\pi(s, a)$ that represents an expected cumulative reward when the agent performs action a in state s, and then follows the policy π. This function is represented as Equation (4).

$$Q_\pi(s,a) \doteq \mathbb{E}_\pi[G_t | S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = s, A_t = a\right] \tag{4}$$

where π is the agent's policy and G_t is the cumulative reward given by Equation (5).

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{5}$$

The number of calculations in RL algorithms can be reduced significantly by simplifying Equation (5) as follows.

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \cdots = r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} + \cdots)$$

$$= r_{t+1} + \gamma G_{t+1} \tag{6}$$

In this algorithm, the agent performs a_t in s_t and receives r_t. Then, the Q-values are updated using the Bellman equation (Bellman, 1966), given by Equation (7).

$$Q_\pi(s_t, a_t) = r_{t+1} + \gamma max_a Q_\pi(s_{t+1}, a) \tag{7}$$

The agent approximates Q-values iteratively using Equation (7) as it interacts with the environment until it converges to the optimal Q-function $Q^*(s, a)$. In many real-world problems, the state and action spaces expand exponentially; as a result, the number of Q-value pairs explodes, leading to the *curse of dimensionality*. When the environment's space dimensions increase, it is challenging and sometimes impossible to update the Q-values since the agent will not be able to explore the environment. This issue eventually leads to sub-optimal or non-optimal policies. In this regard, DQN uses deep learning (Q-network) as a function approximator to overcome the curse of dimensionality. In this algorithm, artificial neural networks approximate $Q(s, a; \theta)$, where θ is the matrix of parameters in the Q-network. The objective of the algorithm is to find the optimal parameters (θ^*) by minimizing the loss function formulated as Equation (8):

$$\mathrm{L}(\theta_i) = \mathrm{E}[\![(y - Q(s_t, a_t; \theta_i))^2]\!] \tag{8}$$

where y is the updated Q-value with regard to previous parameters of the Q-network (θ_{i-1}), which is given by Equation (9):

$$y = r_t + \gamma max_a Q(s_{t+1}, a; \theta_{i-1}) | s_t, a_t \tag{9}$$

At each iteration i, $\mathrm{L}(\theta_i)$ can be minimized using the stochastic gradient descent (SGD) algorithm as represented by Equation (10):

$$\nabla_{\theta_i} \mathrm{L}(\theta_i) = \mathrm{E}[\![(r_t + \gamma max_a Q(s_{t+1}, a; \theta_{i-1}) - Q(s_t, a_t; \theta_i)) \nabla_{\theta_i} Q(s_t, a_t; \theta_i)]\!] \tag{10}$$

where $\nabla_{\theta_i} Q(s_t, a_t; \theta_i)$ is calculated by the back-propagation algorithm. In this paper, the root-mean-squared propagation (RMSProp) is employed as an extension of SGD to minimize $\mathrm{L}(\theta_i)$.

RL-based financial trading can be considered as both episodic and

non-episodic problems. The issue concerned with the episodic RL problem is that it repeats using the same data for training the agent. In this case, the agent memorizes the same time series as the number of episodes increases rather than learning the market dynamics. Therefore, the agent's performance during the out-of-sample period will not be as good as its performance during the in-sample period. There are fewer data points for training the agents at higher timeframes, so the problem has been usually defined episodically in previous research studies. In this research, lower timeframes (1 h and below) are considered to train the agents, so the problem is approached non-episodically. The DQN algorithm proposed by Mnih et al. (2015) is adopted to train the trading agents. This algorithm has been modified to fit the trading model defined in this research. We have employed the algorithm for both training the independent agents on distinct timeframes (Algorithm 1) and training the agents in the proposed multi-agent framework in which the cooperative agents are trained on multiple timeframes (Algorithm 2).

Algorithm 1
The DQN trading algorithm for independent agents

Input: experience replay memory and its size (N), Q network and its parameters, target network $\widehat{Q}$ and its parameters, market environment for the trading timeframe (S), the time window of action stabilization (tw), look back period (n), the initial value of exploration parameter (ε) and its decay rate, number of steps (C) to restart weights of the target network, discount factor (γ)
Output: optimal parameters of Q network
Initialize the experience replay memory M to capacity N
Initialize Q network with random weights θ
Initialize target network $\widehat{Q}$ with weights $\theta^{-} = \theta$
Set *terminal* = last time step in the *OHLCV* dataset (market environment)
Set tw = time window of action stabilization
Set $w = 0$
Set n = look back period
Initialize s_n to a sequence of n previous *OHLCV*s for the trading timeframe
For $t = n, terminal - tw$ **do**
  if $w = 0$ **then**
    With probability ε, select a random action a_t from $\{1, 0, -1\}$
    Otherwise, select $a_t = argmax_{a \in A} Q(s_t, a; \theta)$
    $a_c = a_t$
    $w = tw$
  else
    $a_t = a_c$
  Perform a_t and get a new observation $s_{t+1} = \left((OHLCV)_{t+1}, (OHLCV)_t, \cdots, (OHLCV)_{t-n+2}\right)$
  Get the reward $r_t = \frac{(p_{t+tw} - p_t)}{p_t} \times a_t$
  Store experience $e_t = (s_t, a_t, r_t, s_{t+1})$ in M
  Randomly sample a minibatch of experiences (s_i, a_i, r_i, s_{i+1}) from M
  Set $y_i = r_i + \gamma max_a \widehat{Q}(s_{i+1}, a; \theta^{-})$
  Perform a gradient descent step on $\left(y_i - Q(s_i, a_i; \theta)\right)^2$ considering network parameters θ
  Reset $\widehat{Q} = Q$ every C steps
  Decay the ε-Greedy exploration parameter ε.
  $s_t = s_{t+1}$
  $w = w - 1$
End For

3.4. Multi-agent DRL trading framework

In the proposed multi-agent trading framework, several agents, each of which is an expert trader on a specific timeframe, trade collaboratively. This framework aims to perform financial trading on the collective intelligence of multiple expert agents to obtain better performance than the intelligence of each independent expert agent. In this framework, Λ is a set of agents $\{agent_1, agent_2, \cdots, agent_n\}$ where $agent_i$ is an expert trader on timeframe tf^i. As the value of i increases, the trading timeframe increases such that $tf^{i+1} > tf^i$. Consequently, $agent_1$ and $agent_n$ trade on the lowermost and uppermost timeframes, respectively. As the trading timeframe increases, both the noise and the details of the price movements decrease. The noise reduction can improve the feedback of the environment in the agent's learning process. However, the lack of details in price movements might deprive the agent of more profitable trading opportunities. On the other hand, as the trading timeframe decreases, although the details of price movements increase, more noise at lower timeframes can destabilize the agent's learning process.

Accordingly, the expert agents at lower timeframes, in addition to benefiting from the details of price movements in their specific timeframe, can be more resistant to the noise of their timeframe, by receiving feedback from the expert agents at higher timeframes. Besides, a hierarchical feedback transfer mechanism with a flow of information from the agents at higher timeframes to the agents at lower timeframes can transfer the knowledge and experience of all trading agents to the agent at the lowermost timeframe so that it becomes a super-intelligent trading agent. Therefore, the final trading signal is always issued by the lowermost timeframe's agent, namely $agent_1$. The feedback transfer mechanism is designed in such a way that the state space of each agent consists of a sequence of n previous *OHLCV*s for its specific timeframe as well as the current trading position of all agents trading on higher timeframes. For instance, $agent_i$ in addition to observing a sequence of n previous *OHLCV*s for tf^i, observes the current trading position of $agent_{i+1}, agent_{i+2}, \cdots, agent_n$ as its state at time step t ($s_t^{agent_i}$ or s_t^i). Obviously, $agent_n$ does not observe the trading position of any agent, and no agent observes the trading position of $agent_1$. The current trading position of all agents is stored in a position pool $P = \{a_c^1, a_c^2, \cdots, a_c^n\}$ where a_c^i is the current trading position of $agent_i$. The position pool is accessible to all agents so that $agent_i$ can observe the current trading position of all trading agents at higher timeframes ($tf^{i+1}, tf^{i+2}, \cdots, tf^n$) in a set of $\Phi_c^{>i} = \left\{a_c^j \in P | j > i\right\}$. A schematic representation of the hierarchical feedback transfer mechanism in the proposed multi-agent trading framework is represented in Fig. 5.

This framework is inspired by the fractal nature of markets (Peters, 1994). In the fractal theory, there are parental patterns in which the child patterns are born similarly. Analogously, in financial markets, it can be assumed that the patterns of lower timeframes originate within the patterns of higher timeframes. According to this inspiration from the fractal theory, the proposed multi-agent trading framework is designed in a hierarchical procedure in which the flow of information is from the expert agents at higher timeframes to the expert agents at lower timeframes. Regarding the fractal nature of trading timeframes, the behaviour of price in higher timeframes influences the behaviour of price in lower timeframes. In other words, in a bullish market, although there are downward fluctuations in lower timeframes of prices, the upward movement of prices in higher timeframes overcomes these fluctuations. On the other hand, in a bearish market, the upward price fluctuations in lower timeframes do not have the power to overcome the downward movement of prices in higher timeframes. Accordingly, the effect of information in the lower timeframes is negligible on the agents that trade in the higher timeframes. Therefore, the flow of information in the proposed framework is top-down.

Algorithm 2
The DQN trading algorithm for multi-agent framework

Input (for each agent): experience replay memory and its size (N^i), Q^i network and its parameters, target network $\widehat{Q}^i$ and its parameters, market environment (S^i), the time window of action stabilization (tw^i), look back period (n^i), the initial value of exploration parameter (ε^i) and its decay rate, number of steps (C^i) to restart weights of the target network, discount factor (γ^i), trading timeframe (tf^i)
Output: optimal parameters of Q^i networks
For each agent **do**
  Set tf^i = trading timeframe for the agent i
  Initialize the experience replay memory M^i to capacity N^i

(*continued on next page*)

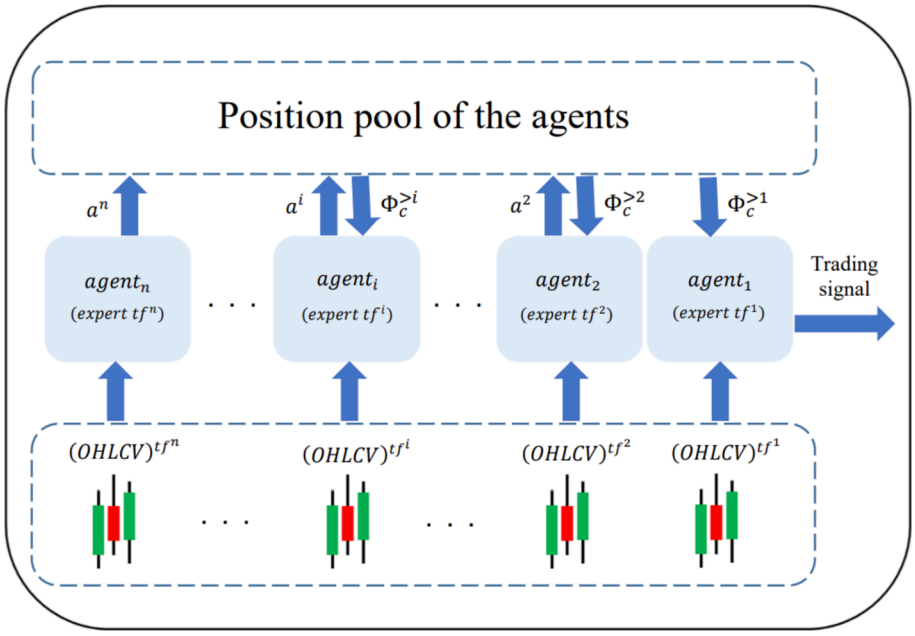

Fig. 5. The proposed multi-agent trading framework.

Algorithm 2 (*continued*)

Initialize Q^i network with random weights θ^i
Initialize target network $\widehat{Q}^i$ with weights $\theta^{-i} = \theta^i$
Set tw^i = time window of action stabilization for tf^i
Set $w^i = 0$
Set n^i = look back period
Initialize current position of the agent, a_c^i
Initialize s_t^i to a sequence of n previous $OHLCV^{tf^i}$ and $\Phi_c^{>i}$
End For
Repeat
For each agent **do**
if $OHLCV^{tf^i}$ is completed at the current time (candle is closed) **then**
if $w^i = 0$ **then**
With probability ε^i, select a random action a_t^i from $\{1, 0, -1\}$
Otherwise, select $a_t^i = argmax_{a^i} Q(s_t^i, a^i; \theta^i)$
$a_c^i = a_t^i$
$w^i = tw^i$
else
$a_t^i = a_c^i$
Observe $s_{t+1}^i = \left((OHLCV)_{t+1}^{tf^i}, (OHLCV)_t^{tf^i}, \cdots, (OHLCV)_{t-n+2}^{tf^i}, \Phi_c^{>i}\right)$
Get the reward $r_t^i = \dfrac{\left(p_{t+tw^i}^{tf^i} - p_t^{tf^i}\right)}{p_t^{tf^i}} \times a_t^i$.
Store experience $e_t^i = (s_t^i, a_t^i, r_t^i, s_{t+1}^i)$ in M^i
Randomly sample a minibatch of experiences $(s_j^i, a_j^i, r_j^i, s_{j+1}^i)$ from M^i
Set $y_j^i = r_j^i + \gamma^i max_{a^i} \widehat{Q}^i\left(s_{j+1}^i, a^i; \theta^{-i}\right)$
Perform a gradient descent step on $\left(y_j^i - Q\left(s_j^i, a_j^i; \theta^i\right)\right)^2$ considering network parameters θ^i
Reset $\widehat{Q}^i = Q^i$ every C^i steps
Decay the ε-Greedy exploration parameter ε^i
$s_t^i = s_{t+1}^i$
$w^i = w^i - 1$
End For
Until termination time

4. Numerical experiments

In this section, the performance of the proposed framework is evaluated based on several well-known performance metrics that include both return-based and risk-based measures. In this regard, details of the numerical experiments are presented at first. Then, the results of numerical experiments conducted on the single independent agents and the proposed multi-agent framework are presented and discussed. In addition to comparing the agents' performance with each other, their performance is also compared to some common trading strategies. Statistical hypothesis tests are also implemented to validate the obtained results.

4.1. Framework formulation

In the proposed framework, there is no limitation on the number of cooperating agents and accordingly the number of trading timeframes. In order to incorporate various timeframes with a reasonable computational burden for training the agents in the proposed framework, three different timeframes have been selected: 1-hour, 15-minute, and 5-minute. The reason for choosing these timeframes is that there are enough data points in each of the mentioned timeframes for the non-episodic training of the agents. For each timeframe, an expert agent is trained on the corresponding *OHLCV* to trade in that timeframe. These agents are referred to as long-term, mid-term, and short-term trading agents relative to each other. In this regard,

- The long-term agent ($agent_3$) is the expert trader of 1-hour timeframe (tf^3). This agent is referred to as H1 agent.
- The mid-term agent ($agent_2$) is the expert trader of 15-minute timeframe (tf^2). This agent is referred to as M15 agent.
- The short-term agent ($agent_1$) is the expert trader of 5-minute timeframe (tf^1). This agent is referred to as M5 agent.

According to the proposed framework, each agent at each timeframe observes n previous time steps of the corresponding *OHLCV* as well as the current trading position of its higher timeframe agents as $s_t^{agent_i}$. The state of agent H1, M15, and M5 at each time step t for their corresponding timeframes, namely when the candle is closed in that timeframe, is given in Equations (11)-(13), respectively.

$$s_t^{agent_3} = \left[OHLCV_t^{tf^3}, OHLCV_{t-1}^{tf^3}, OHLCV_{t-2}^{tf^3}, \cdots, OHLCV_{t-n}^{tf^3}\right] \tag{11}$$

$$s_t^{agent_2} = \left[OHLCV_t^{tf^2}, OHLCV_{t-1}^{tf^2}, OHLCV_{t-2}^{tf^2}, \cdots, OHLCV_{t-n}^{tf^2}, a_c^3\right] \tag{12}$$

$$s_t^{agent_1} = \left[OHLCV_t^{tf^1}, OHLCV_{t-1}^{tf^1}, OHLCV_{t-2}^{tf^1}, \cdots, OHLCV_{t-n}^{tf^1}, a_c^3, a_c^2\right] \quad (13)$$

According to Equation (11), the uppermost timeframe's trading agent, H1, only observes n previous time steps of its corresponding *OHLCV* as the state since there is no agent trading in a higher timeframe than 1-hour. The lowermost timeframe's agent, M5, is the super-intelligent agent that receives information from all trading agents at higher timeframes and issues the final trading signal. The sample of the proposed framework is shown in Fig. 6.

4.2. Data description

The foreign exchange market, also known as Forex, has been selected to conduct the numerical experiments. Forex is the largest financial market in the world in which the average daily trading turnover is by far more than any other financial market. Among all currency pairs in Forex, EUR/USD is the most traded one. Therefore, to obtain valid results, the numerical experiments are conducted on the historical *OHLCV* data of EUR/USD from 06/29/2012 to 05/25/2021. Specifications of the historical data for the selected timeframes are provided in Table 2.

It should be noted that numerous hyperparameters exist in DRL problems that are complex and time-consuming to be tuned simultaneously. Most of the hyperparameter tuning methods in the field of ML are based on trial and error. In this paper, the Grid search method is used for hyperparameter tuning by which the hyperparameters are tuned by examining the agents' performance on grids of possible hyperparameter values. The tuned values of hyperparameters for each independent agent are presented in Table 3.

The look back period specifies the number of previous time steps of *OHLCV*. The last layer of the agents' network consists of 3 neurons, namely, the size of available actions at each time step. The activation function for the last layer in the network of DQN is linear as the last layer approximates a real state-action value for each of the available actions.

4.3. The analysis and comparison of the results

In this section, independent non-episodic experiments are conducted on the proposed multi-agent framework and every independent single agent to evaluate their performance. The obtained results for 20 independent experiments are presented in Table 4. According to this table, the trading performance is evaluated based on eight performance metrics, including the average cumulative return, average annual return, max cumulative return, min cumulative return, win rate, Sharpe ratio, coefficient of variation, and the average of max drawdowns in 20 independent non-episodic experiments over the trading period.

Table 2
The specifications of historical data in each timeframe.

Timeframe	Currency pair	Trading period	Number of time steps
1-hour (H1)	EUR/USD	29/06/2012 to 25/05/2021	56,554
15-minute (M15)	EUR/USD	29/06/2012 to 25/05/2021	223,843
5-minute (M5)	EUR/USD	29/06/2012 to 25/05/2021	673,239

Table 3
The tuned values of DQN hyperparameters.

Hyperparameter	H1 agent	M15 agent	M5 agent
Number of Q-network layers	3	4	4
Number of neurons in each layer	(300, 150, 3)	(300, 200, 100, 3)	(400, 200, 100, 3)
Activation function of each layer	(Tanh, Tanh, Linear)	(Tanh, Tanh, Tanh, Linear)	(Tanh, Tanh, Tanh, Linear)
Learning rate (α)	0.00025	0.00025	0.00025
Replay memory size	10,000	40,000	120,000
Discount factor (γ)	0.45	0.5	0.6
Decay rate of ε	0.999	0.9999	0.99995
Look back period (n)	60	200	300
Time window (tw)	5	10	20

Table 4
Performance of the agents in the trading period.

Performance metric	Multi-agent framework	H1 agent	M15 agent	M5 agent
Average cumulative return	**56.4 %**	38.1 %	32.2 %	30.7 %
Average annual return	**5.1 %**	3.7 %	3.2 %	3.0 %
Max cumulative return	**91.2 %**	67.2 %	58.4 %	60.3 %
Min cumulative return	**27.5 %**	14.3 %	16.6 %	14.0 %
Win rate	**65.0 %**	62.1 %	59.0 %	61.0 %
Sharpe ratio	**0.63**	0.50	0.41	0.46
Coefficient of variation	**1.60**	2.01	2.43	2.05
Average of max drawdowns	**11.89 %**	12.86 %	14.70 %	12.63 %

The results indicate that the proposed multi-agent framework outperforms all the single independent agents in all the performance metrics. Among single independent agents, the H1 agent has a better performance in a majority of metrics. Although there are more data

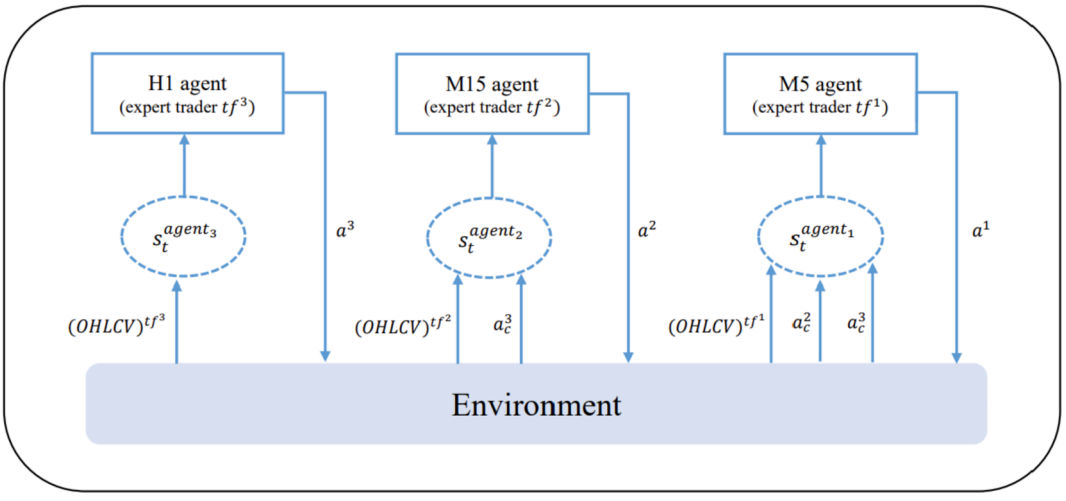

Fig. 6. Three agent sample of the proposed framework.

points in the lower timeframes to train the agents, the performance of the agents trading on the lower timeframes is inferior to those trading on the higher timeframes, in the same trading period. The cause of the worse performance of the agents at the lower timeframes might be their high noise. The flow of information from the agents at the higher timeframes to those at the lower timeframes enables the multi-agent framework to cope with the high noise in addition to benefiting from the details of price movements in the lower timeframes.

The cumulative return is the principal performance metric for evaluating the agents as it is the main feedback mechanism in the reward function for training the agents. Accordingly, a statistical analysis is performed on the cumulative returns in order to validate the obtained results. In this regard, three hypothesis tests are designed to compare the average cumulative return of the multi-agent framework, (μ_{MRL}), with that of the single independent agents, $(\mu_{H1}, \mu_{M15}, \mu_{M5})$. The mathematical representations of the null and alternative hypotheses for 1-tailed tests are represented as follows:

$$\text{Test 1}\rightarrow\begin{cases} H_0 : \mu_{MRL} = \mu_{M5} \\ H_1 : \mu_{MRL} > \mu_{M5} \end{cases},$$

$$\text{Test 2}\rightarrow\begin{cases} H_0 : \mu_{MRL} = \mu_{M15} \\ H_1 : \mu_{MRL} > \mu_{M15} \end{cases},$$

$$\text{Test 3}\rightarrow\begin{cases} H_0 : \mu_{MRL} = \mu_{H1} \\ H_1 : \mu_{MRL} > \mu_{H1} \end{cases}$$

The independent samples *t*-test is considered to examine these hypotheses. This test is valid when a number of assumptions are held, including independence of samples, normality of distribution, homogeneity of variances, and random sampling. As mentioned before, 20 independent experiments have been conducted in this research to obtain the results. Therefore, the assumptions of independence and random sampling are met. In order to test the normality of distributions, the Anderson-Darling test has been implemented. According to the obtained p-values, the null hypotheses of this test could not be rejected; namely, the cumulative returns for all single independent agents and the multi-agent framework are normally distributed. Given that we intend to compare the means of different groups, it is better to use Welch's *t*-test. This test is adapted from the *t*-test and is more reliable when the variances of groups are not homogenous. The t statistic and degrees of freedom associated with Welch's *t*-test are given in Equations (14) and (15).

$$t = \frac{\Delta\overline{X}}{s_{\Delta\overline{X}}} = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{s^2_{\overline{X}_1} + s^2_{\overline{X}_2}}}, s_{\overline{X}_i} = \frac{s_i}{\sqrt{N_i}} \tag{14}$$

$$\nu \approx \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2\nu_1} + \frac{s_2^4}{N_2^2\nu_2}}, \nu_i = N_i - 1 \tag{15}$$

The results obtained from this test are summarized in Table 5.

According to the results from Welch's *t*-test for testing the hypotheses, the cumulative return of the multi-agent framework is significantly higher than that of every independent agent. Given that the cumulative return has been the main feedback mechanism in the reward function for training the agents, the obtained results from the hypothesis tests validate the better performance of the proposed multi-agent framework.

In addition to comparing the performance of the proposed multi-agent framework with single independent agents, it is also compared to that of four common financial trading strategies as benchmark algorithms. These strategies consist of MACD, RSI, MovingAvg Cross, and the Buy and Hold (B&H) with default settings. The B&H strategy is assumed to be implemented in a short position on EUR/USD, meaning that with an optimistic assumption for B&H strategy in the trading period, a long position is opened on USD/EUR at the beginning of the trading period. All strategies are evaluated on the same trading period and with the same assumptions as RL agents. In this regard, the results of the benchmark strategies in the trading period are obtained and summarized in Table 6. Based on the obtained results, the performance of the proposed multi-agent framework is far superior to the common trading strategies in all performance metrics. It should be considered this fact that the performance of the single independent agents is significantly better than the benchmark strategies, meaning that RL agents potentially perform better than the rule-based strategies in algorithmic trading.

Furthermore, the average balance of the multi-agent framework and that of the single independent agents in 20 independent non-episodic experiments over the trading period is represented in Fig. 7. All the agents start trading with an initial balance of 1000$. As seen in the charts, the agents' balances from the beginning of the trading period to the middle of 2014 have almost a neutral trend because the agents have no experience and are in the training phase in this period. However, after obtaining enough experience, they all start a general upward trend in their balance until the end of the trading period. The proposed multi-agent framework has fewer fluctuations in its balance uptrend, indicating that it is more robust than single independent agents in financial trading. The proposed framework trades with the combined intelligence of the expert trading agents in all timeframes, leading to learning the interactions between various timeframes and performing better than every independent agent in the trading period.

Despite the large fluctuations of EUR/USD rate in the trading period, as shown in Fig. 8, all the trading agents are able to maintain reliable performance in all conditions, such as upward, downward, and neutral trends of the currency pair. For example, EUR/USD has experienced a sudden drop from the middle of 2014 to the beginning of 2015 and a sharp rise from 2017 to the beginning of 2018. However, the agents have performed consistently in these volatile periods. The robustness of the RL trading agents makes them suitable for algorithmic trading in all market conditions.

5. Conclusions and recommendations for the future research

In this paper, a multi-agent DRL framework is proposed to perform ML-based algorithmic trading in financial markets. The framework comprises multiple cooperative agents, each of which is an expert trader on a specific timeframe, to perform algorithmic trading based on the collective intelligence of several expert trading agents. A hierarchical

Table 6
Performance of the benchmark strategies in the trading period.

Performance metric	MovingAvg Cross	MACD	RSI	B&H
Cumulative return	−0.65 %	−16.00 %	−29.30 %	5.40 %
Average annual return	−0.07 %	−1.70 %	−2.90 %	0.60 %
Win rate	31.3 %	31.1 %	40.0 %	–
Max drawdown	17.16 %	27.96 %	32.62 %	21.00 %

Table 5
The obtained results of testing the hypotheses using Welch's *t*-test.

Test name	t statistic	p-value	Significance level	Degrees of freedom	Power of test	Result
Test 1	5.06	<0.00001	1%	37	98.3%	Reject the null hypothesis
Test 2	5.09	<0.00001	1%	37	96.3%	Reject the null hypothesis
Test 3	3.47	0.00067	1%	37	67.4%	Reject the null hypothesis

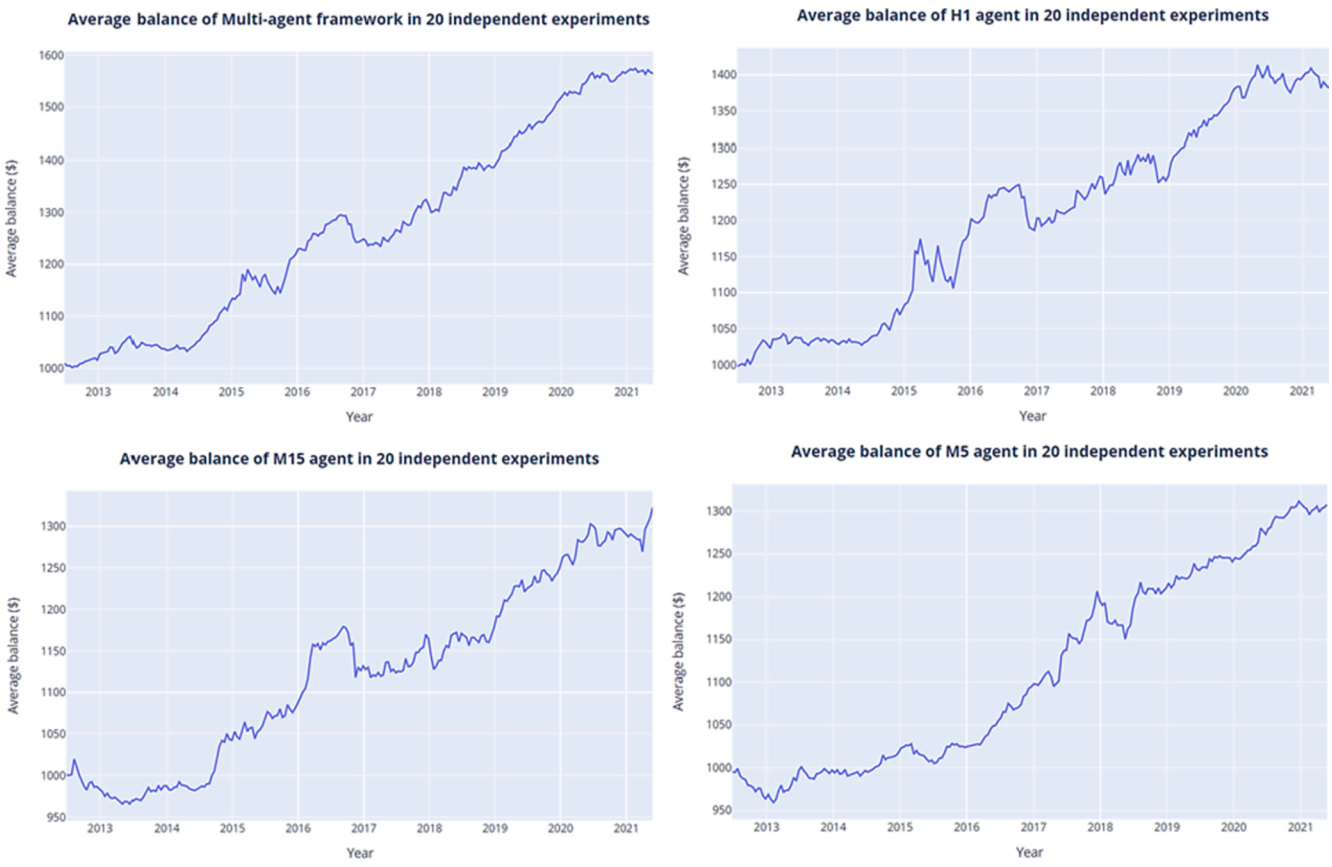

Fig. 7. The average balance of the agents in 20 independent non-episodic experiments.

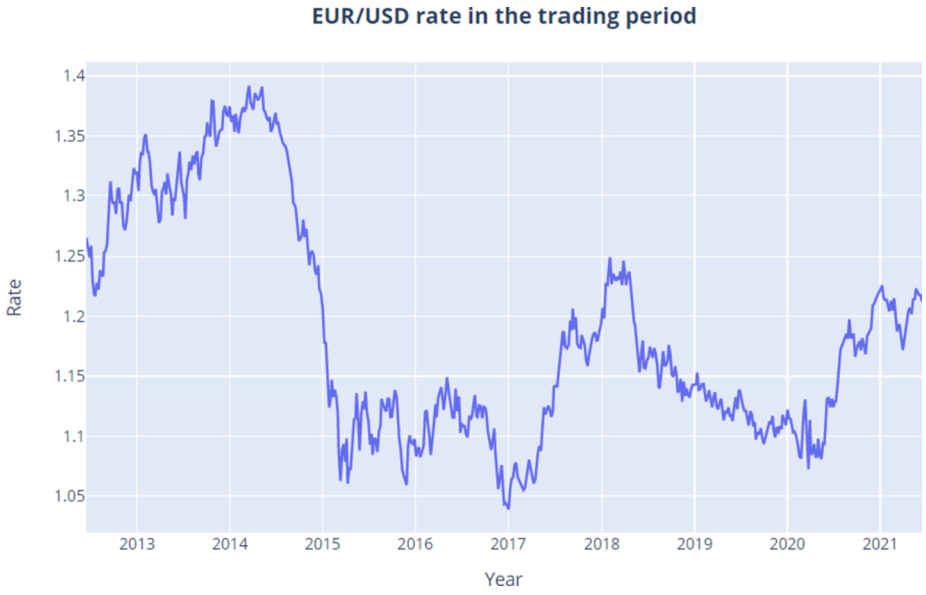

Fig. 8. EUR/USD rate in the trading period.

feedback transfer mechanism is designed to share information among the agents in a top-down flow. In order to train the multi-agent framework and single independent agents, the DQN algorithm is implemented. A sample of the proposed framework consisting of three agents, trading on long-term, mid-term, and short-term timeframes is formulated to evaluate the performance of the proposed framework. Some validated numerical experiments on nine years of EUR/USD historical data show that the proposed framework outperforms the single independent agents as well as the common trading strategies in all investigated performance metrics, including return-based and risk-based measures. Furthermore, the single independent agents perform better than the benchmark trading strategies based on the evaluation measures.

This research has several implications for science and practice. First, the intelligence of several expert trading agents that trade on multiple distinct timeframes can be aggregated through the proposed framework

to trade better than each expert agent alone. Second, the hierarchical feedback transfer mechanism that transfers the knowledge from the agents in a top-down flow enables the agents to cope with the noise in addition to benefiting from the details of price movement in their timeframe. Third, whether it is the multi-agent framework or an independent single agent, RL agents perform better than the benchmark strategies, which demonstrates the potential advantage of the RL-based trading strategies over the rule-based ones. It should be noted that, despite all the advantages, this research has also some limitations where, the main of which is the high computational burden of training and tuning the hyperparameters of several agents simultaneously. Another limitation is that the state and action spaces of the RL trading problem are designed very simplistic, while the environments of financial markets are highly complex and volatile. However, the purpose of this research was not to design a sophisticated model of the RL trading problem.

This research can be further expanded in several aspects in future research. First, the proposed framework can be improved by manipulating the state and action spaces of the trading problem, such as re-engineering the state input or preprocessing the historical data with a novel approach. Second, other DRL algorithms, such as policy-based ones, can be utilized to train the trading agents. Third, manipulating the agents' interaction can improve the suggested framework. Finally, this research can be further investigated by testing the multi-agent framework on different combinations of timeframes or assessing it in less uncertain environments than the Forex market.

CRediT authorship contribution statement

Ali Shavandi: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Visualization. **Majid Khedmati:** Conceptualization, Methodology, Formal analysis, Writing – review & editing, Supervision, 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.

References

Abarbanell, J. S., & Bushee, B. J. (1997). Fundamental Analysis, Future Earnings, and Stock Prices. *Journal of Accounting Research, 35*(1), 1–24. https://doi.org/10.2307/2491464

AbdelKawy, R., Abdelmoez, W. M., & Shoukry, A. (2021). A synchronous deep reinforcement learning model for automated multi-stock trading. *Progress in Artificial Intelligence, 10*(1), 83–97. https://doi.org/10.1007/s13748-020-00225-z

Almahdi, S., & Yang, S. Y. (2017). An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications, 87*, 267–279. https://doi.org/10.1016/j.eswa.2017.06.023

Anderson, N., & Noss, J. (2013). The Fractal Market Hypothesis and its implications for the stability of financial markets. *Bank of England Financial Stability Paper, 23*. https://ssrn.com/abstract=2338439.

Bao, W., & Liu, X. Y. (2019). Multi-Agent Deep Reinforcement Learning for Liquidation Strategy Analysis. *arXiv preprint arXiv:1906.11046*.

Barbosa, R. P., & Belo, O. (2010). Multi-agent forex trading system. In *Agent and Multi-Agent Technology for Internet and Enterprise Systems* (pp. 91–118). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-642-13526-2_5.

Bellman, R. (1966). Dynamic Programming. *Science, 153*(3731), 34–37. https://doi.org/10.1126/science.153.3731.34

Bisht, K., & Kumar, A. (2020, December). Deep Reinforcement Learning based Multi-Objective Systems for Financial Trading. In *2020 5th IEEE International Conference on Recent Advances and Innovations in Engineering (ICRAIE)* (pp. 1-6). IEEE. https://doi.org/10.1109/ICRAIE51050.2020.9358319.

Busoniu, L., Babuska, R., & De Schutter, B. (2006). *Multi-Agent Reinforcement Learning: A Survey*. 9th International Conference on Control, Automation, Robotics and Vision, https://doi.org/10.1109/icarcv.2006.345353.

Carapuço, J., Neves, R., & Horta, N. (2018). Reinforcement learning applied to Forex trading. *Applied Soft Computing, 73*, 783–794. https://doi.org/10.1016/j.asoc.2018.09.017

Carta, S., Corriga, A., Ferreira, A., Podda, A. S., & Recupero, D. R. (2021). A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning. *Applied Intelligence, 51*(2), 889–905. https://doi.org/10.1007/s10489-020-01839-5

Carta, S., Corriga, A., Ferreira, A., Recupero, D. R., & Saia, R. (2019). A holistic auto-configurable ensemble machine learning strategy for financial trading. *Computation, 7*(4), 67. https://doi.org/10.3390/computation7040067

Chaboud, A. P., Chiquoine, B., Hjalmarsson, E., & Vega, C. (2014). Rise of the Machines: Algorithmic Trading in the Foreign Exchange Market. *The Journal of Finance, 69*(5), 2045–2084. https://doi.org/10.1111/jofi.12186

Dempster, M. A. H., & Leemans, V. (2006). An automated FX trading system using adaptive reinforcement learning. *Expert Systems with Applications, 30*(3), 543–552. https://doi.org/10.1016/j.eswa.2005.10.012

Deng, Y., Bao, F., Kong, Y., Ren, Z., & Dai, Q. (2017). Deep Direct Reinforcement Learning for Financial Signal Representation and Trading. *IEEE Transactions on Neural Networks and Learning Systems, 28*(3), 653–664. https://doi.org/10.1109/tnnls.2016.2522401

Fama, E. F. (1970). Efficient Capital Markets: A Review of Theory and Empirical Work. *The Journal of Finance, 25*(2), 383–417. https://doi.org/10.2307/2325486

Fengqian, D., & Chao, L. (2020). An adaptive financial trading system using deep reinforcement learning with candlestick decomposing features. *IEEE Access, 8*, 63666–63678. https://doi.org/10.1109/ACCESS.2020.2982662

Frost, A. J., Prechter, R. R., & Collins, C. J. (1999). *Elliott Wave Principle: Key to Market Behaviour*. John Wiley & Sons.

Gabrielsson, P., & Johansson, U. (2015). *High-Frequency Equity Index Futures Trading Using Recurrent Reinforcement Learning with Candlesticks*. 2015 IEEE Symposium Series on Computational Intelligence, https://doi.org/10.1109/ssci.2015.111.

Gao, X., & Chan, L. (2000). An algorithm for trading and portfolio management using q-learning and sharpe ratio maximization. *In Proceedings of the international conference on neural information processing*, 832-837.

García-Galicia, M., Carsteanu, A. A., & Clempner, J. B. (2019). Continuous-time reinforcement learning approach for portfolio management with time penalization. *Expert Systems with Applications, 129*, 27–36. https://doi.org/10.1016/j.eswa.2019.03.055

Gold, C. (2003). *FX trading via recurrent reinforcement learning*. 2003 IEEE International Conference on Computational Intelligence for Financial Engineering, https://doi.org/10.1109/cifer.2003.1196283.

Guresen, E., Kayakutlu, G., & Daim, T. U. (2011). Using artificial neural network models in stock market index prediction. *Expert Systems with Applications, 38*(8), 10389–10397. https://doi.org/10.1016/j.eswa.2011.02.068

Jiang, Z., & Liang, J. (2017). *Cryptocurrency portfolio management with deep reinforcement learning*. 2017 Intelligent Systems Conference (IntelliSys), https://doi.org/10.1109/intellisys.2017.8324237.

Lee, M. C. (2009). Using support vector machine with a hybrid feature selection method to the stock trend prediction. *Expert Systems with Applications, 36*(8), 10896–10904. https://doi.org/10.1016/j.eswa.2009.02.038

Lee, J. W., Park, J., O, J., Lee, J., & Hong, E. (2007). A Multiagent Approach to Q-Learning for Daily Stock Trading. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, *37*(6), 864-877. https://doi.org/10.1109/tsmca.2007.904825.

Leem, J., & Kim, H. Y. (2020). Action-specialized expert ensemble trading system with extended discrete action space using deep reinforcement learning. *PloS one, 15*(7), e0236178. https://doi.org/10.1371/journal.pone.0236178

Li, Y., Ni, P., & Chang, V. (2019a). Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing, 102*(6), 1305–1322. https://doi.org/10.1007/s00607-019-00773-w

Li, Y., Zheng, W., & Zheng, Z. (2019b). Deep Robust Reinforcement Learning for Practical Algorithmic Trading. *IEEE Access, 7*, 108014–108022. https://doi.org/10.1109/access.2019.2932789

Lussange, J., Lazarevich, I., Bourgeois-Gironde, S., Palminteri, S., & Gutkin, B. (2020). Modelling Stock Markets by Multi-agent Reinforcement Learning. *Computational Economics, 57*(1), 113–147. https://doi.org/10.1007/s10614-020-10038-w

McNally, S., Roche, J., & Caton, S. (2018). *Predicting the Price of Bitcoin Using Machine Learning*. 2018 26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), https://doi.org/10.1109/pdp2018.2018.00060.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533. https://doi.org/10.1038/nature14236

Moody, J., Wu, L., Liao, Y., & Saffell, M. (1998). Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting, 17*(5–6), 441–470. https://doi.org/10.1002/(sici)1099-131x(1998090)17:5/6

Murphy, J. J. (1999). *Technical Analysis of the Financial Markets: A Comprehensive Guide to Trading Methods and Applications*. New York Institute of Finance.

Nevmyvaka, Y., Feng, Y., & Kearns, M. (2006). *Reinforcement learning for optimized trade execution*. Proceedings of the 23rd international conference on Machine learning – ICML '06, https://doi.org/10.1145/1143844.1143929.

Nguyen, T. H., Shirai, K., & Velcin, J. (2015). Sentiment analysis on social media for stock movement prediction. *Expert Systems with Applications, 42*(24), 9603–9611. https://doi.org/10.1016/j.eswa.2015.07.052

Pendharkar, P. C., & Cusatis, P. (2018). Trading financial indices with reinforcement learning agents. *Expert Systems with Applications, 103*, 1–13. https://doi.org/10.1016/j.eswa.2018.02.032

Peters, E. E. (1994). *Fractal Market Analysis: Applying Chaos Theory to Investment And Economics*. John Wiley & Sons.

Selvin, S., Vinayakumar, R., Gopalakrishnan, E. A., Menon, V. K., & Soman, K. P. (2017). *Stock price prediction using LSTM, RNN and CNN-sliding window model.* 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), https://doi.org/10.1109/icacci.2017.8126078.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961

Sornmayura, S. (2019). Robust forex trading with deep q network (dqn). *American Journal of Agricultural Economics, 39*(1), 15–33.

Suhail, K. M., Sankar, S., Kumar, A. S., Nestor, T., Soliman, N. F., Algarni, A. D., … Abd El-Samie, F. E. (2022). Stock Market Trading Based on Market Sentiments and Reinforcement Learning. *Computers, Materials & Continua, 70*(1), 935–950. https://doi.org/10.32604/cmc.2022.017069.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning* (Vol. 135). https://doi.org/10.1023/a:1025696116075

Tsantekidis, A., Passalis, N., & Tefas, A. (2021a). Diversity-driven knowledge distillation for financial trading using Deep Reinforcement Learning. *Neural Networks, 140*, 193–202. https://doi.org/10.1016/j.neunet.2021.02.026

Tsantekidis, A., Passalis, N., Toufa, A.-S., Saitas-Zarkias, K., Chairistanidis, S., & Tefas, A. (2021b). Price Trailing for Financial Trading Using Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems, 32*(7), 2837–2846. https://doi.org/10.1109/tnnls.2020.2997523

Watkins, C. J. C. H., & Dayan, P. (1992). Technical Note. In *Reinforcement Learning* (pp. 55–68). US: Springer.

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

Yoo, P. D., Kim, M. H., & Jan, T. (2005). *Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation*. International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06), https://doi.org/10.1109/cimca.2005.1631572.

Zarkias, K. S., Passalis, N., Tsantekidis, A., & Tefas, A. (2019). *Deep Reinforcement Learning for Financial Trading Using Price Trailing*. ICASSP 2019 – 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), https://doi.org/10.1109/icassp.2019.8683161.

Zhang, Z., Zohren, S., & Roberts, S. (2020). Deep Reinforcement Learning for Trading. *The Journal of Financial Data Science, 2*(2), 25–40. https://doi.org/10.3905/jfds.2020.1.030