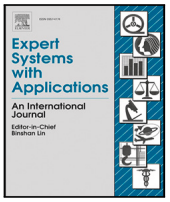
Expert Systems With Applications 235 (2024) 121245

Contents lists available at ScienceDirect

Expert Systems With Applications

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



Algorithmic trading using continuous action space deep reinforcement learning

Naseh Majidi[1], Mahdi Shamsi [*,1], Farokh Marvasti

Multimedia and Signal processing Lab (MSL), Advanced Communications Research Institute (ACRI), EE Department of Sharif University of Technology, Tehran, Islamic Republic of Iran

ARTICLE INFO

Keywords:
Algorithmic trading
Stock market prediction
risk management
Deep reinforcement learning
Financial markets

ABSTRACT

Finding a more efficient trading strategy has always been one of the main concerns in financial market trading. In order to create trading strategies that lead to higher profits, historical data must be used. Due to a large amount of financial data and various factors affecting them, algorithmic trading and, more recently, artificial intelligence are employed to overcome the decision-making complexity. This paper aims to introduce a new approach using Twin-Delayed DDPG (TD3) and the daily close price to create a trading strategy. As a continuous action space deep reinforcement learning algorithm, in contrast to the discrete ones, the TD3 provides us with both the number of trading shares and the trading positions. In order to evaluate the performance of the proposed algorithm, the comparison results of our approach and other commonly-used algorithms such as technical analysis, reinforcement learning, supervised learning, stochastic strategies, and deterministic strategies are reported. By employing both position and the number of trading shares, we show that the performance of a trading strategy can be improved in terms of Return and Sharpe ratio.

1. Introduction

Forecasting the price movements in a financial market has been of interest to both traders and researchers since more accurate price predictions can result in a higher profit. According to the Efficient-Market hypothesis (Kirkpatrick & Dahlquist, 2008), the stock market prices follow a random walk process with unpredictable future fluctuations. When it comes to cryptocurrencies such as Bitcoin, the price value severely fluctuates, which makes the price forecasting a challenging problem (Phaladisailoed & Numnonda, 2018). Technical and fundamental analyses are two typically-used tools to build trading strategies in the financial markets. Technical analysis provides trading signals according to the price movement and the trading volume (Murphy, 1999). Fundamental analysis, unlike the former, examines related economic and financial factors to determine a stock's underlying worth (Drakopoulou, 2016). Expert traders and computers both analyze financial data using technical or/and fundamental analysis. By keeping an eye on financial charts (such as prices), traders make decisions based on their own past experiences. However, due to several factors influencing the price movement, managing a vast volume of data can become unattainable, forcing us to use a computer to analyze the data. Moreover, the computer performance is not affected by emotions and environmental conditions, which is considered a disadvantage for human traders (Chakole, Kolhe, Mahapurush, Yadav, & Kurhekar, 2021). Therefore, algorithmic trading has emerged to tackle this issue. Trading in a financial market can be followed based on two different but related approaches: price prediction and algorithmic trading. The price prediction approach aims to build a model that can precisely predict future prices in a passive manner. On the other hand, an algorithmic trading method that employs computer algorithms and mathematical rules is not limited just to the price prediction. It attempts to actively participate in the financial market (e.g. choosing a position and the number of trading shares) to maximize profit (Hirchoua, Ouhbi, & Frikh, 2021; Théate & Ernst, 2021).

Classical Machine Learning (ML) and Deep Learning (DL) have shown outstanding performance in pattern recognition in various research fields, such as computer vision, image analysis, and healthcare (Esteva et al., 2019; Faes et al., 2019; Khan & Al-Habsi, 2020; Nair, Kumari, Tyagi, & Sravanthi, 2021). Considering the algorithmic trading, to the best of our understanding, the Reinforcement Learning (RL) algorithms can be more beneficial than the ML/DL algorithms in

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Corresponding author.
E-mail addresses: naseh.majidi@alum.sharif.edu (N. Majidi), Mahdi.Shamsi@alum.sharif.edu (M. Shamsi), marvasti@sharif.edu (F. Marvasti).

[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.eswa.2023.121245
Received 26 October 2022; Received in revised form 17 August 2023; Accepted 17 August 2023
Available online 20 August 2023

two aspects. Firstly, supervised learning is not satisfactory for learning problems with long-term and time-delayed rewards (Dang, 2019; Jangmin, Lee, Lee, & Zhang, 2006; Lee, 2001; Shi, Li, Zhu, Guo, & Cambria, 2021), such as trading in financial markets. In other words, the future decisions are affected by the rewards obtained by the trading agent (e.g., the available cash is a function of the reward, and this cash affects future decisions) while the successive decisions in supervised learning (both classification and regression) are made independently and only involve maximizing the accuracy of the prediction without considering the long-term effects (Dang, 2019). Furthermore, since a trader's overall loss due to incorrect actions may be greater than the overall profit due to correct ones, a more precise prediction (the supervised learning goal) does not necessarily result in a higher profit (Li, Zheng, & Zheng, 2019). Hence, the RL approach seems to be useful to solve such a decision-making problem (trading) in an uncertain environment (financial market). Although the RL may suffer from credit assignment, the actor–critic version of the RL (e.g., TD3) using temporal difference addresses this issue. Secondly, since there is no specific rule for labeling the data in the supervised learning approach, the performance of a well-trained model is heavily affected by the data preprocessing procedure. Therefore, it seems reasonable to employ an RL approach that does not require labels, and its policy is determined by properly defining its reward function.

A discrete action space RL algorithm has been commonly employed for algorithmic trading (Chakole et al., 2021; Jeong & Kim, 2019; Shi et al., 2021; Théate & Ernst, 2021), which restricts traders to buy/sell a specific number of shares. On the other hand, Jeong and Kim (2019), Li et al. (2019) showed that increasing the number of trading shares increases the profit. Hence, we propose a trading algorithm employing TD3, which is a continuous action space Deep Reinforcement Learning (DRL), to allow the trader to buy/sell a dynamic number of shares. The main benefit of our algorithm over the ones proposed in Jeong and Kim (2019), Li et al. (2019) is that there is no limit to the number of the trading shares. In other words, it is calculated based on the output of the TD3 algorithm and available cash.

The remainder of this paper is organized as follows: Section 2 reviews the articles on price prediction and algorithmic trading. The algorithm's components and the proposed trading algorithm are introduced in Section 3. Section 4 provides some baseline algorithms and standard metrics, as well as the evaluation of the algorithms. The final section discusses the findings and some recommendations for further works.

2. Related works

This section reviews some of the latest works focusing on the statistical machine learning techniques (Classical ML, DL, and RL) in financial market trading.

2.1. Statistical method, classical machine learning, and deep learning

Recent research trend in financial market prediction has focused on the statistical methods, ML, and DL with classification and regression approaches. For the stock index prediction, Autoregressive Integrated Moving Average (ARIMA) method has been utilized and shown to obtain acceptable results for short-term forecasting (Ariyo, Adewumi, & Ayo, 2014). By considering the sequential nature of the market data, Long Short-Term Memory (LSTM) and Recurrent Neural Network (RNN) have been used and it has been shown that the LSTM outperforms both the RNN and ARIMA (McNally, Roche, & Caton, 2018).

By successfully extracting informative features, Convolutional Neural Networks (CNN) have achieved great attentions in the research fields such as object detection and image recognition (Pathak, Pandey, & Rautaray, 2018; Traore, Kamsu-Foguem, & Tangara, 2018). By comparing different neural network architectures, it has been shown that the CNN outperforms the LSTMs and Multi-Layer Perceptions (MLPs) in forecasting S&P500 trend (Di Persio & Honchar, 2016). Considering the importance of feature preprocessing, the combination of the CNN and wavelet transform has shown performance enhancement in time series prediction. Authors in Hoseinzade and Haratizadeh (2019) merged different sources of data using 3D-CNNs and 2D-CNNs and showed that the CNN-based algorithms outperformed the technical indicators strategies in terms of the Sharpe ratio and the CEQ return. Researchers in Alonso-Monsalve, Suárez-Cetrulo, Cervantes, and Quintana (2020) showed that a combination of LSTM and CNN surpasses the CNN, since such a combination can detect both dependencies and local patterns in the price time series. Autoencoder is another architecture that has been used to forecast price. It has been reported that the Stacked Denoising Autoencoder (SDAE) outperforms the Support Vector Machine (SVM), Back Propagation Neural Network (BPNN), and Principal Component Analysis-based Support Vector Regression (PCA+SVR) in terms of Mean Absolute Percentage Error, Root Mean Squared Error, and Directional Accuracy (Liu, Li, Li, Zhu, & Yao, 2021).

Regarding the dataset, price is not the only possible input of a Deep Neural Network (DNN) model. Authors in Dutta, Kumar, and Basu (2020) used 15 variables such as google trend, price volatility, dollar index, and interest rates as the price predictors of different networks. Furthermore, researchers in Liu et al. (2021) employed 40 features to predict bitcoin's price. Concerning the network's depth, deeper networks have been shown to perform better than shallower ones (Liu, Tao, Tse, & Wang, 2022).

2.2. Reinforcement learning

The RL-based approaches have been commonly used in algorithmic trading. An RL agent interacts with its environment to learn a policy that maximizes the received reward. This procedure is quite similar to a situation in which someone is learning how to trade in financial markets. In an RL approach, the input data must be mapped into the states to obtain an optimal policy. The policy in algorithmic trading usually contains three actions: long/buy, short/sell, and hold.

Q-learning was utilized in Chakole et al. (2021) to find a trading strategy. Two techniques were employed to embed the OHLCV data (open, high, low, close price, and volume) into the RL states. The first approach utilizes the k-means method to divide states into n groups, while the second one quantizes the percentage change between close and open prices into six levels. According to the results, in most situations, the first approach outperforms the second one.

Furthermore, the DRL is another popular approach that researchers have recently employed. By feeding raw data into a neural network, Q-values or actions are estimated. As one of the latest developed DRL algorithms, Deep Q-Network (DQN) has been employed to handle financial trading decisions in Jeong and Kim (2019). In addition to the RL algorithm, their model is also formed of a DNN regressor in order to estimate the number of shares. Researchers in Li et al. (2019) have also looked at dynamic trading shares, where a model with seven actions outperformed a model with three actions. They added the SDAE and LSTM to Asynchronous Actor–Critic (A3C) and DQN to improve their performances. It has also been shown that the A3C-based algorithms outperform the DQN-based ones.

A DRL agent with a core of Double Deep Q-Network outperformed the LSTM and SVM models in terms of Return (Shi et al., 2021). Since DNN is used at the core of the DRL agent, it is reasonable to use some techniques such as batch Normalization, Dropout, and Gradient Clipping to avoid overfitting (Théate & Ernst, 2021). Furthermore, authors in Jeong and Kim (2019) used transfer learning in order to mitigate data insufficiency.

Researchers have also focused on portfolio management. Authors in Betancourt and Chen (2021) have created a technique that can trade on a dynamic number of assets as a version of algorithmic trading. It has been shown to be beneficial when it comes to the introduction

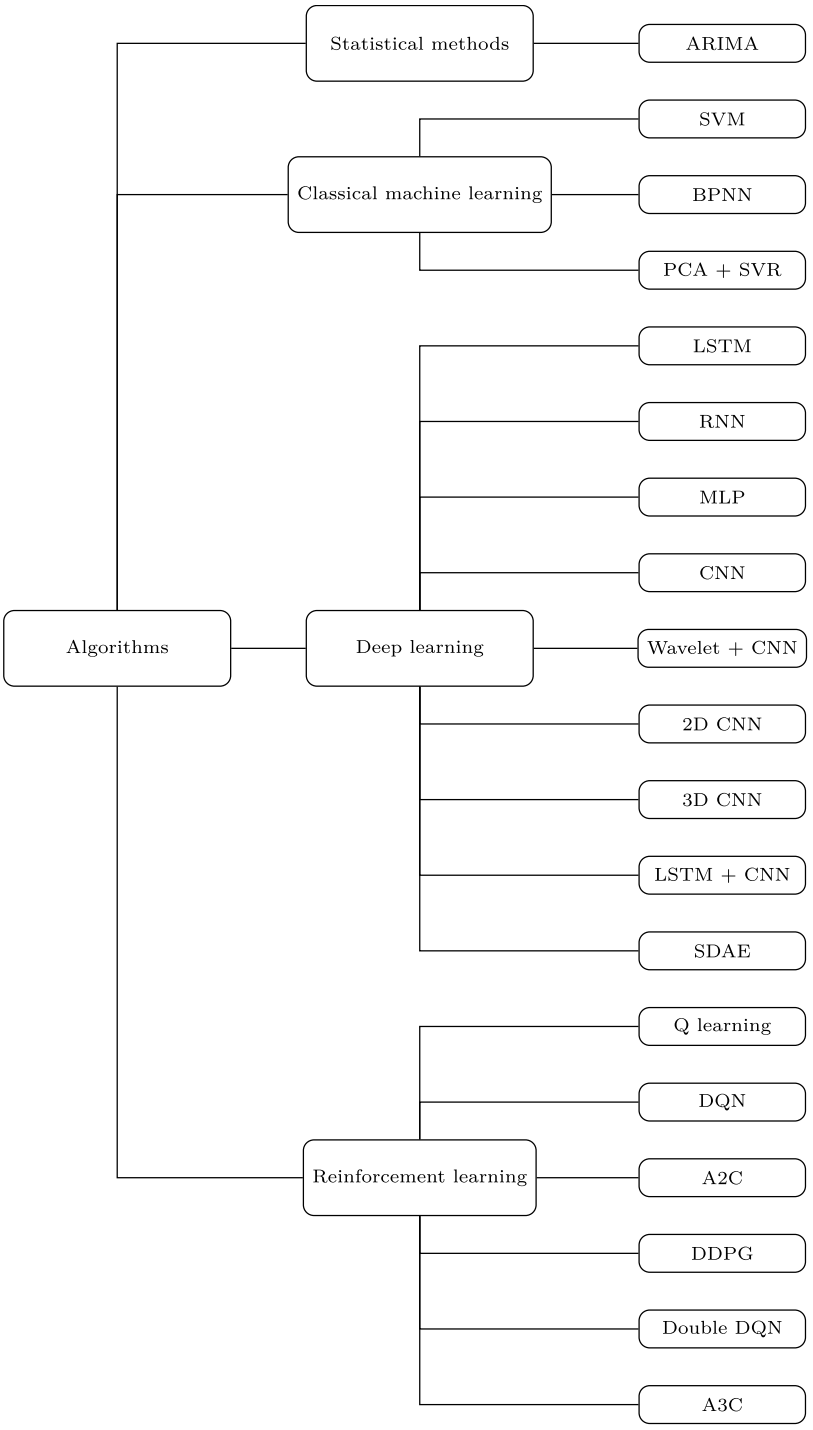


Fig. 1. The algorithms utilized in the recent articles.

of new coins into the cryptocurrency market. Moreover, Researchers in Yang, Liu, Zhong, and Walid (2020) also proposed a model to trade on a portfolio having 30 stocks. They compared different DRL techniques: Advantage Actor–Critic (A2C), Proximal Policy Optimization (PPO), Deep Deterministic Policy Gradient (DDPG), and an ensemble of them. They showed that the ensemble model outperforms all the individual models in terms of the Sharpe ratio, however, the PPO model can obtain the highest Return among the aforementioned models.

For the sake of better illustration, all the mentioned algorithms are summarized in Fig. 1.

3. DRL-based algorithmic trading

In this section, we introduce a new approach for trading in financial markets, based on continuous action space DRL. Table 1 summarizes the definitions of the variables used in the proposed method.

Table 1
List of variables.

Variables	Definitions	Variables	Definitions
t	Time step	$\tilde{a}$	The final value of target action
p	Close price	γ	Discount factor
x	The percentage change of close price	$N_{episode}$	The number of episodes
c	Cash	s	Current state
TC	Transaction cost	s'	Next state
n	The number of trading shares	α_{sig}	Significance level
R	Cumulative reward	α_1	The minimum value of action
r	Immediate reward	α_2	The maximum value of action
T	The length of trading period	Q	Q-value
a	Current action	θ	The weights of critic model
a'	Next action	θ'	The weights of critic target model
π	Policy value	ϕ	The weights of actor model
ϵ_1	Exploration noise	ϕ'	The weights of actor target model
σ	The standard deviation of exploration noise	σ_{init}	The initial value of σ
$\pm K$	The maximum/minimum value of noise limiter	σ_{end}	The final value of σ
ϵ_2	Policy noise	K_{init}	The initial value of K
$\tilde{\sigma}$	The standard deviation of policy noise	K_{end}	The final value of K
τ	Target update rate	$\tilde{\sigma}_{init}$	The initial value of $\tilde{\sigma}$
α	Learning rate	$\tilde{\sigma}_{end}$	The final value of $\tilde{\sigma}$
y	The final target Q-value	D	Decay parameter

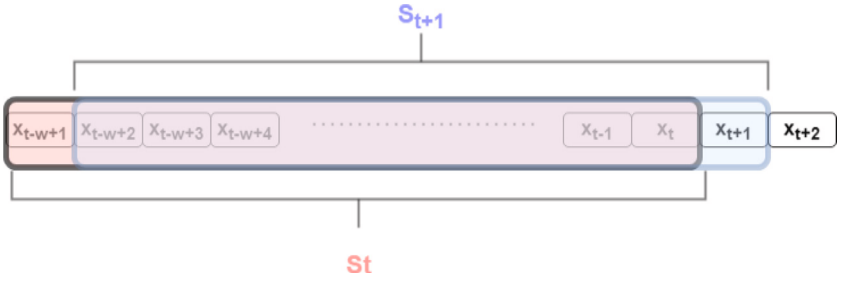


Fig. 2. The sliding window creating the states.

Each DRL problem generally consists of four components that must be properly defined: State, Environment, Action, and Reward. Hence, as follows, we present our proposed approach by accordingly defining those components. Then, the TD3, which is addressed to solve the DRL problem, is briefly explained. Finally, the proposed trading system is presented.

3.1. State space

Candlestick datasets, which include open, close, high, low price, and volume, are provided in several time steps, such as weekly, daily, and hourly steps. Traders extract a trading signal from the data (short-term and long-term) according to their aims. The daily resolution is the most common one among the traders.

Since the price data are not stationary, the percentage change of the close price is utilized to obtain stationary data:

$$x_t = 100 \times \frac{p_t - p_{t-1}}{p_{t-1}}, \tag{1}$$

where p_t and x_t are the close price and its percentage change at time t, respectively. As shown in Fig. 2, a sliding window is used to define the state space. At time t, the current state (red) is made up of the last w data samples (x_{t-i}; $i \in \{0, 1, \ldots, w-1\}$), where w is the length of the sliding window. The next state (blue) is formed when the window moves forward by one step.

3.2. Environment and action space

Traders make decisions based on the data provided in the financial markets. Long, Hold, and Short are considered to be the three actions (indicated by a_t) that the traders are allowed to take at each time t. By anticipating the price to rise, traders open a "Long" position, while a "Short" one shows that the trader expects the price to fall. It is obvious that a profit is made when the forecast is correct. The term "Hold" refers to remaining in the position and making no changes to the asset.

Trading involves opening a position at the start of the day and closing it at the end. The agent starts with initial cash (c_0) and follows a specified trading strategy. It picks actions in the interval $[-1, 1]$, where -1 and $+1$ represent the total cash is paid to open short and long position, respectively. In addition, if $a_t \in (-1, 1)$, the agent opens a position by spending $|a_t|\, c_{t-1}$ amount of the cash, while the remainder is unchanged. Obviously, positive and negative actions reveal long and short positions, respectively. Then, the number of trading shares is calculated according to $n_t = \frac{|a_t| c_{t-1}}{p_t}$. Finally, after closing the position, the amount of the cash is updated through:

$$c_t = c_{t-1} - |a_t|\, c_{t-1} + max\Big(n_t\,(p_{t+1} \pm p_t) + |a_t|\, c_{t-1} - \frac{n_t \times TC \times p_t}{100}, 0\Big), \tag{2}$$

where p_t is the close price at time t and TC is the transaction cost of each action. Furthermore, "+" denotes a long position, whereas "−" means a short one.

3.3. Reward

When an agent interacts with the environment, to achieve an optimal policy, it is guided by its received feedback (reward function). Simply, a positive reward encourages the agent to execute the action, but a negative one discourages the agent from taking the action. The Return function ($Return_t = \frac{c_t - c_{t-1}}{c_{t-1}}$) is one of the widely-used objective

functions in the RL approaches. By employing the logarithm of the Return function, we can control the final return of the trade ($\frac{c_T}{c_0}$), as illustrated in Eq. (4):

$$r_t = log\left(1 + Return_t\right) = log\left(\frac{c_t}{c_{t-1}}\right), \tag{3}$$

$$R_T = \sum_{i=1}^{T} r_i = \sum_{i=1}^{T} log(\frac{c_i}{c_{i-1}}) = log(\prod_{i=1}^{T} \frac{c_i}{c_{i-1}}) \; = \; log(\frac{c_T}{c_0}), \tag{4}$$

where r_t, R_T, and T are the immediate reward, cumulative reward, and the length of the trading period, respectively.

3.4. Twin-delayed deep deterministic (TD3)

The TD3 (Fujimoto, Hoof, & Meger, 2018) is a continuous action space approach that seems to fit into the problem at hand. It is an actor–critic technique where the actor network estimates the policy based on the states via a neural network and critic ones use both the action (chosen by the actor network) and the states in order to estimate the value function. To overcome the problem of overestimation, the TD3 includes two critic networks and two critic target networks, unlike the DDPG (Fujimoto et al., 2018; Silver et al., 2014). Nevertheless, the actor network of the TD3 is the same as the DDPG's, with one actor and one actor target network.

Turning into the details, in the training phase, several episodes with random actions are first run to explore the environment, and the transitions (s, a, r, s') are stored in a replay buffer. Following this, in each episode according to the current state (s), an action $(\pi_\varphi(s))$ is chosen based on the actor network. Then, exploration noise (ε_1) is added to the $\pi_\varphi(s)$:

$$a = \pi_\varphi(s) + \varepsilon_1, \tag{5}$$

$$\varepsilon_1 \sim \mathcal{N}(0, \sigma), \tag{6}$$

where a and σ are the noisy action and the standard deviation of the noise, respectively. Roughly speaking, since at the beginning of learning, the agent has less familiarity with the environment a significant amount is assigned to the exploration noise to help the agent to further explore the environment. This noise should also diminish exponentially as the number of episodes increases. As the agent becomes more familiar with the environment, the exploration chance decrease such that:

$$\sigma = \sigma_{end} + \left(\sigma_{init} - \sigma_{end}\right) exp\left(-\frac{N_{episode}}{D_\sigma}\right), \tag{7}$$

where σ_{init} and σ_{end} are respectively the initial and final values of the σ, $N_{episode}$ is the number of episodes, and D_σ is the decay parameter of the exponential function.

Similarly, according to a small batch of the transitions (s, a, r, s') (stored in the replay buffer), the actor target network determines the next actions $(a' = \pi_{\varphi'}(s'))$ from the next states (s'). Meanwhile, policy noise (ε_2: a Gaussian noise) is added to the next action. Successively, a Clip function is used to limit the next action and policy noise to two predefined values:

$$\widetilde{a} = Clip_{\alpha_1,\alpha_2}(\pi_{\varphi'}(s') + \varepsilon_2), \tag{8}$$

$$\varepsilon_2 \sim Clip_{-K,K}\left(\mathcal{N}\left(0, \widetilde{\sigma}\right)\right), \tag{9}$$

where $\widetilde{a}$ is the final value of the target action, α_1 and α_2 are the minimum and maximum possible values of the target action, $\widetilde{\sigma}$ is the standard deviation of policy noise, and $-K$ and K are the minimum and maximum possible values of the noise. According to the above argument for the exploration noise, an exponential decay approach is employed for both the standard deviation of the policy noise $(\widetilde{\sigma})$ and noise limiter (K):

$$\widetilde{\sigma} = \widetilde{\sigma}_{end} + \left(\widetilde{\sigma}_{init} - \widetilde{\sigma}_{end}\right) exp\left(-\frac{N_{episode}}{D_{\widetilde{\sigma}}}\right), \tag{10}$$

$$K = K_{end} + \left(K_{init} - K_{end}\right) exp\left(-\frac{N_{episode}}{D_K}\right). \tag{11}$$

Then, the final target Q-value (y), which is in turn used in the optimization of critic models, is calculated as follows:

$$y \; = \; r \; + \; \gamma \; \min_{i=1,2} Q_{\theta'_i}\left(s', \widetilde{a}\right), \tag{12}$$

where r is the immediate reward, $Q_{\theta'_i}(s', \widetilde{a})$ is the Q-value of the i^{th} critic target model, and γ is the discount factor. The next step involves updating the weights of critic models:

$$\theta_j \leftarrow argmin_{\theta_j} N_b^{-1} \sum_{i=1}^{N_b} (y_i - Q_{i,\theta_j}(s, a)), \tag{13}$$

where N_b is the batch number. Then, after every N_0 iterations, the weights of the actor network are updated according to:

$$\nabla_\varphi J(\varphi) = N_b^{-1} \sum_{i=1}^{N_b} \nabla_a Q_{\theta_1}(s, a)|_{\; a=\pi_\varphi(s)} \nabla_\varphi \pi_\varphi(s), \tag{14}$$

$$\varphi \leftarrow \varphi + \alpha \nabla_\varphi J(\varphi). \tag{15}$$

The gradient clipping technique is applied to Eq. (14) in order to stabilize the training stage. Finally, the target networks are updated every N_0 iterations using:

$$\theta'_i \leftarrow \tau\theta_i + (1 - \tau)\,\theta'_i, \tag{16}$$

$$\varphi'_i \leftarrow \tau\varphi_i + (1 - \tau)\,\varphi'_i, \tag{17}$$

where $0 \le \tau \le 1$ is a hyper-parameter to control incremental updating. For the sake of more clear demonstration, the TD3 algorithm (as proposed in Fujimoto et al., 2018) is encapsulated in Algorithm 1. Furthermore, the proposed trading algorithm is depicted in 3.

Algorithm 1 The TD3 pseudo-code:
the same as Algorithm 1 in Fujimoto et al. (2018).

```
Initialize critic networks θ1, θ2, and actor network φ.
Initialize target networks θ'1 ← θ1, θ'2 ← θ2, and φ' ← φ.
Initialize the experience replay buffer and store transitions via
random actions within several episodes in the replay buffer.
for t = 1 to N_episode * T do
    Select an action with exploration noise using Eq. (5).
    Update the standard deviation of exploration noise (σ) using Eq.
    (7).
    Store transitions in the replay buffer.
    Sample mini-batch of N_b transitions from the replay buffer.
    Calculate the next action (ã) using Eq. (8).
    Update the standard deviation of policy noise (σ̃) and noise limiter
    (K) using Eqs. (10) and (11), respectively.
    Calculate the final target Q-value (y) using Eq. (12).
    Update weights of critic networks using Eq. (13).
    if t mod N_0 then
        Update weights of actor network (φ) using Eqs. (14) and (15).
        Update weights of target networks (θ'_i and φ'_i) using Eqs. (16)
        and (17), respectively.
    end if
end for
```

4. Experimental results

The simulation results, reported in this section, have been done on Python 3.7, and the neural network models have been built by PyTorch 1.12.1. The dataset was acquired from the free version of the Yahoo Finance API in Python, containing the close price of Bitcoin (BTC) from 2014-10-15 to 2020-01-01 and Amazon (AMZN) from 2010-01-01 to 2021-06-11 with the daily resolution. Each experiment has been divided into three phases: training, validation, and testing, in which we

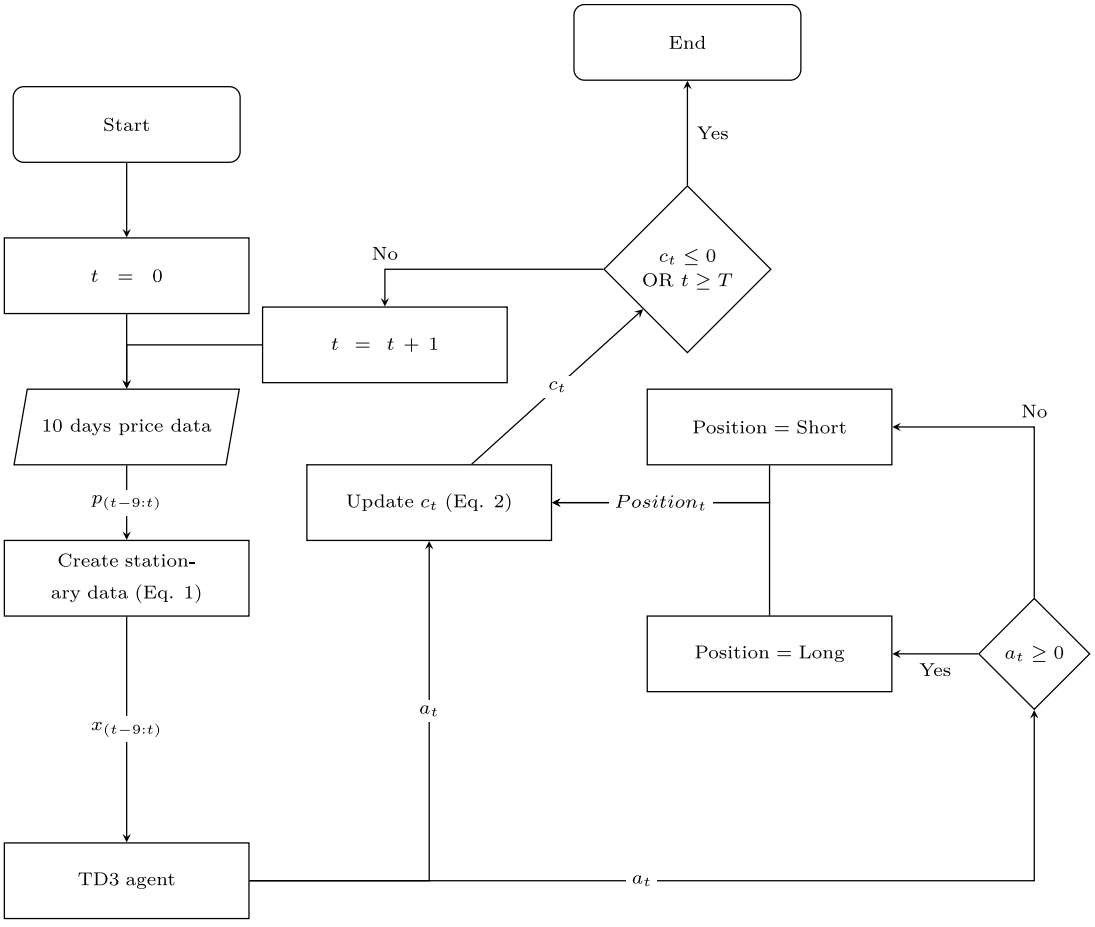


Fig. 3. The proposed trading algorithm.

divided the dataset into 80%, 10%, and 10% portions, respectively. The training phase has been done in 50 episodes.

Eleven models and two metrics are briefly introduced in order to be used in the evaluation of the algorithms. The TD3 is first evaluated in the stock market using the AMZN. In this scenario, the TD3 is compared to two discrete algorithms in order to demonstrate the superiority of our proposed continuous algorithm. Finally, this algorithm is examined in the cryptocurrency market (Bitcoin). In all cases, the agent started with 100,000$ as its initial cash.

4.1. Baseline models and metrics

Table 2 provides the descriptions of two random models (Random-C, Random-D), four deterministic models (Buy and Hold, Sell and Hold, Long, and Short), two technical-indicator-based models (MRMA[2] and TFMA[3]), one DRL-based model (TDQN) (Théate & Ernst, 2021), and two supervised learning models (LSTM and SVM) (Shi et al., 2021). Moreover, the models are evaluated using two standard metrics (Return and Sharpe ratio) as described in Table 3.

Table 2
Baseline model descriptions.

Model	Approach	Description
Random-C	Random	Continuous Uniform Distribution of $[-1, 1]$
Random-D	Random	Discrete Uniform Distribution of $\{-1, 1\}$
Buy and Hold (BH)	Deterministic	Opens a long position and holds it till the end of trading period
Sell and Hold (SH)	Deterministic	Opens a short position and holds it till the end of trading period
Long	Deterministic	Opens a long position and closes it after a 1-day interval
Short	Deterministic	Opens a short position and closes it after a 1-day interval
MRMA	Technical Indicator	Assumes that the price goes back to its average
TFMA	Technical Indicator	Follows the previous trend of the price
TDQN	DRL	DQN
LSTM	Supervised Learning	Binary Classification
SVM	Supervised Learning	Binary Classification

4.2. Amazon

In this subsection, we report the TD3 performance in the AMZN market. As shown in Fig. 4, continuous distribution of the actions (between +1 and −1) seems promising to capture its true nature. In order to make a fair comparison between continuous (TD3) and discrete action space approaches, Sign (Eq. (18)) and D3 (Eq. (19)) are defined as follows:

$$f_{Sign}(a) = \begin{cases} -1; & a <= 0 \\ +1; & a > 0 \end{cases}, \tag{18}$$

[2] Mean Reversion Moving Average.
[3] Trend Following Moving Average.

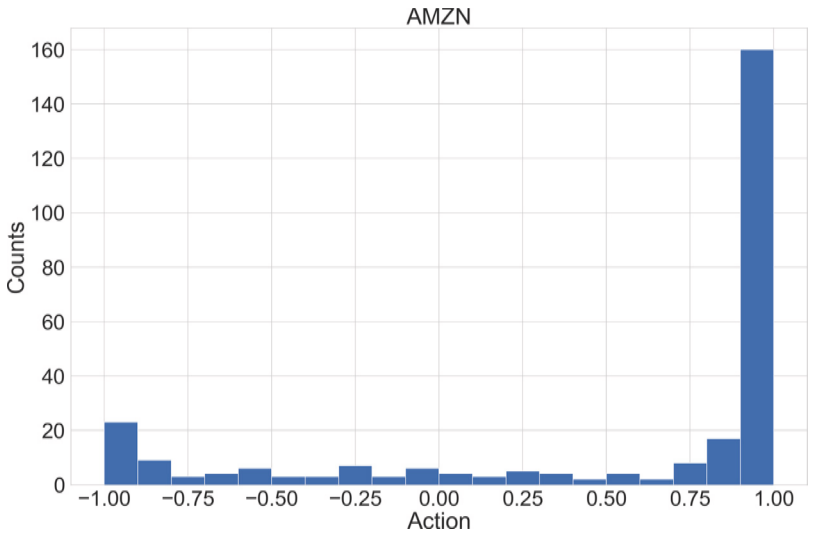


Fig. 4. The histogram of Amazon market actions in the TD3 algorithm.

Table 3
Performance metrics.

Metric	Formula
Return (%)	$100 \times \frac{\text{Final cash - Initial cash}}{\text{Initial cash}}$
Sharpe ratio	$\sqrt{\text{The number of trading days in a year}} \times \frac{\text{The average of Returns}}{\text{The standard deviation of Returns}}$

Table 4
T-test results.

Algorithms	TD3 and Sign		TD3 and D3	
	T_0	P-value	T_0	P-value
Return	−2.84	0.004	−1.65	0.053
Sharpe ratio	−4.44	0.00004	−3.97	0.00015

$$f_{D3}(a) = \begin{cases} -1; & a <= -\frac{1}{3} \\ 0; & -\frac{1}{3} < a <= \frac{1}{3} \\ +1; & a > \frac{1}{3} \end{cases}, \tag{19}$$

where a, $f_{Sign}(a)$, and $f_{D3}(a)$ are the actions of the TD3, Sign, and D3 algorithms, respectively. As illustrated in Fig. 5, by repeating the study over 40 experiments and comparing the results in terms of median values of Return and Sharpe ratio, the continuous action space algorithm outperforms the discrete ones. It should be noted that, using the TD3, the Return improves in 75% and 60% of cases rather than using the Sign and the D3, respectively. These values for the Sharpe ratio were 80% and 70%, respectively. It is worth mentioning that although, in some cases, the Sign algorithm may achieve a higher Return, this increase is accompanied by a decline in the Sharpe ratio value. In other words, the flexibility of the TD3 in controlling over the amount of invested money (especially in high-risk situations) has resulted in a greater chance of realizing a better Sharpe ratio than two other algorithms.

For Return and Sharpe ratio, the results of the mean comparison test ($\alpha_{sig} = 0.01$) have been provided in Table 4. We deduce that the null hypothesis is rejected in all cases with the confidence level of 99%, except for the Return test between the D3 and TD3, where it can be rejected with the confidence level of 94%. Therefore, the TD3 outperforms Sign and D3 in average.

Table 5 reports the simulation results of the performance of some of the commonly-used algorithms in the AMZN market. The BH outperforms the other algorithms in terms of Return and Sharpe ratio. Moreover, the TD3 and TDQN are the second and third best algorithms, which implies that the DRL can be a useful approach in such a case. This also indicates that a continuous model performs better than a discrete one.

Table 5
Algorithm comparison in AMZN market.

Algorithm	Return (%)	Sharpe ratio
TD3	22.0	0.80
TDQN	8.1	0.38
BH	**35.4**	**1.05**
SH	−35.5	−0.48
MRMA	−20.0	−0.42
TFMA	−10.9	−0.20
Long	7.4	0.37
Short	−51.6	−2.02
Random-C	−9.5	−0.39
Random-D	−56.4	−2.34
LSTM	7.4	0.37
SVM	−27.8	−0.82

4.3. Bitcoin

The histogram of the TD3 actions in the BTC scenario is illustrated in Fig. 6. Almost all the actions are either +1 or −1, demonstrating that the distribution of the actions is almost discrete; thus, there is no meaningful difference between continuous and discretized models. In such a case, the strategy is barely distinguishable from a discrete algorithm which suggests no risk control over the trading procedure. This can be due to the higher volatility of the cryptocurrency market than the stock market so the agent leads to spend more amount of cash for opening a position.

5. Conclusion and future works

In recent years, the use of the machine learning approach in algorithmic trading has become increasingly ubiquitous. Therefore, this research aimed to present an experiment investigating this issue in both the stock and cryptocurrency markets. Since both position and the number of trading shares are crucial in trading, a continuous action

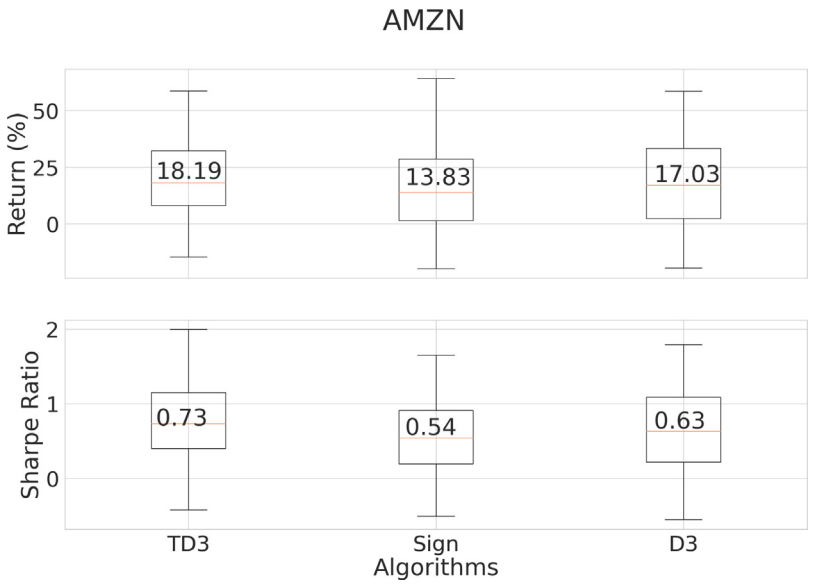


Fig. 5. Box-plot compares the TD3, Sign, and D3.

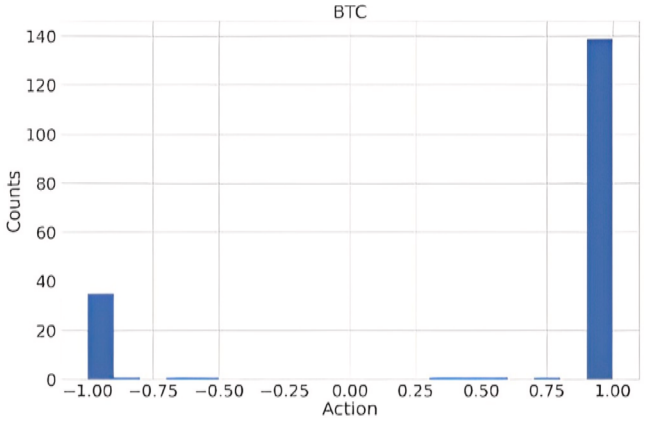


Fig. 6. The histogram of Bitcoin market actions in the TD3 algorithm.

space DRL (TD3) was employed. It was shown that in the AMZN market, the proposed technique has the advantage of reducing the risk associated with trading by leaving a portion of a trader's cash out. It helps reduce losses and protect the trader from losing all their money. Furthermore, there is no limit to the number of trading shares in our model since it is determined using the available cash. The results indicated that our proposed method outperforms the baseline ones in most cases. However, our strategy may not befit all types of traders, especially those planning high-risk trades.

Nevertheless, there are some certain limitations to this method, which should be addressed in the future research. To begin with, it is valuable to compare other continuous action space DRL algorithms using our proposed method. In addition, by modifying the reward function to be more realistic, one can improve the agent's performance. Furthermore, the performance can be improved by using an ensemble of techniques from diverse fields. Finally, a modification can be made in order to allow the trader to choose the degree of trading risk. These adjustments are introduced to make the method closer to how real traders develop a profitable strategy.

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

The used dataset is freely accessible as mentioned in the manuscript.

References

Alonso-Monsalve, S., Suárez-Cetrulo, A. L., Cervantes, A., & Quintana, D. (2020). Convolution on neural networks for high-frequency trend prediction of cryptocurrency exchange rates using technical indicators. *Expert Systems with Applications*, *149*, Article 113250.

Ariyo, A. A., Adewumi, A. O., & Ayo, C. K. (2014). Stock price prediction using the ARIMA model. In *2014 UKSim-AMSS 16th international conference on computer modelling and simulation* (pp. 106–112). IEEE.

Betancourt, C., & Chen, W.-H. (2021). Deep reinforcement learning for portfolio management of markets with a dynamic number of assets. *Expert Systems with Applications*, *164*, Article 114002.

Chakole, J. B., Kolhe, M. S., Mahapurush, G. D., Yadav, A., & Kurhekar, M. P. (2021). A Q-learning agent for automated trading in equity stock markets. *Expert Systems with Applications*, *163*, Article 113761.

Dang, Q.-V. (2019). Reinforcement learning in stock trading. In *International conference on computer science, applied mathematics and applications* (pp. 311–322). Springer.

Di Persio, L., & Honchar, O. (2016). Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International Journal of Circuits, Systems and Signal Processing*, *10*(2016), 403–413.

Drakopoulou, V. (2016). A review of fundamental and technical stock analysis techniques. *Journal of Stock & Forex Trading*, *5*.

Dutta, A., Kumar, S., & Basu, M. (2020). A gated recurrent unit approach to bitcoin price prediction. *Journal of Risk and Financial Management*, *13*(2), 23.

Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., et al. (2019). A guide to deep learning in healthcare. *Nature Medicine*, *25*(1), 24–29.

Faes, L., Wagner, S. K., Fu, D. J., Liu, X., Korot, E., Ledsam, J. R., et al. (2019). Automated deep learning design for medical image classification by health-care professionals with no coding experience: A feasibility study. *The Lancet Digital Health*, *1*(5), e232–e242.

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing function approximation error in actor-critic methods. In *International conference on machine learning* (pp. 1587–1596). PMLR.

Hirchoua, B., Ouhbi, B., & Frikh, B. (2021). Deep reinforcement learning based trading agents: Risk curiosity driven learning for financial rules-based policy. *Expert Systems with Applications*, *170*, Article 114553.

Hoseinzade, E., & Haratizadeh, S. (2019). CNNpred: CNN-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, *129*, 273–285.

Jangmin, O., Lee, J., Lee, J. W., & Zhang, B.-T. (2006). Adaptive stock trading with dynamic asset allocation using reinforcement learning. *Information Sciences*, *176*(15), 2121–2147.

Jeong, G., & 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.

Khan, A. I., & Al-Habsi, S. (2020). Machine learning in computer vision. *Procedia Computer Science*, *167*, 1444–1451.

Kirkpatrick, C., & Dahlquist, J. (2008). Analysis: The complete resource for financial market technicians. Prentice Hall/Financial Times, Upper Saddle River, NJ. In *Encyclopedia of alternative investments* (p. 413). CRC Press.

Lee, J. W. (2001). Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE international symposium on industrial electronics proceedings (Cat. No. 01TH8570),vol. 1* (pp. 690–695). IEEE.

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

Liu, M., Li, G., Li, J., Zhu, X., & Yao, Y. (2021). Forecasting the price of bitcoin using deep learning. *Finance Research Letters*, *40*, Article 101755.

Liu, Q., Tao, Z., Tse, Y., & Wang, C. (2022). Stock market prediction with deep learning: The case of China. *Finance Research Letters*, *46*, Article 102209.

McNally, S., Roche, J., & Caton, S. (2018). Predicting the price of bitcoin using machine learning. In *2018 26th euromicro international conference on parallel, distributed and network-based processing* (pp. 339–343). IEEE.

Murphy, J. J. (1999). *Technical analysis of the financial markets: a comprehensive guide to trading methods and applications* (2nd). (pp. 1–2). Penguin.

Nair, M. M., Kumari, S., Tyagi, A. K., & Sravanthi, K. (2021). Deep learning for medical image recognition: open issues and a way to forward. In *Proceedings of the second international conference on information management and machine intelligence* (pp. 349–365). Springer.

Pathak, A. R., Pandey, M., & Rautaray, S. (2018). Application of deep learning for object detection. *Procedia Computer Science*, *132*, 1706–1717. http://dx.doi.org/10.1016/j.procs.2018.05.144, URL https://www.sciencedirect.com/science/article/pii/S1877050918308767, International Conference on Computational Intelligence and Data Science.

Phaladisailoed, T., & Numnonda, T. (2018). Machine learning models comparison for bitcoin price prediction. In *2018 10th international conference on information technology and electrical engineering* (pp. 506–511). IEEE.

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.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *International conference on machine learning* (pp. 387–395). PMLR.

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

Traore, B. B., Kamsu-Foguem, B., & Tangara, F. (2018). Deep convolution neural network for image recognition. *Ecological Informatics*, *48*, 257–268.

Yang, H., Liu, X.-Y., Zhong, S., & Walid, A. (2020). Deep reinforcement learning for automated stock trading: An ensemble strategy. In *Proceedings of the first ACM international conference on AI in finance* (pp. 1–8).