IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 February 2023, accepted 13 March 2023, date of publication 20 March 2023, date of current version 27 March 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3259108

RESEARCH ARTICLE

A Mean-VaR Based Deep Reinforcement Learning Framework for Practical Algorithmic Trading

BOYI JIN
School of Insurance, Shandong University of Finance and Economic, Jinan 250014, China
e-mail: 20110983@sdufe.edu.cn

ABSTRACT It is difficult to automatically produce trading signals based on previous transaction data and the financial status of assets because of the significant noise and unpredictability of capital markets. This paper proposes an innovative algorithm to solve the optimal portfolio problem in stock market trading activities. Our novel portfolio trading strategy utilizes three features to outperform other benchmark strategies in a real-market environment. First, we propose a mean-VaR portfolio optimization model, the solution of which is based on the actor-critic architecture. Unlike the existing literature that learns the expectation of cumulative returns, the critic module learns the cumulative returns distribution by quantile regression, and the actor module outputs the optimal portfolio weight by maximizing the objective function of the optimization model. Secondly, we use a linear transformation function to realize short selling to ensure investors have profit opportunities in the bear market. Third, A multi-process method, called Ape-x, was used to accelerate the speed of deep reinforcement learning training. To validate our proposed approach, we conduct backtesting for two representative portfolios and observe that the proposed model in this work is superior to the benchmark strategies.

INDEX TERMS Deep reinforcement learning, algorithmic trading, actor-critic architecture, trading strategy.

I. INTRODUCTION

Algorithmic trading, which has been widely used in the financial market, is a technique that uses a computer program to automate buying and selling of stocks, options, futures, and cryptocurrency. Institutional investors such as pension, mutual, and hedge funds usually use algorithmic trading to find the most favourable execution price, reduce the impact of severe market fluctuations and improve execution efficiency. Algorithmic trading has a far-reaching effect on the overall efficiency and microstructure of the capital market. Therefore, asset pricing, portfolio investment, and risk measurement may undergo revolutionary changes.

Classical algorithmic strategies includes arrival price strategy [1], [2]; volume-weighted average price strategy [3], [4]; time-weighted average price strategy [5], implementation shortfall strategy [6], [7]; guerrilla strategy [8], among others. As artificial intelligence (AI) advances quickly, more academics have begun utilizing deep reinforcement learning to implement algorithmic trading. Through its excellent feature representation ability of the deep neural network to fit the state, action, value, and other functions, deep reinforcement learning has demonstrated strong learning ability and decision-making ability in autonomous driving [9], [10], job scheduling [11], [12], and game playing [13], [14]. In addition to being more advanced than human experts in particular domains, it also has the potential to achieve artificial general intelligence [15]. As a result, researchers started using deep reinforcement learning to handle portfolio optimization issues. The actor-only method [16], [17], [18], the critic-only method [19], [20], [21], and the actor-critic method [22], [23], [24] are now being used by researchers to examine financial trading.

According to the modern portfolio theory [25], investors need to allocate their assets in different securities markets and products at appropriate proportions to maximize the expected return in a certain period while controlling the risk at a certain level [26]. Hence, balancing risk and return

The associate editor coordinating the review of this manuscript and approving it for publication was Emanuele Lattanzi.

is very important in portfolio optimization decision [27]. Nevertheless, classical reinforcement learning algorithms, such as Deep Q-Network(DQN), Policy Gradient(PG), and Asynchronous Advantage Actor-Critic(A3C), can only learn the expectation of a portfolio's cumulative return by training the value function or action-value function. These algorithms ignore other important information, such as volatility and tail risk, and are only suitable for investors as a neutral risk type. Although [22], [24], [28] consider the risk-adjusted return in the reward function, these methods can not measure the cumulative risk under multi-period conditions. So far, no research has focused on cumulative risk control in multi-period investment strategies.

In this paper, we propose a novel algorithm that balances the risk and return in long-term investment horizons. Specifically, we first construct a mean value-at-risk (MVaR) optimization model, the solution of which is based on an actor-critic architecture. Both the actor and critic modules are parameterized with neural networks. Unlike previous studies that only focus on the expectation of a portfolio's cumulative returns, in the paper, the critic module is to learn the distribution of cumulative returns by the distributional reinforcement learning algorithm proposed by [29]. The advantage of this method is that it can measure the tail risk. Then, the actor module outputs the optimal weights by maximizing the abovementioned MVaR optimization model. Experimental result shows that the proposed algorithm yields higher profit and undertakes lower risk than the four benchmark strategies.

Additionally, short selling is allowed in lots of stock markets. Short selling can provide investors with profit opportunities in the bear market, help reduce the stock market's speculation and volatility, and increase the capital market's liquidity. Short selling has been considered in many deep reinforcement learning algorithms. For instance, [30] proposed an improved deep reinforcement learning (DRL) method for automated stock trading. Experimental results show that allowing short selling can improve the algorithm's overall performance. [31] use the classic Q-learning algorithm, which allows short selling, to evaluate the performance of cumulative profits by maximizing different forms of value functions. Nevertheless, these studies focus on single-asset trading. Most traders generally hold multiple securities, and few studies have considered the short-selling problem in this typical case. In this paper, we use a linear function to transform the output of the softmax function, which was used by [24], into the final portfolio weight. This transformation can not only realize the self-financing constraint but also control the scale of short selling. Furthermore, the maximum short-selling amount, which can also be considered the maximum loss that investors may suffer, can be determined by adjusting related parameters of the linear transformation function mentioned above. We further compare the performance of algorithms with and without short selling constraints. Experimental results show that short selling can increase overall performance in some cases.

Finally, reinforcement learning training is mainly divided into two parts: the sample collection process, that is, the agent interacts with the environment to generate training samples, The other is the training process that uses the collected training samples to update the strategy. These two parts are coupled in classical algorithms such as PG [16], [18] and DQN [32], [33], [34], making the training process very time-consuming. However, in the securities market, profit opportunities are fleeting. Therefore, it is worthwhile to research how to increase the reinforcement learning algorithm's training speed. In this paper, we leverage a multi-process algorithm, called the Ape-x algorithm proposed by [35], to speed up the training. The Ape-x algorithm separates data collection from strategy learning, uses multiple parallel agents to collect experience data, shares a large experience data buffer, and sends it to learners for learning. The original Ape-X, which was based on DQN and Deep Deterministic Policy Gradient(DDPG), was utilized in the feedback flow separation control system [53], StarCraft games [54] and controlling vehicles for autonomous driving [55]. The following two characteristics are where this paper most clearly improves: we first connect the Ape-x with a distributed reinforcement learning framework for off-policy learning. The portfolio optimization issue is then solved using this architecture for the first time.

The rest of this paper is organized as follows. In Section II, we review the related literature and present the differences between our study and previous studies. Section III provides an overview of the preliminaries and the background of financial trading problems, and Section IV introduces our proposed R3L algorithm. In Section V, we detail the setups of our experiments and results. Finally, Section VI presents the conclusions and directions of future work.

II. RELATED WORKS

In this section, we review the literature works regarding deep reinforcement learning in financial trading. As mentioned in the introduction, the algorithms used in financial trading mainly include the critic-only method, the actor-only method, and the actor-critic method. Then, we will elaborate on the application of the three methods.

A. THE CRITIC-ONLY METHOD

The critic-only method is also called the value-based method. DQN is a well-known value-based deep reinforcement learning algorithm, the core of which is to use a neural network to replace the q-label, i. e. the action value function. The input of the neural network is state information, and the output is the value of each action. Therefore, The DQN algorithm can be used to solve the problem of continuous state space and discrete action space but can not solve the problem of continuous action space.

DQN and its improved algorithms have received extensive attention in the research of financial trading. [33] adopted a DRQN-based algorithm, which was applied to the foreign

exchange market from 2012 to 2017, and developed a motion enhancement technology to mitigate the lack of exploration in Q-learning. Experimental results showed that the annual return of some currencies was 60%, with an average of about 10%. [28] proposed a novel deep Q-learning portfolio management strategy. The framework consists of two main components: a group of local agents responsible for trading a single asset and a global agent that rewards each local agent based on the same reward function. The results showed that the proposed approach was a promising method for dynamic portfolio optimization. The deep Q-learning network proposed by [37] is based on an improved Deep Neural Networks(DNN) structure consisting of two branches, one of which learned action values while the other learned the number of shares to take to maximize the objective function. [34] discretized the action space by fixing the trading amount, in which traders can buy or sell a specific asset for a fixed amount. Short selling is not allowed, and a mapping function is designed to transform the infeasible action into feasible action.

Since accurate prediction of stock prices is beneficial for investment decisions, researchers use deep learning to solve the problem of time series prediction of stock returns. [20] deploys Long-Short Term Memory(LSTM) networks for predicting out-of-sample directional movements for the constituent stocks of the S&P 500 from 1992 until 2015. [56] uses self-organizing maps (SOMs) for clustering the historical prices and produces a low-dimensional discretized representation of the input space. [57] proposes a hierarchical attention network for stock prediction (HATS) which uses relational data for stock market prediction.

B. ACTOR-ONLY METHOD

In the actor-only method, the action taken by the agent can be learned directly by a neural network. The advantage of this method is that it makes it available for continuous action space. [16] constructed a recursive deep neural network for environmental perception and recursive decision-making. Then, deep learning is used for feature extraction and combined with fuzzy learning to reduce the uncertainty of input data. Another novelty of this paper is a variant of Backpropagation through time(BTT) to handle the vanishing gradient problem. [17] used PG approach for financial trading. Its main contribution is to analyze the advantages of the LSTM network structure over the fully connected network and to compare the impact of some combination of technical indicators on revenue. [18] used the PG algorithm to study the trading problem of financial and commodity futures. The innovation is that they used candlesticks as a generalization of price movements over some time, and the candlesticks are decomposed into different subparts by three clustering methods(i.e., K-means, fuzzy c-means clustering method, and online clustering method). They proved that the above-preprocessing data methods could improve the algorithm's overall representation.

The drawback of the Actor-only method is that, due to the policy strategy, the number of interactions between agents and the environment is significantly increased compared with a value-based method. As a result, the training is very time-consuming.

C. ACTOR-CRITIC METHOD

The actor-critic method contains two parts: an actor, which selects action based on probability or certainty, and a critic, which evaluates the score of the action taken by the actor. The actor-critic method combines the advantages of the value-based and policy-based methods, which can not only deal with continuous and discrete problems but also carry out the one-step update to improve learning efficiency. [22] exploited Gated Recurrent Unit(GRU) to extract financial features. They proposed two algorithms: the Gated Deep Q-learning trading strategy(GDQN) based on a critic-only method and the Gated Deterministic Policy Gradient trading strategy(GDPG) based on the Actor-Critic method. Experimental results show that GDQN and GDPG outperformed the Turtle trading strategy [38] and DRL trading strategy [16], and the performance of GDPG is more stable than the GDQN in the ever-evolving stock market. [23] adopted three versions of the RL algorithm based on Deep Deterministic Policy Gradient (DDPG), Proximal Policy Optimization (PPO), and Policy Gradient (PG) for portfolio management. A so-called Adversarial Training was proposed to reach a robust result and to consider possible risks in optimizing the portfolio more carefully. Using Proximal Policy Optimization, [24] proposed a unique reinforcement learning technique for portfolio optimization. One of its key features is that the proposed model's asset count is dynamic. According to experiments conducted on a bitcoin market, the proposed method outperformed three state-of-the-art algorithms presented by [20], [39], and [40]. Reference [41] compared the performance of deep double Q-learning and PPO to several benchmark execution policies and found that PPO realizes the lowest total implementation short-fall across exchanges and currency pairs.

To summarise, these studies used various RL algorithms for financial trading problems in different settings. The existing literature also has the following problems. First, most literature determines the optimal action(portfolio decision) based on maximizing the expectation of the cumulative return and pays less attention to the risk. Although [22], [24], [28] considers the risk factor in the reward function, these authors can not measure the risk of cumulative return. Second, short selling is allowed in many strategies [16], [37], [50], but these papers considered trading for only one asset; investors often hold multiple assets to diversify risk. Although [34], [39] consider trading for multiple assets, short selling was not allowed, which means that investors will not be able to make profits in the bear market. Third, the interaction between the agent and the environment is very time-consuming in the training process. Most existing literature does not consider how to improve the training speed. Thus, we contribute to the

literature by deriving a portfolio trading strategy with no such issues mentioned above.

III. PRELIMINARIES

In this section, we introduce some preliminaries about MDP and elaborate state space, action space, and rewards function of the proposed algorithm.

A. MARKOV DECISION PROCESS

Markov decision process (MDP) is a mathematical model for sequential decision problems, which is used to simulate the random strategies and rewards that agents can realize in the environment where the system state has Markov properties. Portfolio optimization is a typical sequential decision problem. Investors dynamically adjust portfolio weights according to market information. Thus, we can consider the portfolio allocation problems as a MDP. In the stock market, it is difficult for agents to observe complete state variables, and some information that affects the stock price can not be observed, which means that the Markov characteristic is rarely established. The partially observable Markov decision process(POMDP) is an alternative method. In this method, the agent only captures part of the information of the current state environment. A four-tuple$(S, A, P_a(s, s'), R_a(s, s'))$ can describe the POMDP.

where:

- S is a finite set of the observable state.
- A is a finite set of actions (and As is the finite set of actions available from state s).
- $R_a(s, s')$ is reward function.
- $P_a(s, s') = Pr(s_{t+1} = s' \| s_t = s, a_t = a)$ is the state transition probability.

The interaction between an agent and financial environment will produce a trajectory $\tau = \{s_0, a_0, R_0, s_1, a_1, R_1, \ldots\}$. G_t is the discounted cumulative reward, which the agent can obtain at time t expressed as follows:

$$G_t = \sum_{i=t}^{T} \gamma^{i-t} R(s_i, s_{i+1}) \tag{1}$$

where $\gamma \in [0, 1]$ is the discounted rate.

To learn the optimal strategy, we use the value function. There are two types of value functions in reinforcement learning: state value function, denoted by $V(s)$, and action-value function, denoted by $Q(s, a)$. The state value function, shown in Eq.(2), represents the expectation of cumulative rewards from a certain state s.

$$V(s) = E[G_t \| s_t = s] \tag{2}$$

The action value function, given in Eq.(3), is the expected return obtained after the agent executes an action in the current state s.

$$Q(s, a) = E[G_t \| s_t = s, a_t = a] \tag{3}$$

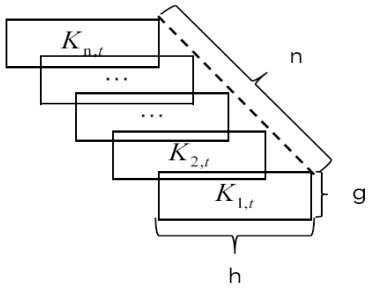

FIGURE 1. Market feature tensor.

B. STATE SPACE

The limited observability of the market environment is a significant and challenging issue in algorithmic trading. Investors might have access to a broader range of information than they have in the current complex market environment. It is crucial to know how to handle this restricted amount of information. The state at the period t, denoted by s_t, consists of three types of variables: historical data for the assets(K_t), portfolio weights(w_t) and trading time step(t). The historical data of the selected portfolio consists of raw data and technical indicators. The raw data includes the price open, close, high, low, and volume(OCHLV). Technical indicators consist of a list of candlestick patterns, including bearish, bullish, significant, hammer, inverse hammer, bullish engulfing, piercing line, morning star, bullish harami, hanging man, shooting star, bearish engulfing, evening star, three black crows, dark cloud cover, bearish harami, doji, spinning top(see [43] for details). These technical indicators are heuristic or mathematical calculations based on the price, volume and other data of stocks. All historical data can be represented by a tensor with dimension $n \times g \times h$ illustrated in Fig.1. N is the number of risky assets, g is the number of features per asset, and h denotes a window size which is the number of latest feature values the agent can observe. Portfolio weight, also called portfolio vector, is the percentage of a total portfolio represented by a single asset. The rationality of adding a time index to the critic network is that it reflects the time value of money, which refers to the fact that a dollar in hand is worth more than a dollar promised in future. It means a cash flow (or cash flows) has different values at different time points.

$$\begin{aligned} s_t &= (K_t, w_t, t) \\ w_t &= (w_{1,t}, w_{2,t}, \ldots w_{n,t}) \\ K_t &= (K_{1,t}, K_{2,t}, \ldots K_{n,t}) \\ K_{i,t} &= (K_{i,t,1}, \ldots K_{i,t,g}), \quad i = 1, \ldots n \\ K_{i,t,j} &= (K_{i,t,j,t-h+1}, \ldots K_{i,t,j,t}), \quad j = 1, \ldots g \end{aligned} \tag{4}$$

where $K_{i,t,j,t-k} (k = 0, \ldots h-1)$ represents the value of the jth feature lag period k of asset i in period t.

C. ACTION SPACE

At each time step, the agent must redetermine the optimal portfolio weight according to the updated state at the end of

each period. The agent executes a trading action according to the actor network. We get the actor network through the following steps. First, We construct a neural network $H_\varphi(s)$ whose output is an n-dimensional vector. Since the deterministic strategies of DDPG lack exploration, we need to add noise artificially. [44] uses Ornstein Uhlenbeck process(OU) as action noise; however, [45] found that noise drawn from the OU process offered no performance benefits. In this paper, following [46], we add Gaussian noise, denoted by ϵ, to output of $H_\varphi(s)$, which is given by Eq.5.

$$\begin{aligned} A &= H_\varphi(s) + \epsilon \\ \epsilon &\sim clip(N(0, \sigma), -c, c) \end{aligned} \tag{5}$$

where $A = \{A_1, A_2, \ldots A_n\}$ is a n-dimensional vector. Second, to meet self-financing conditions, we use the softmax layer to transform A into a new variable vector $SA = \{SA_1, SA_2, \ldots SA_n\}$ that represents portfolio weight. This transformation is given in Eq.(6).

$$SA_i = \frac{exp^{A_i}}{\sum_{i=1}^{n} exp^{A_i}}, \qquad i = 1, 2, \ldots n \tag{6}$$

The drawback of this transformation is that the weight of each asset in a portfolio is greater than zero, which makes short selling impossible. Short selling occurs when an investor borrows a security and sells it on the open market, planning to repurchase it later for less money. It allows investors to benefit from a bear market and use capital proceeds to overweight the portfolio's long-only component. Different from the existing literature, we realize short selling of assets through the following transformation(Eq. (7)).

$$w_i = SA_i \times \delta - \frac{\delta - 1}{n}, \qquad i = 1, 2, \ldots n \tag{7}$$

where $w_i (i = 1, 2, \cdots n)$ represents the proportion of asset i after transformation and δ(called Delta) represents adjustment parameter. It can be observed that this linear transformation can not only meet the self-financing constraints of a portfolio but also realize short selling. In particular, $\delta = 1$ means short selling is not allowed, and $\delta = 0$ indicates that each asset's portfolio weight equals $1/n$. Furthermore, in this paper, our portfolio includes four risky assets and a risk-free asset(n=5); if the adjustment parameter is set to 3($\delta = 3$), the portfolio weight of a single asset ranges from -40% to 260%, which means the maximum proportion of short selling in total assets is 160%.

D. REWARDS

Multiple portfolio problems require investors to make continuous dynamic decisions, and each decision will produce an immediate reward denoted by $reward_t$. The reward reflects the performance of an agent's action. Let $asset_t$ denote portfolio value at the end of period t. The reward is defined as portfolio return in period t, computed as the following equation:

$$reward_t = \frac{asset_t}{asset_{t-1}} - 1 \tag{8}$$

Suppose we do not take into account transaction cost, portfolio value at the end of period t equals portfolio value at the end of period $t-1$ plus the value-added part in period t, which can be expressed as follows:

$$asset_t = asset_{t-1} + asset_{t-1} \times \sum_{i=1}^{n} w_{i,t-1} \times R_{i,t} \tag{9}$$

According to the Eq.(8) and Eq.(9), $reward_t$ can be rewritten as follows:

$$reward_t = \sum_{i=1}^{n} w_{i,t-1} R_{i,t} \tag{10}$$

If we consider transaction cost, $asset_t$ equals $asset_{t-1}$ plus the value-added part in period t and minus transaction costs, which can be computed as follows:

$$\begin{aligned} asset_t = asset_{t-1} + asset_{t-1} \times \sum_{i=1}^{n} w_{i,t-1} \times R_{i,t} - \\ c1 \times \sum_{i=1}^{n} sell_{i,t} \times sellin_{i,t} \times price_{i,t} - \\ c2 \times \sum_{i=1}^{n} buy_{i,t} \times buyin_{i,t} \times price_{i,t} \end{aligned} \tag{11}$$

where:

- c1 and c2 is the transaction cost for selling and buying.
- $sell_{i,t}$ and $buy_{i,t}$ are dummy variables, indicating whether or not to buy or sell asset i at the end of period t.
- $buyin_{i,t}$ and $sellin_{i,t}$ denote the trade size of asset i at the end of period t, which should be greater than or equal to zero.

In addition, the share of asset i held by an investor at the end of period t, represented by $asset_t \times w_{i,t}$, satisfies the following equation:

$$\begin{aligned} asset_t \times w_{i,t} = asset_{t-1} \times w_{i,t-1} \times (1 + R_{i,t}) \\ + buy_{i,t} \times buyin_{i,t} \times price_{i,t} \\ - sell_{i,t} \times sellin_{i,t} \times price_{i,t} \end{aligned} \tag{12}$$

Obviously, $asset_t \times w_{i,t}$ equals to the cumulative value of $asset_{t-1} \times w_{i,t-1}$ in period t plus the shares bought($buy_{i,t} \times buyin_{i,t} \times price_{i,t}$) minus the shares sold($sell_{i,t} \times sellin_{i,t} \times price_{i,t}$). Since it does not make sense to buy and sell an asset simultaneously, which increases transaction costs, one of $sell_{i,t}$ and $buy_{i,t}$ must be zero.

If $asset_{t-1}$, $w_{i,t-1}$ and $w_{i,t}$ are given, we could adjust trading strategies to maximize the portfolio value $asset_t$. This optimization problem can be formulated as nonlinear programming, the objective function of which is to maximize $asset_t$, and the decision variables are $sell_{i,t}$, $buy_{i,t}$, $buyin_{i,t}$ and $sellin_{i,t}$. The nonlinear programming model can be expressed

as follows:

$$
\begin{aligned}
&\textbf{Maximize}: \quad asset_t \\
&\textbf{Subject to}: \\
&asset_t \times w_{i,t} = asset_{t-1} \times w_{i,t-1} \times (1 + R_{i,t-1}) - \\
&\qquad sell_{i,t} \times sellin_{i,t} \times price_{i,t} + \\
&\qquad buy_{i,t} \times buyin_{i,t} \times price_{i,t} \\
&asset_t = asset_{t-1} + asset_{t-1} \times \sum_{i=1}^{n} (w_{i,t-1} \times R_{i,t-1}) - \\
&\qquad c1 \times \sum_{i=1}^{n} sell_{i,t} \times sellin_{i,t} \times price_{i,t} - \\
&\qquad c2 \times \sum_{i=1}^{n} buy_{i,t} \times buyin_{i,t} \times price_{i,t} \\
&sell_{i,t} \times buy_{i,t} = 0 \\
&sell_{i,t} + buy_{i,t} = 1 \\
&sellin_{i,t} \geq 0 \\
&buyin_{i,t} \geq 0
\end{aligned} \tag{13}
$$

The Nelder-Mead simplex algorithm is used to solve the above optimization problems. This algorithm is one of the simplest ways to optimize a fairly good function. It only requires functional evaluation and is an ideal choice for minimizing problems. However, since it does not use any gradient evaluation, it may take longer to find the minimum value. To accelerate the convergence rate of the optimization model mentioned above, selecting an appropriate starting point is essential. An ideal starting point can shorten the model optimization time and avoid falling to a local minimum. This paper uses the following methods to determine the starting point: First, set the transaction cost as zero, that is, $c_1 = c_2 = 0$, and solve the asset value($asset_t$) through Eq.(11); Then, the transaction share($sellin_{i,t}$, $buyin_{i,t}$) is obtained by solving Eq.(12). The transaction cost of single trading is relatively low compared with the asset value. Therefore, the transaction share obtained by the above method should be close to the solution of the optimal model.

IV. METHODOLOGY

In the first part of this section, we describe the risk-return reinforcement learning(R3L) algorithm in detail. The second part of this paper introduces the structure of the neural network. The third part of this section is the architecture of Ape-X.

A. PROPOSED ALGORITHM

In the context of the deep reinforcement learning algorithm, the agent chooses the optimal action in light of the current environmental state to maximize cumulative return. This paper's algorithm is built on actor-critic architecture, consisting of actor and critic networks. Its framework is shown in Fig.2. In addition, each network has its corresponding target network, so the algorithm includes four networks, namely the actor-network, denoted by $\mu_\varphi(s)$ with parameter φ, and the critic- network, denoted by $K_\omega(s, a)$ with parameter ω, the target actor-network, denoted by $\mu'_{\varphi'}(s)$ and the target critic-network, denoted by $K'_{\omega'}(s, a)$.

In classical actor-critic architectures such as A3C, TD3, and DDPG, the actor-network updates φ by maximizing the expectation of cumulative rewards, and the critic network updates ω by minimizing the error between the evaluation value and the target value. The algorithm proposed in this paper is different from the above algorithms. Inspired by distributional reinforcement learning(DRL) initially proposed by [46], we estimate the distribution, rather than the expectation, of cumulative rewards by the critic network.

Distributional reinforcement learning is a new kind of reinforcement learning algorithm, mainly learning the distribution of cumulative rewards. The distributional Bellman operator(τ) is shown in Eq.(14).

$$\tau Z(s, a) \stackrel{D}{=} R(s, a) + \gamma Z(s', a') \tag{14}$$

where $Z(s, a)$ represents the cumulative return obtained by taking action a in state s, which is a random variable, $R(s, a)$ is the rewards function.

Under the Wasserstein metric, [46] proved that the distributional Bellman operator is a γ-contract operator. The learning task of DRL is to make the distribution $Z(s, a)$ and the target distribution $R(s, a) + \gamma Z(s', a')$ as similar as possible. Following [29], we utilize quantile regression to estimate network parameters. Quantile regression, which projects the update of distributional Bellman to the quantile distribution, uses a parameterized quantile distribution to approximate the value distribution. Let $[\theta_1, \theta_2, \ldots, \theta_N]$, which is the output of critic network, denotes the N quantiles of $Z(s, a)$. The target distribution is shown in Eq.(15), which can be regarded as ground truth.

$$\tau\theta'_j = r + \gamma\theta'_j, \quad \forall j \tag{15}$$

The loss function of critic network is defined in Eq.(16).

$$L_\omega = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{N} [\rho_{\hat{\tau}_i}(\tau\theta'_j - \theta_i)] \tag{16}$$

where

$$\rho_\tau = \|\tau - \delta_{\mu<0}\||u| = (\tau - \delta_{\mu<0})u \tag{17}$$

Because $|u|$ is not differentiable at zero, we take the Huber loss function, given in Eq.(18), instead of $|u|$.

$$\Gamma_\kappa = \begin{cases} \frac{1}{2}u^2 & if \quad |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & otherwise \end{cases} \tag{18}$$

Thus, we get a new loss function, also called the quantile Huber loss function, expressed as follows:

$$\rho^\kappa_\tau = \|\tau - \delta_{\mu<0}\|\Gamma_\kappa \tag{19}$$

As mentioned in the introduction section, the objective function of portfolio optimization is the weighted sum of the

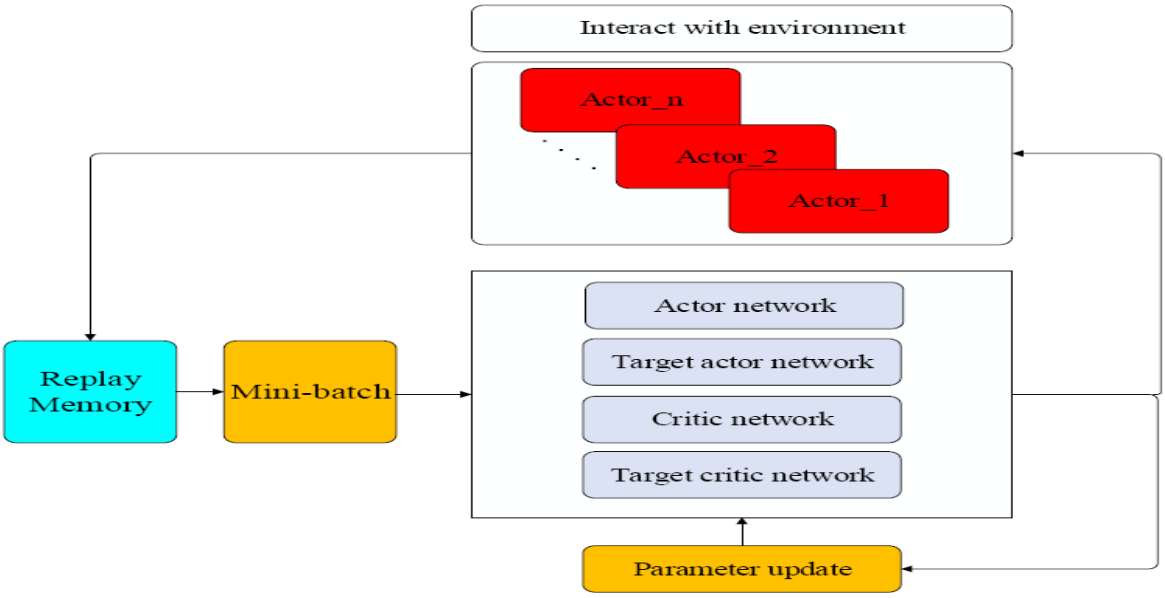

FIGURE 2. The general architecture of the proposed model.

mean and VaR of portfolio cumulative return. If we take N uniform quantiles, the mean of portfolio cumulative return, denoted by MR, can be approximately equal to the average of quantiles of cumulative return, shown in Eq.(20).

$$MR = \frac{\sum_{i=1}^{N} \theta_i}{N} \tag{20}$$

VaR is used to measure risk in this paper. VaR refers to the maximum portfolio loss for a given confidence level in a specific period. It can be described in Eq.(21), in which R_p and α denote portfolio cumulative return and confidence level, respectively.

$$Prob(R_p \leq -VAR_\alpha) = 1 - \alpha \tag{21}$$

Because VaR_α is a quantile of portfolio cumulative return, in this paper, it can also be expressed as the following:

$$VaR_\alpha = -\theta_{N\times(1-\alpha)} \tag{22}$$

Given the value of N and α, we can easily get VaR_α. For example, if $N = 100, \alpha = 95\%$, $VaR_{95\%} = -\theta_5$, if $N = 200, \alpha = 90\%$, $VaR_{90\%} = -\theta_{20}$.

According to classical modern portfolio theory, portfolio selection aims to construct an optimal portfolio model that maximizes expected returns under a given acceptable risk level(ϖ). This optimal model is shown in the following:

$$\begin{aligned} &Max \quad MR \\ &s.t. \quad VaR_\alpha = \varpi \end{aligned} \tag{23}$$

The optimization mentioned above process determines the optimal weights for various risk levels. In other words, investors have a variety of portfolio options, which unquestionably makes decision-making more challenging. The programming as mentioned above problem can be changed into a single objective programming problem using the following function in order to arrive at a single ideal solution:

$$\begin{aligned} U_\varphi &= MR - \zeta \times VAR_\alpha \\ &= \frac{\sum_{i=1}^{N} \theta_i}{N} + \zeta \times \theta_{N*(1-\alpha)} \end{aligned} \tag{24}$$

where U_φ is the so-called utility function, ζ (called Zeta) denotes the risk attitude of the investor, the higher ζ, the higher the risk aversion of investors, and the more conservative investment strategies will be adopted.

The loss function and utility function for a mini-batch, denoted by L and U respectively, are given in Eq.(25).

$$\begin{aligned} L &= \sum_{k=1}^{M} L_\omega^k \\ U &= \sum_{k=1}^{M} U_\varphi^k \end{aligned} \tag{25}$$

where M represent batch size.

B. NEURAL NETWORK

We need to model the neural network structure to explore functional patterns and extract informative features. Based on the time series nature of stock data, the gated recurrent unit(GRU) is utilized to construct an informative feature representation. The GRU has two gates, i.e., a reset gate and an update gate. These two gates determine which information can ultimately be used as the output of the gating recurrent unit. The reset gate determines how to combine the new input information with the previous memory, and the update gate defines the amount of prior memory saved to the current time step. The actor and critic network structure is shown in Fig.3,

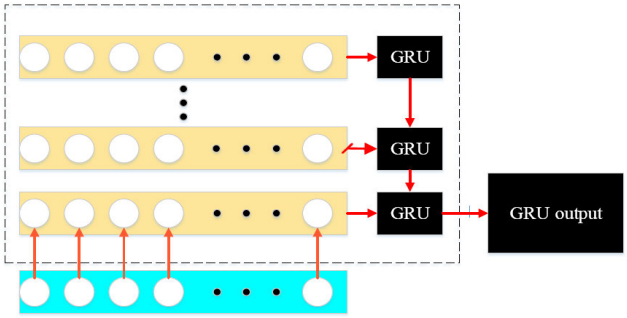

FIGURE 3. GRU structure.

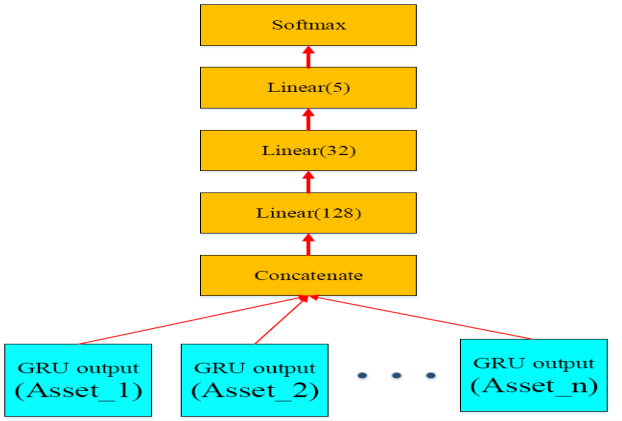

FIGURE 4. Actor network structure.

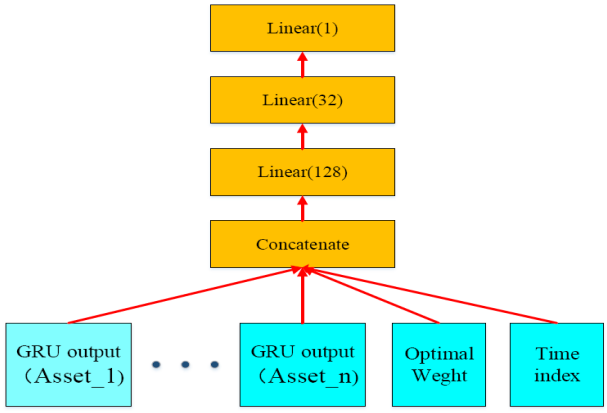

FIGURE 5. Critic network structure.

Fig.4 and Fig.5. Since GRU, LSTM, and Recurrent Neural Networks(RNN) have similar structural features, we also used LSTM and RNN networks in the sensitivity analysis.

In addition, Several authors have leveraged convolutional Neural Networks(CNN) to perform financial trading strategy [47], [48], treating the stock trading problem as a computer vision application by using time-series representative images as input. The convolutional neural network has unique advantages in speech recognition and image processing with its particular structure of local weight sharing. Its design is closer to the actual biological neural network. Weight sharing reduces the complexity of the network, especially the feature that the images of multi-dimensional input vectors can be directly input into the network, which avoids the complexity of data reconstruction in the feature extraction and classification process. Because of CNN's robust feature representation ability, we studied the impact of CNN on the performance of the proposed algorithm in the sensitivity analysis.

C. DISTRIBUTED FRAMEWORK

In this paper, to make the training result robust, the actor must interact with the environment at different epochs(or iterations) several times, which is very time-consuming. We utilize Ape-X architecture, proposed by [35], to speed up the training procedure. This algorithm decouples acting from learning and decomposes the learning process into three parts. In the first part, there are multiple actors. Each actor interacts with its respective environment based on the shared neural network, accumulating experience and putting it into the shared experience replay memory. We refer to this part, running on CPUs, as acting. In the second part, a single learner samples data from the replay memory and then updates the neural network. We refer to this part as the learning part running on a GPU. The third part is mainly responsible for data transmission. We refer to this part, running on CPUs, as communication.

We use a multiprocessing method to implement Ape-X. Specifically, there are 22 parallel processes, of which 20 are responsible for interacting with the environment, one process is accountable for updating network parameters, and one process is responsible for data exchange. The proposed method's general architecture and the algorithm's pseudo-code are shown in 1.

V. EXPERIMENTAL SETUP AND RESULT

In order to verify the effectiveness and robustness of the proposed trading strategy, this section details datasets, performance measures, benchmark models, technical details and experimental results. In addition, we also need to make the following assumptions: First, all transactions are made at the close price at the end of each trading period; Second, the market size is large enough that the price of security and market environment is not affected by the transactions; Third, since frequent adjustment of portfolio weights will generate many transaction costs, we adjust the portfolio weights once a week; finally, the investment period is set to one year.

A. DATASETS

We experiment with two different portfolios. The first portfolio consists of four famous exchange trading funds (ETFs) in the US market and a risk-free asset. The ETFs portfolio includes SPDR S&P 500 ETF Trust (SPY), Invesco QQQ Trust ETF(QQQ), SPDR Dow Jones Industrial

TABLE 1. Pseudocode of the proposed method.

Algorithm1. R3L

Input : bath size M, number of actors K, replay size R, exploration constant, learning rate

1: Initialize network weights (φ, ω)
2: Launch K actors and replicate network weights φ to each actor
4: **for** t=1,2,...,T, **do**
5: Sample M transitions (s_k, a_k, s'_k) of length N from replay buffer
6: Compute the N quantiles of $Z(s,a)$: $[\theta_{1,k}, \theta_{2,k}, \ldots, \theta_{N,k}] = K_\omega(s_k, a_k)$
6: Construct the target distributions:
$[\tau\theta'_{1,k}, \tau\theta'_{2,k}, ...\tau\theta'_{N,k}] = r + \gamma K_\omega(s'_k, \mu(s'_k))$
7: Update the parameters of critic network(ω) by minimize:
$loss = \frac{1}{M}\sum_k L^k_\omega$
8: Update the parameters of critic network(φ) by gradient ascend:
$$\nabla_\varphi \approx \frac{1}{M}\sum_k \nabla_{a_k} U(s_k, a_k)\nabla_\varphi \mu(s_k)$$
9: If $t = 0$ mod t_{target}, update the target network:
$$\varphi \leftarrow \tau\varphi + (1-\tau)\varphi', \omega \leftarrow \tau\omega + (1-\tau)\omega'$$
10: If $t = 0$ mod t_{actor}, replicate network weights to the actors
11: **end for**
12: **return** actor network parameters φ

Actor

1: **repeat**
2: Sample action $a = \mu_\varphi(s) + \epsilon$
3: Execute action a, observe reward r and state s'
4: Store (s, a, r, s') in replay buffer
5: **until** learner finishes

Average ETF(DIA), and iShares Russell 2000 ETF(IWM). The reason for choosing these four funds is that they are very representative and have a considerable scale. The SPDR S&P 500 ETF was jointly launched by State Street Corp and the American Stock Exchange on January 22, 1993. As of June 21, 2022, the ETF was the world's largest, with $335.7 billion under management. The QQQ is an index ETF that tracks the performance of 100 NASDAQ stocks, including the top 100 NASDAQ stocks. The Russell 2000 Index comprises 2000 listed companies with the lowest market value in the Russell Index, reflecting the stock price changes of small and medium-sized listed companies, including retail, real estate, finance and other industries. The second portfolio consists of stocks of four technology companies on NASDAQ, including ORCL, AAPL, TSLA, and GOOG, and a risk-free asset. The selected sample rate is daily. All data used in this paper is available on Yahoo Finance.

The sample period selected in this paper is from January 1, 2008, to September 31, 2022. The data for this period is rich enough to represent current market conditions. In addition, we can find that the sample period also includes the 2008 financial crisis period, which is more conducive to evaluating the robustness of trading strategies to this special event. Finally, The trading horizon is divided into both training and testing sets as follows:

- **Training set** : 01/01/2008 → 31/12/2018.
- **testing set** : 01/01/2019 → 31/09/2022.

The potential distribution of the stock trading data stream may change unpredictably over time, called concept drift. These may be caused by unexpected events (e.g. the Covid-19 pandemic). Following [51], we use passive detection methods to solve the problem of concept drift. First, we train agents based on historical data and then update the learning strategy with the arrival of new data to allow concept drift without discarding helpful knowledge.

Although the portfolio only includes five assets, it is straightforward to increase the number of assets.

B. PERFORMANCE MEASURES

Risk and return are inseparable in the investment decision process. Under normal circumstances, the risk is highly correlated with the return, and a high return means high risk. Therefore, the performance measures must include these two aspects. In this article, we use four types of performance measures to evaluate the proposed trading strategy. The first type of metric measures the profitability of the investment strategy, i.e., total return. The second type of metric measures investment risk, including variance and VaR. The third metric type considers risk and returns, including the Sharpe and Sortino ratios. The last type of metric is average turnover. More detail about performance measures is as follows.

- **Total return(TR)**. The total return is the rate of return over a period. Total return, computed using Eq.(26), includes capital gains, interest, realized distributions, and dividends.

$$TR = (Q_{[T]} - Q_{[0]})/Q_{[0]} \tag{26}$$

where $Q_{[0]}$ is the value of the initial investment;$Q_{[T]}$ is the value of the portfolio at the end of the investment period.

- **Value at risk(VaR)**. As mentioned above, VaR calculates the maximum loss of a portfolio over a given period on a specified confidence level.

- **Sharpe ratio(SR1)**. The Sharpe ratio, first proposed by [49], is a measure of risk-adjusted return. This ratio, computed using Eq.(27), reflects how much the return on risk per unit exceeds the risk-free return. If the Sharpe ratio is

positive, a portfolio's average net value growth rate exceeds the risk-free interest rate.

$$SR = (E[R_p] - R_f)/\sigma_p \tag{27}$$

where $E[R_p]$ is the expectation of portfolio return, R_f is risk-free rate, σ_p is standard deviation of portfolio return.

• **Sortino ratio(SR2)**. The Sortino ratio is a risk-adjustment metric used to determine the additional return for each unit of downside risk. It is similar to the Sharpe ratio, but the Sortino ratio uses the lower partial standard deviation rather than the total standard deviation to distinguish between adverse and favorable fluctuations. The Sortino ratio can be expressed as follows:

$$SR = (E[R_p] - R_f)/DR \tag{28}$$

where DR is the lower partial standard deviation of portfolio return.

• **Standard deviation(SD)**. The standard deviation of the portfolio variance can be calculated as the square root of the portfolio variance.

• **Average turnover(AT)**. Average turnover represents the average level of change in portfolio weight, which is defined in Eq.(29).

$$AT = 1/(2t_f)\sum_{t=0}^{t_f-1}\sum_{i=1}^{l} \mid w_{t+1,i} - w_{t,i} \mid \tag{29}$$

where t_f is the investment horizon,$w_{t,l}$ denote weight parameter of asset i in investment period t.

C. BENCHMARK MODELS

To analyze the effectiveness of the proposed strategy, some benchmark strategies, summarised hereafter, are selected for comparison.

• **Buy and hold(B&H)**. B&H is used as a benchmark strategy by many researchers to compare with their proposed strategies. Suppose that the holding proportion of all five assets is 20% in the B&H strategy and remains unchanged throughout the investment period.

• **Sell and hold(S&H)**. S&H is also widely used as a benchmark strategy. Assuming that in the S&H strategy, the holding proportion of all four risky assets is -25%, the proportion of risky-free assets is 200%, all of which remain unchanged throughout the investment period.

• **Random selected(RN)**. According to the efficient market hypothesis (EMH), all valuable information has been timely, accurate, and fully reflected in the stock price trend, including the current and future value of the enterprise. Without market manipulation, investors cannot obtain excess profits higher than the market average by analyzing past prices. Any trading strategy based on historical data differs from the randomly selected strategy.

• **Mean − variance model**. The mean-variance model, introduced by Markowitz in 1952, aims to find the best portfolio only by the first two moments of cumulative return.

TABLE 2. Summary of hyper-parameters.

hyper-parameter	value	hyper-parameter	value
time window size(n)	60	replay memory	2000
learning rate(lr)	1e-5	number of parameter update	80000
batch size	32	discount factor(γ)	0.9
confidence level of VaR(α)	0.95	parameter of Huber Loss(κ)	1.0
short selling parameter(δ)	3.0	soft update parameter(τ)	0.5
initial money	10000.	risk-free rate	3.8e-4
risk attitude parameter(ζ)	0.5	num of layer for GRU	2
Output number of critic network	200	transaction cost	0.0020

Suppose there are n kinds of assets, $R = (r_1, \ldots r_n)^T$ represents the expected return of a portfolio, $W = (W_1, \ldots W_n)^T$ is the weight vector, Σ is the covariance vector of return, ζ is risk aversion coefficient, $\mathbf{1}$ represents $n \times 1$ dimensional unit vector, we establish the following optimization model based on utility maximization:

$$max \;\; U = W^T R - \zeta W^T \Sigma W \tag{30}$$

$$s.t. 1^T W = 1 \tag{31}$$

D. TECHNIQUE DETAIL

We obtain time window size and other hyper-parameters, including replay buffer size, batch size, and discount factor, through several tuning rounds. In addition, we need to set some other hyper-parameters before training. Suppose the risk-free rate of return is 2%, and the equivalent weekly return is 0.038%. The transaction cost of buying and selling an asset is set to 0.02%. The risk aversion parameter(ζ) and short selling parameter(δ)are set to 0.5 and 3; because these two parameters have significant impacts on investment decisions and portfolio returns, we perform sensitivity analysis for them in subsection V-E. The networks were trained through the ADAM optimization procedure with a learning rate of 10^{-5}. The activation function is the Leaky Relu function. All parameters are summarised in Table 2. Finally, the algorithms proposed in the paper are implemented in python3.7.11 using pytorch1.10.2 and were run on a PC that contains a sixteen-core 2.50GHz CPU with 16GB RAM and NVIDIA Geforce RTX 3060 GPU.

E. RESULT AND DISCUSSION

The first detailed analysis concerns the execution of the proposed algorithm on the ETFs portfolio. Fig.6 illustrates the portfolio value trend when applying the proposed algorithm and benchmark strategies in the testing set. We observe that the final portfolio value of the proposed algorithm is 19.51% higher than the S&H strategy, 18.0% higher than the RN strategy, 8.1% higher than the MV strategy, and 7.9% higher than the B&H strategy.

Table 3 further presents output performance measures results when using the R3L algorithm and the benchmark strategies. The R3L algorithm is optimal regarding risk, return, and overall performance. The Sharpe ratio of the proposed algorithm is 19.51% higher than the S&H strategy, 18.0% higher than the RN strategy, 8.1% higher than the MV

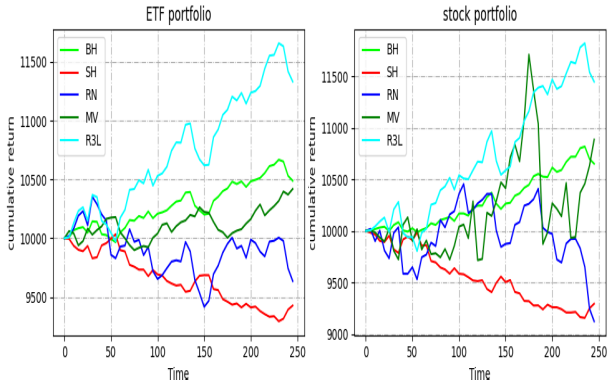

FIGURE 6. Profit curves of different models on each portfolio.

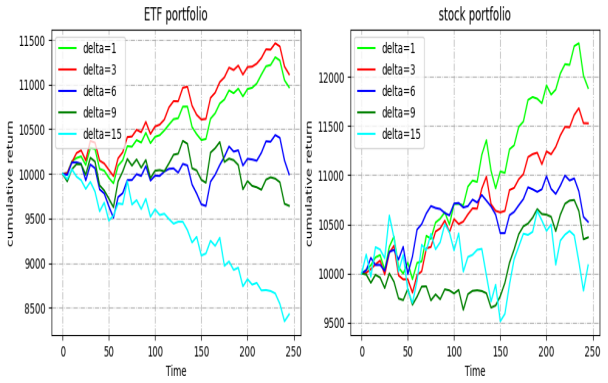

FIGURE 7. Profit curves of different δ on each portfolio.

TABLE 3. Main result.

	delta	TR	SD	SR1	VAR	SR2	AT
	B&H	5.00%	0.0264	0.0889	0.0374	0.1260	0.0000
	S&H	-5.24%	0.0330	-0.0870	0.0419	-0.1392	0.0000
	RN	-4.00%	0.0329	-0.0414	0.0435	-0.0541	0.3291
	MV	4.86%	0.0605	0.0968	0.0875	0.1128	0.0929
	R3L	13.32%	0.0266	0.1100	0.0338	0.1681	0.0966
	B&H	10.09%	0.0317	0.0780	0.0203	0.1173	0.0000
	S&H	-20.62%	0.0396	-0.1280	0.0575	-0.1657	0.0000
	RN	-8.76%	0.0423	-0.0469	0.0584	-0.0536	0.3046
	MV	6.75%	0.0947	0.0893	0.0115	0.1718	0.0929
	R3L	14.47%	0.0298	0.0943	0.0369	0.1571	0.0849

TABLE 4. Sensitive analyse of delta.

	delta	TR	SD	SR1	VAR	SR2	AT
	1	9.70%	0.0266	0.0815	0.0367	0.1177	0.0247
	3	11.16%	0.0270	0.0912	0.0362	0.1391	0.1029
	6	-0.05%	0.0323	0.0060	0.0484	0.0183	0.2362
	9	-3.59%	0.0349	-0.0346	0.0605	-0.0452	0.3681
	15	-15.74%	0.0383	-0.1034	0.0620	-0.1302	0.6059
	1	18.85%	0.0332	0.1117	0.0432	0.1962	0.0288
	3	15.26%	0.0288	0.0991	0.0346	0.1834	0.0941
	6	5.24%	0.0276	0.0220	0.0344	0.0827	0.1995
	9	3.66%	0.0350	0.0300	0.0478	0.0514	0.2721
	15	0.83%	0.0532	-0.0095	0.0845	-0.0170	0.4229

strategy, and 18.0% higher than the B&H strategy. In addition, as the stock market was primarily bullish throughout the test period, the B&H strategy outperformed other benchmark strategies most of the time, while the performance of the S&H strategy was the worst. The performance of the RN strategy could be better. Since we did not consider the company's financial and internal non-public information, it does not mean that the efficient market hypothesis is not tenable.

The same detailed analysis is performed on the stock portfolio, which shows different characteristics compared to the ETFs portfolio. Fig.6 illustrates the portfolio value trend of different strategies. Similar to the ETFs portfolio, the R3L strategy has the highest cumulative value at the end of the investment period. Table 3 presents output performance measures results when using different trading strategies. It can be noticed that from the perspective of the Sharpe ratio, Sortino ratio, and VaR, the R3L algorithm is also optimal. In this study, the critical parameters delta (*delta*) and zeta (*zeta*) significantly impact the suggested algorithm's overall performance. Delta refers to the maximum permitted short selling ratio; the higher the delta, the greater the potential for profit for investors during a bad market. Zeta is a measure of an investor's risk attitude; the larger it is, the more risk-averse the investor is, and the more daring their investment plans are likely to be. The ability to extract features from various network structures vary, as indicated in subsection IV-B. In order to choose the optimum algorithm, it is vital to examine how network architecture affects algorithm performance. On the R3L strategy, we perform the sensitive analysis of the delta, theta, and network structure. Regarding various values, the proposed model's performance is assessed.

• **Sensitive analysis of delta**. Fig.7 illustrates the portfolio value trend when applying the R3L algorithm with a different delta. Portfolio value shows an upward trend over time for the ETFs and stock portfolios when delta equals 1 and 3. In this case, investors can obtain positive returns at the end of the investment period. In contrast, in other cases, the portfolio value shows significant volatility over time, and investors receive negative returns at the end of the investment period. Table 4 further shows the overall performance of the proposed strategy concerning different delta values. From the perspective of total return, Sharpe ratio, and Sortino ratio, $\delta = 3$ is optimal for the ETFs portfolio, and $\delta = 1$ is optimal for the stock portfolio.

It can be seen that the maximum amount of short selling allowed is not the bigger, the better, which is explained by the fact that, although short selling can provide investors with profit opportunities in a bear market, it also brings more risks. Therefore, it is essential to control the scale of short selling appropriately.

• **Sensitive analysis of zeta**. Fig.8 illustrates the portfolio value trend when applying the R3L algorithm with different zeta. Table 5 further shows the overall performance of the proposed strategy. It can be noticed that the change in portfolio value over time is similar for different theta. At the end of the investment period, the portfolio value is the greatest for the ETFs portfolio when theta equals 3. From the Sharpe and Sortino ratios perspective, $\zeta = 3$ is also

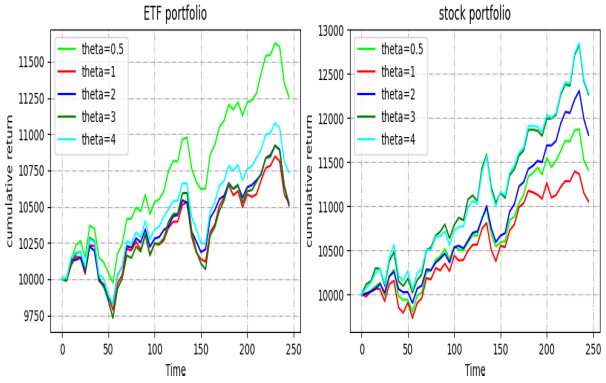

FIGURE 8. Profit curves of differentθ on each portfolio.

TABLE 5. Sensitive analyse of zeta.

	zeta	TR	SD	SR1	VAR	SR2	AT
	0.5	12.58%	0.0267	0.1029	0.0341	0.1576	0.0991
	1.0	5.23%	0.0275	0.0453	0.0383	0.0703	0.1122
	2.0	5.10%	0.0270	0.0486	0.0395	0.0814	0.1141
	3.0	5.26%	0.0305	0.0521	0.0432	0.0854	0.1019
	4.0	7.37%	0.0264	0.0570	0.0401	0.0887	0.0955
	0.5	14.11%	0.0297	0.0898	0.0372	0.1548	0.0979
	1.0	10.53%	0.0281	0.0649	0.0383	0.1169	0.1156
	2.0	18.09%	0.0253	0.1295	0.0344	0.2470	0.0788
	3.0	22.66%	0.0338	0.1236	0.0448	0.2623	0.0867
	4.0	22.83%	0.0351	0.1247	0.0445	0.2450	0.0835

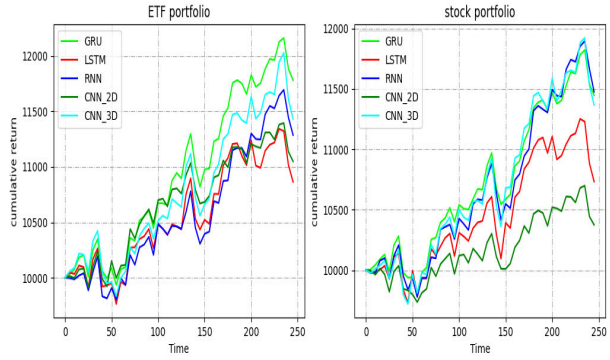

FIGURE 9. Profit curves of different network structure on each portfolio.

optimal. The portfolio value is the greatest for the stock portfolio when theta equals 4. From the standpoint of the Sharpe ratio and Sortino ratio, $\zeta = 4$ is also optimal. Nevertheless, the VaR is the lowest when theta equals 2 for the ETFs and stock portfolios.

From the above analysis, it can be seen that a higher risk aversion coefficient does not necessarily lead to lower risk and returns, which contradicts the classical portfolio theory. One possible explanation is that our algorithm can predict the stock based on historical data, so the algorithm proposed in this article can enhance portfolio returns and reduce risk.

• **Sensitive analysis of network structure**. Fig. 9 illustrates the portfolio value trend when applying the R3L algorithm with a different network structure. It can be observed

TABLE 6. Sensitive analyse of network.

	Network	TR	SD	SR1	VAR	SR2	AT
	GRU	17.82%	0.0342	0.1089	0.0457	0.2118	0.0865
	LSTM	8.65%	0.0337	0.0433	0.0531	0.0723	0.1060
	RNN	12.88%	0.0335	0.0660	0.0455	0.1173	0.1093
	COONV2D	10.50%	0.0321	0.0680	0.0355	0.1381	0.1258
	CONV3D	14.33%	0.0352	0.0758	0.0526	0.1584	0.0985
	GRU	14.47%	0.0298	0.0943	0.0369	0.1571	0.0849
	LSTM	7.34%	0.0357	0.0594	0.0534	0.0917	0.1110
	RNN	14.75%	0.0300	0.0907	0.0431	0.2078	0.1125
	CONV2D	3.76%	0.0283	0.0377	0.0326	0.0840	0.1541
	CONV3D	13.67%	0.0360	0.0710	0.0521	0.1460	0.0903

that the changing trend of portfolio value is consistent under different network structures, and only the final cumulative value is slightly different. GRU and RNN obtained the maximum cumulative value for the ETFs and stock portfolios. Table 6 further shows the overall performance. We find that GRU performs best in the ETFs portfolio from the perspective of the Sharpe ratio and Sortino ratio, and RNN performs best in the stock portfolio from the perspective of the Sortino ratio. In contrast, GRU performs best from the perspective of the Sharpe ratio.

VI. CONCLUSION AND FUTURE WORK

We propose a novel deep reinforcement learning algorithm called risk-return reinforcement learning(R3L) to derive a portfolio trading strategy. Compared with previous studies, the main innovations in this paper include three aspects:First, we build a portfolio optimization model solved by an enhanced deep reinforcement learning algorithm based on actor-critic architecture.Second we implement portfolio short selling through a linear transformation.Third, we use the Ape-x algorithm to accelerate training. Experiments carried out on the performance of the R3L algorithm demonstrate that the proposed R3L is superior to the traditional benchmark strategies, such as buy-and-hold, sell-and-hold, random select, and mean-variance strategies. Based on the total return, Sharpe ratio, VaR and Sortino ratio results in the one-year investment period, the R3L is more profitable with lower risk than other benchmark strategies. In addition, based on the results for the average turnover rate, the R3L algorithm is more suitable for application in the real world than the benchmark strategies are. In addition, although short selling allows investors to profit in a bear market, the maximum short selling permitted ratio is not the bigger, the better. Therefore, investors should choose optimal short-selling parameters according to investment objectives, asset types, and other factors to maximize portfolio return. Similarly, we must choose appropriate risk attitude parameters and network structure to optimize the portfolio's overall performance.

Future research can be carried out from the following aspects: First, according to to [50], VaR is not a coherent risk measure, so we could consider using other risk measurements, such as conditional value at risk(CVaR), to construct the portfolio optimization problem in future research; Second, this paper assumes that the trading volume is small

compared to the market size, so the trading behaviour of a single agent does not affect the market environment and stock prices. However, the trading behaviour, even with a small volume, still has a subtle impact on the stock price and market environment. How to model the market environment is one of the future research directions; Third, the decision-making process of algorithmic trading can be divided into two parts: selecting asset types and determining optimal portfolio weight. Therefore, subsequent research can be used hierarchical deep reinforcement learning(HDRL) to handle portfolio optimization problems; Fourth, Our proposed paper applies a model-free RL algorithm that is sample inefficient and does not account for the stability issues caused by the non-stationary financial market environment. In the following work, we could construct a model-based RL architecture, which uses an architecture consisting of a prediction module(e.g. [52]) and RL.

REFERENCES

[1] A. F. Perold, "The implementation shortfall: Paper versus reality," *J. Portfolio Manag.* vol. 14, no. 3, p. 4, 1988.
[2] A. Robert and J. Lorenz, "Adaptive arrival price," *Trading* vol. 2007, no. 1, pp. 59–66, 2007.
[3] S. A. Berkowitz, D. E. Logue, and E. A. Noser, "The total cost of transactions on the NYSE," *J. Finance*, vol. 43, no. 1, pp. 97–112, Mar. 1988.
[4] A. N. Madhavan and V. Panchapagesan, "The first price of the day," *J. Portfolio Manage.*, vol. 28, no. 2, pp. 101–111, Jan. 2002.
[5] P. N. Kolm and L. Maclin, *Algorithmic Trading, Optimal Execution, and Dyna Mic Port Folios*. 2011.
[6] M. Kritzman, "Are optimizers error maximizers?" *J. Portfolio Manage.*, vol. 32, no. 4, pp. 66–69, Jul. 2006.
[7] T. Hendershott and R. Riordan, "Algorithmic trading and the market for liquidity," *J. Financial Quant. Anal.*, vol. 48, no. 4, pp. 1001–1024, Aug. 2013.
[8] E. Elina and A. Swartling, "UCD guerrilla tactics: A strategy for implementation of UCD in the Swedish Defence," in *Human Work Interaction Design-HWID*. 2012.
[9] M. S. Tovarnov and N. V. Bykov, "Reinforcement learning reward function in unmanned aerial vehicle control tasks," 2022, *arXiv:2203.10519*.
[10] M. Tommaso, "Safe online robust exploration for reinforcement learning control of unmanned aerial vehicles," Tech. Rep., 2017.
[11] S. Bär, F. Bär, S. Pol, and T. Meisen, "Skalierung der online-job-shop-planung durch reinforcement learning: In flexiblen fertigungssystemen für verschiedene produkte," *atp magazin*, vol. 63, no. 5, pp. 52–59, May 2022.
[12] B. Ryu, A. An, Z. Rashidi, J. Liu, and Y. Hu, "Towards topology aware pre-emptive job scheduling with deep reinforcement learning," in *Proc. 30th Annu. Int. Conf. Comput. Sci. Softw. Eng.*, Nov. 2020, pp. 83–92.
[13] M. Yuan, J. Shan, and K. Mi, "Deep reinforcement learning based game-theoretic decision-making for autonomous vehicles," *IEEE Robot. Autom. Lett.*, vol. 7, no. 2, pp. 818–825, Apr. 2022.
[14] J. Laserna, A. Otero, and E. D. L. Torre, "A multi-FPGA scalable framework for deep reinforcement learning through neuroevolution," in *Proc. Int. Symp. Appl. Reconfigurable Comput.* Cham, Switzerland: Springer, Cham, 2022, pp. 47–61.
[15] G. Gil and F. Axel, "A very brief introduction to deep reinforcement learning," B.S. thesis. Universitat Politècnica de Catalunya, Barcelona, Spain, 2017.
[16] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai, "Deep direct reinforcement learning for financial signal representation and trading," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 3, pp. 653–664, Mar. 2017.
[17] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1995–2003.
[18] D. Fengqian and L. Chao, "An adaptive financial trading system using deep reinforcement learning with candlestick decomposing features," *IEEE Access*, vol. 8, pp. 63666–63678, 2020.
[19] M. R. Alimoradi and A. Husseinzadeh Kashan, "A league championship algorithm equipped with network structure and backward Q-learning for extracting stock trading rules," *Appl. Soft Comput.*, vol. 68, pp. 478–493, Jul. 2018.
[20] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, no. 2, pp. 654–669, 2018.
[21] Z. Zihao, S. Zohren, and S. Roberts, "Deep reinforcement learning for trading," *J. Data Sci.* vol. 2, no. 2, pp. 25–40, 2020.
[22] X. Wu, H. Chen, J. Wang, L. Troiano, V. Loia, and H. Fujita, "Adaptive stock trading strategies with deep reinforcement learning methods," *Inf. Sci.*, vol. 538, pp. 142–158, Oct. 2020.
[23] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," 2018, *arXiv:1808.09940*.
[24] C. Betancourt and W.-H. Chen, "Deep reinforcement learning for portfolio management of markets with a dynamic number of assets," *Exp. Syst. Appl.*, vol. 164, Feb. 2021, Art. no. 114002.
[25] S. Portfolio, "Harry Markowitz," *J. Finance*, vol. 7, no. 1, pp. 77–91, 1952.
[26] A. S. Krishna, "Death of portfolio diversification," *Guidelines Authors*, 2013.
[27] S. Rajgopal and T. Shevlin, "Empirical evidence on the relation between stock option compensation and risk taking," *J. Accounting Econ.*, vol. 33, no. 2, pp. 145–171, Jun. 2002.
[28] G. Lucarelli and M. Borrotti, "A deep Q-learning portfolio management framework for the cryptocurrency market," *Neural Comput. Appl.*, vol. 32, no. 23, pp. 17229–17244, Dec. 2020.
[29] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, 2018, pp. 1–10.
[30] E. Asodekar, A. Nookala, S. Ayre, and A. V. Nimkar, "Deep reinforcement learning for automated stock trading: Inclusion of short selling," in *Proc. Int. Symp. Methodologies Intell. Syst.* Cham, Switzerland: Springer, 2022, pp. 187–197.
[31] X. Du, J. Zhai, and K. Lv, "Algorithm trading using Q-learning and recurrent reinforcement learning," *Positions* vol. 1, no. 1, pp. 1–7, 2016.
[32] L. Chen and Q. Gao, "Application of deep reinforcement learning on automated stock trading," in *Proc. IEEE 10th Int. Conf. Softw. Eng. Service Sci. (ICSESS)*, Oct. 2019, pp. 29–33.
[33] C. Y. Huang, "Financial trading as a game: A deep reinforcement learning approach," 2018, *arXiv:1807.02787*.
[34] H. Park, M. K. Sim, and D. G. Choi, "An intelligent financial portfolio trading strategy using deep Q-learning," *Exp. Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113573.
[35] D. Horgan, J. Quan, D. Budden, G. Barth-Maron, M. Hessel, H. Van Hasselt, and D. Silver, "Distributed prioritized experience replay," 2018, *arXiv:1803.00933*.
[36] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 30, no. 1, 2016, pp. 1–7.
[37] G. Jeong and H. Y. Kim, "Improving financial trading decisions using deep Q-learning: Predicting the number of shares, action strategies, and transfer learning," *Exp. Syst. Appl.*, vol. 117, pp. 125–138, Mar. 2019.
[38] D. Vezeris, I. Karkanis, and T. Kyrgos, "AdTurtle: An advanced turtle trading system," *J. Risk Financial Manage.*, vol. 12, no. 2, p. 96, Jun. 2019.
[39] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017, *arXiv:1706.10059*.
[40] P. C. Pendharkar and P. Cusatis, "Trading financial indices with reinforcement learning agents," *Exp. Syst. Appl.*, vol. 103, pp. 1–13, Aug. 2018.
[41] M. Schnaubelt, "Deep reinforcement learning for the optimal placement of cryptocurrency limit orders," *Eur. J. Oper. Res.*, vol. 296, no. 3, pp. 993–1006, Feb. 2022.
[42] F. Soleymani and E. Paquet, "Deep graph convolutional reinforcement learning for financial portfolio management – DeepPocket," *Exp. Syst. Appl.*, vol. 182, Nov. 2021, Art. no. 115127.
[43] M. Taghian, A. Asadi, and R. Safabakhsh, "Learning financial asset-specific trading rules via deep reinforcement learning," *Exp. Syst. Appl.*, vol. 195, Jun. 2022, Art. no. 116523.
[44] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[45] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[46] M. G. Bellemare and W. D. R. Munos, "A distributional perspective on reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 449–458.
[47] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, and D. R. Recupero, "A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning," *Int. J. Speech Technol.*, vol. 51, no. 2, pp. 889–905, Feb. 2021.
[48] S. Barra, S. M. Carta, A. Corriga, A. S. Podda, and D. R. Recupero, "Deep learning and time series-to-image encoding for financial forecasting," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 3, pp. 683–692, May 2020.
[49] W. F. Sharpe, "The Sharpe ratio," *J. Portfolio Manage.*, vol. 21, no. 1, pp. 49–58, Oct. 1994.
[50] R. T. Rockafellar and S. Uryasev, "Optimization of conditional value-at-risk," *J. Risk*, vol. 2, pp. 21–42, Feb. 2000.
[51] F. Bertoluzzo and M. Corazza, "Testing different reinforcement learning configurations for financial trading: Introduction and applications," *Proc. Econ. Finance*, vol. 3, pp. 68–77, Jan. 2012.
[52] P. Mantalos, A. Karagrigoriou, L. Stelec, P. Jordanova, P. Hermann, J. Kisel'ák, J. Hudák, and M. Stehlík, "On improved volatility modelling by fitting skewness in ARCH models," *J. Appl. Statist.*, vol. 47, no. 6, pp. 1031–1063, Apr. 2020.
[53] M. Hussonnois and J.-Y. Jun, "End-to-end autonomous driving using the Ape-X algorithm in Carla simulation environment," in *Proc. 13th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2022, pp. 18–23.
[54] S. Xu, H. Kuang, Z. Zhi, R. Hu, Y. Liu, and H. Sun, "Macro action selection with deep reinforcement learning in starcraft," in *Proc. AAAI Conf. Artif. Intell. Interact. Digit. Entertainment*, vol. 15, no. 1, 2019, pp. 94–99.
[55] M. Hussonnois and J.-Y. Jun, "End-to-end autonomous driving using the Ape-X algorithm in Carla simulation environment," in *Proc. 13th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, Jul. 2022, pp. 18–23.
[56] A. Namdari and T. S. Durrani, "A multilayer feedforward perceptron model in neural networks for predicting stock market short-term trends," *Oper. Res. Forum*. vol. 2, no. 3, p. 38, 2021.
[57] R. Kim, C. Ho So, M. Jeong, S. Lee, J. Kim, and J. Kang, "HATS: A hierarchical graph attention network for stock movement prediction," 2019, *arXiv:1908.07999*.

BOYI JIN was born in Xi'an, Shanxi, China, in 1979. He received the Ph.D. degree in management from the Shanghai University of Finance and Economics, China, in 2011.

Since 2011, he has been a Research Assistant with the Shandong University of Finance and Economics. He is the author of three books and more than 30 articles. His research interests include portfolio optimization and risk management.