IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 7, 2021, accepted May 16, 2021, date of publication May 20, 2021, date of current version June 2, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3082186

Novel Deep Reinforcement Algorithm With Adaptive Sampling Strategy for Continuous Portfolio Optimization

SZU-HAO HUANG[1], (Member, IEEE), YU-HSIANG MIAO[2], AND YI-TING HSIAO[2]
[1]Department of Information Management and Finance, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan
[2]Institute of Information Management, National Yang Ming Chiao Tung University, Hsinchu 30010, Taiwan

Corresponding author: Szu-Hao Huang (szuhaohuang@nctu.edu.tw)

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Contract MOST 110-2622-8-009-014-TM1, Contract MOST 109-2221-E-009-139, Contract MOST 109-2622-E-009-002-CC2, and Contract MOST 109-2218-E-009-015, and in part by the Financial Technology (FinTech) Innovation Research Center, National Yang Ming Chiao Tung University.

ABSTRACT Quantitative trading targets favorable returns by determining patterns in historical data through statistical or mathematical approaches. With advances in artificial intelligence, many studies have indicated that deep reinforcement learning (RL) can perform well in quantitative trading by predicting price change trends in the financial market. However, most of the related frameworks display poor generalizability in the testing stage. Thus, we incorporated adversarial learning and a novel sampling strategy for RL portfolio management. The goal was to construct a portfolio comprising five assets from the constituents of the Dow Jones Industrial Average and to achieve excellent performance through our trading strategy. We used adversarial learning during the RL process to enhance the model's robustness. Moreover, to improve the model's computational efficiency, we introduced a novel sampling strategy to determine which data are worth learning by observing the learning condition. The experimental results revealed that the model with our sampling strategy had more favorable performance than the random learning strategy. The Sharpe ratio increased by 6%–7%, and profit increased by nearly 45%. Thus, our proposed learning framework and the sampling strategy we employed are conducive to obtaining reliable trading rules.

INDEX TERMS Portfolio management, reinforcement learning, adversarial learning.

I. INTRODUCTION

Portfolio management is a process of selecting and supervising a group of financial products that meet an individual's long-term financial objectives. Within a certain period, people buy and sell portions of diverse assets to obtain profits and achieve risk diversification. To reach this goal, portfolio managers who employ quantitative trading methods attempt to obtain favorable returns by identifying patterns in historical data through statistical or mathematical approaches. With advances in artificial intelligence, numerous studies have indicated that deep reinforcement learning (RL) performs well in quantitative trading in terms of predicting price change trends in the financial market. Thus, various machine learning, deep learning, and RL applications have been developed for financial market trading. In particular, numerous studies have attempted to predict price trends by employing supervised learning applications, such as support vector machines or deep neural networks. Moody and Saffell [1] and Deng *et al.* [2] have applied the model-free method to solve the trading problem without the step of predicting future prices. However, both frameworks generated discrete trading signals for one asset, which might not represent the optimal trading decision in real-world conditions. To address this shortcoming—focusing on a single asset—some studies have considered the portfolio problem. Jiang *et al.* [3] applied recurrent network techniques to a portfolio in the cryptocurrency market, and Yu *et al.* [4] proposed a framework that could predict future prices, generate data for training, and learn from the experience of experts. Nevertheless, according to the random walk theory, market price changes are highly stochastic and challenging to simulate. Hence, the aforementioned methods have the problem of poor generalizability. That is, a given model may exhibit favorable performance in the training set but perform poorly during testing. To address these problems, [5] and [6] proposed an adversarial learning framework for stock prediction. However, these studies have only focused on predicting future market trends rather than on a particular trading decision. Thus, to maximize the generalizability of our portfolio management model, we introduced

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

adversarial learning to introduce perturbations into our input stock price streams, making our model more robust against highly stochastic price data. This measure prevents our model from ''memorizing'' the input data and model overfitting; it also increases the agent's ability to explore the environment. Most RL methods typically fail to control risks. To address this shortcoming, understanding the correlations between different assets is essential. References [7] and [8] utilized autoencoders for important feature extraction among different stock prices, but these methods were only applied to passive investment strategies. Therefore, in this study, we proposed a novel market representation network to extract latent information between different stocks. Based on this latent information, our RL agent can acquire the future variance of various assets and further control the risks. Another problem for RL frameworks is the high computational costs [9]. Therefore, we designed a novel sampling strategy to reduce the immense computational resources required for RL; this strategy involved selecting the assets that cannot be effectively managed by agents during the training phase. Specifically, by increasing the probability of sampling these assets, our agent is more likely to spend more time and computing power on managing portfolios that they are unfamiliar with. Thus, in this paper, we propose a comprehensive RL framework with the following contributions:

- We provided an adaptive sampling strategy to increase computation efficiency; training performance increased by 6 %–7 % in terms of the Sharpe ratio and 45% in terms of profit.
- We adopted adversarial learning to add a sampled noise to the input feature (stock price data); this measure increased the generalization ability of the model.
- We adopted the market representation network to pretraining our model to control for the risks during the training period. The model with the market representation network reduced the maximum drawdown (MDD) by 40 % and increased the Sharpe ratio by 22 times.

The remaining parts of this paper are organized as follows: Section II provides a review of the related works; Section III provides an introduction of the improved deep RL network; Section IV describes the experimental results; and finally, the major contributions and future research directions are summarized in Section V.

II. LITERATURE REVIEWS

In this section, we introduce works related to portfolio management. We first introduce conventional approaches for portfolio management and then introduce modern (learning-based) portfolio management frameworks. Finally, we also introduce several common RL and deep learning generalization techniques.

A. CONVENTIONAL APPROACH IN PORTFOLIO MANAGEMENT

Modern portfolio theory or mean–variance theory proposed by Markowitz uses a mathematical solution that constructs a portfolio maximizing the expected return and considers the given level of risk [10]. According to that theory, investors must only consider the trade-off between the mean and variance of the portfolio return and must seek the maximum expected return for a given level of the risk profile to increase the certainty of achieving an expected return. However, mean–variance theory has some limitations [11] because asset volatility is required for constructing the model, and determining an asset's future volatility is challenging in practice. Momentum investment is a well-known quantitative investment strategy. According to this strategy, the momentum effect is used to reveal the price stickiness of stocks over a certain period; this information is then used to predict price trends and make investment decisions [12]. Rouwenhorst observed similar momentum patterns for European and emerging stock markets and reported that the momentum profit was not limited to a particular market but was present in all 12 European markets investigated [13].

B. MACHINE LEARNING AND DEEP LEARNING APPROACH IN PORTFOLIO MANAGEMENT

Machine learning, with its excellent pattern-extraction ability from input data, has also been applied in portfolio management. Koyano and Ikeda [14] applied semisupervised learning to the posts in stock blogs to expand the follow-the-loser portfolio strategy. This approach aimed to predict the stock price by analyzing blog posts associated with bullish or bearish emotions to maximize the expected cumulative return of the portfolio. Xing *et al.* [15] converted sentiment information into market information by applying an ensemble of evolving clustering and long short-term memory (LSTM). These market information perspectives were later integrated into modern portfolio theory through a Bayesian approach. Malandri *et al.* [16] proposed an optimal allocation strategy for portfolios by using financial sentiment analysis, in which three machine learning algorithms were applied: LSTM, multilayer perceptron, and random forest classifier. The results revealed that LSTM extracted emotional data on the market more efficiently than could other machine learning algorithms. Mean–variance theory has a limited impact on practice because of estimation problems when applied to real data. Hence, Ban *et al.* [17] applied two machine learning concepts, regularization and cross-validation, for portfolio optimization. Branke *et al.* [18] applied a multiobjective evolutionary algorithm with a critical line algorithm to obtain a continuous Pareto front. Hachicha *et al.* [19] utilized fuzzy logic control and differential evolution techniques to explain the financial market for addressing the fuzzy portfolio selection problem. Macedo *et al.* [20] employed a genetic algorithm for portfolio selection and proposed the use and comparison of discovered technical analysis indicators for pursuing higher returns under certain risk levels. However, these methods cannot effectively address continuous decision problems such as an end-to-end portfolio management framework. Numerous recent studies have applied financial-model-free deep RL for portfolio optimization

problems [3], [4], [21], [22] because of its ability to address the continuous decision problem. Almahdi and Yang [23] extended the previous recurrent RL framework by using a risk-adjusted performance objective function and the expected MDD. Yu *et al.* [4] proposed a framework that combined three major components. Liang *et al.* [24] implemented fundamental RL algorithms, including deep deterministic policy gradient (PG), deterministic PG, and proximal policy optimization (PPO), for portfolio optimization; however, these algorithms still exhibited weak generalization ability in the testing stage.

Numerous studies have also focused on model generalization for RL and deep learning [25]. Reference [26] proposed a metacontroller to address the delayed reward during the decision process. Reference [27] proposed a comprehensive framework to solve the overfitting problem in RL. Reference [28] designed a regularization method to prevent deep Q-network overfitting. Moreover, [5], [6], [29], [30] proposed an adversarial learning framework for stock prediction and portfolio management.

The current literature review revealed that deep RL is the most suitable approach for portfolio optimization. The RL framework enables us to dynamically determine the allocation of assets at each moment by employing deep learning. This approach is also more likely to extract meaningful information from the original data without the necessity of making mathematical or model assumptions. Furthermore, RL makes it easier to address the continuous decision problem while considering the transaction fee. Therefore, for the aforementioned reasons, we applied deep RL to conduct our research.

III. METHODOLOGY

In this section, we provide a brief overview of the portfolio management system. A schematic of the system is provided in Fig. 1. Motivated by the Ensemble of Identical Independent Evaluators (EIIE) framework, the policy network is our agent that makes investment decisions at every time step; it also interacts with our environment—the financial market. However, instead of applying the deterministic PG as typical in the EIIE framework, we applied a stochastic approach by adopting three types of standard RL algorithms: the PG, actor–critic (AC), and PPO. More details are provided in Section III A–D.

The current portfolio work differs from the previous portfolio works in that its primary goal is to identify a generalized trading strategy given limited training time and multiple portfolio pools. The implication is that we could not adopt various solutions to reach all possible combinations during the training process. This task is extremely time-consuming and resource-intensive. Consequently, enabling the agent to learn the generalized trading strategy under such a diverse environment is a major challenge. Therefore, in Section III-F, we propose a weight-adjusted sampling method to determine which portfolios are worth learning for the current agent. In portfolio optimization, managers consider not only maximizing profits but also the risks of investors. However,

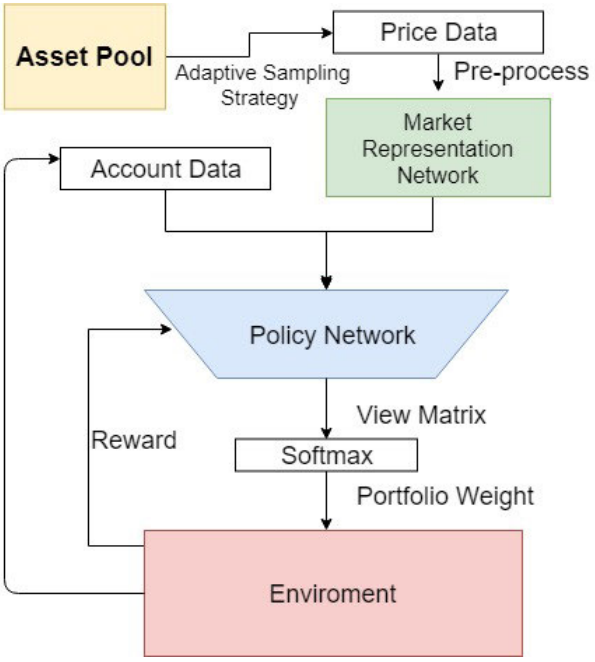


FIGURE 1. System overview.

we could not guarantee that our model would determine an appropriate balance between maximizing returns and minimizing risks. Thus, we designed a market representation network (Section III-G) to pretrain our model to acquire the possible risks. Finally, we integrated adversarial learning (Section III-H) into the state-of-the-art RL algorithm to enhance model robustness.

A. RL FRAMEWORK FOR PORTFOLIO OPTIMIZATION

RL has two major components: the agent and environment. In a typical portfolio optimization problem, the agent acts as an investor in the financial market. The mission of the agent is to obtain reliable profit by continuously determining the portfolio weight of m assets for interaction with the financial market environment. The environment is where the agent obtains information such as comments on the Internet, articles in the news media, or changes in stock prices. The information provided by the environment is called the "state." Crucially, the agent might only receive partial information because some of the information may be difficult for the agent to access. In addition to informing investors, the state also provides feedback on the actions of the agent. This feedback is called "reward," which is used to adjust the agent's behavior. The complete procedure is as follows: The agent determines the portfolio weight by observing the information. Subsequently, asset allocation performed by the agent may result in changes in the environment. The changes may also result in investor rewards. Reward in the portfolio problem may be any performance indicator, both risk and profit. In this manner, we apply RL to formulate the portfolio problem. The following sections describe the state, action, and reward in detail.

B. STATE

Our environment is composed of external and internal states. This distinction mainly depends on whether the action generated by our agent can influence the future environment. For example, when someone drives a car, the surroundings they observe comprise the state. Any action that the agent takes may have a considerable influence on the surroundings. However, in the financial market, we conduct back testing, meaning that actions cannot have any influence on historical price data. Such information that cannot be affected by the agent's actions is classified as the external state. Information that can be affected by the agent's decision is called the account state. In the current paper, we develop a trading strategy for high-frequency trading, and the trading frequency is 30 seconds. Short-term traders focus more on technical and chip factors and less on the news side; that is, they focus solely on the price changes. By analyzing changes in the financial market, they determine whether to long or short the asset to reach the purpose of portfolio management. Therefore, our simulation investors only pay attention to changes in asset prices and current asset allocation when determining their actions. The external state contains the historical price of each asset. The internal state contains the latest portfolio weight, the current total profit, and loss value.

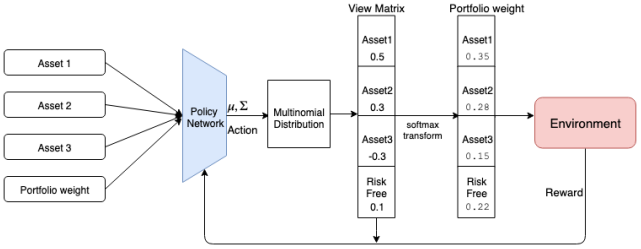


FIGURE 2. The design of action.

C. ACTION

The agent's duty in the portfolio framework is to determine the portfolio weight vector. The design of this action is demonstrated in Fig. 2. In our portfolio action space, we do not short assets; we do not allow investors to borrow assets and then return them in the future. We use a stochastic policy to design the action space. The action space can be divided into the discrete action space and continuous action space. For the discrete type, *softmax* is often used to represent each action's probability. Some portfolio works have also used a discrete action space to determine the portfolio weight vector.

However, in this design, a specific portfolio configuration is defined, which the agent uses to make decisions. Hence, a discrete action space cannot include all possible configurations. If the design of the action space is continuous, the action is typically sampled from the Gaussian distribution. If a continuous value for each asset can be produced, all the continuous values can be converted into a portfolio weight vector by using the *softmax* function.

All continuous values that are output can be regarded as the view matrix that predicts the potential growth of each asset in the future. The Gaussian distribution can only produce one continuous value. Therefore, we adopt the multinomial distribution to generate multiple continuous values. We assume that each asset is independent. Thus, the network only outputs the mu and diagonal sigma. For the output μ, we adapt the *tanh* as the activation function to control the value range from -1 to 1. For the output σ, we use *softplus* as the activation function.

Notably, after being transformed through *softmax*, the portfolio vector cannot be used to update the network. Instead, the portfolio vector is only used to interact with the environment to generate the portfolio return. We adapt the original view matrix to update the policy network in Fig. 2.

D. REWARD FUNCTION

Our reward design only considers profit; thus, we use the most straightforward approach to maximize the long-term reward. Each time step reward is r_t, which is expressed as the difference in portfolio value from time $t+1$ to time t. The target of the agent is to select a series of actions that maximize the accumulated reward over time. In long-term portfolio management, we pursue the highest final portfolio value p_f. The immediate reward for each time step t is the portfolio return, which is expressed as $r_t = p_{t+1} - p_t$. After considering the long-term reward with the discounted factor, we can express the cumulative return R_t as follows [31]:

$$R\left(s_t, a_t, \ldots, s_{t_f}, a_{t_f}, s_{t_f+1}\right) = \sum_{l=0}^{t_f+1} \left(\gamma^l r_{t+l}\right) \tag{1}$$

where γ is the discount factor. When considering the cumulative rewards in this stage, low portfolio profitability should not result in drastic actions because the profit at that time is affected by other decision points. Hence, we use the discount factor to reduce the influence of long-term reward.

E. POLICY NETWORK

1) POLICY GRADIENT

PG is used to boost the probability of the action if the action is likely result in a high expected reward. Conversely, the algorithm reduces the probability of the action if the action is likely to result in an unfavorable reward. The objective function is expressed as follows [31]:

$$\nabla_\theta J(\theta) \leftarrow E_\pi[\sum_t^n \nabla_\theta \log \pi_\theta(a_t|s_t) R_t)] \tag{2}$$

$$\leftarrow E_\pi[\sum_t^n \nabla_\theta \log \pi_\theta(a_t|s_t)(R_t - b(s_t))] \tag{3}$$

In Equation 3, we consider the baseline in our reward. The goal of this mechanism is to reduce the impact of variance by collecting numerous samples, which results in a more stable training process. In our problem setting, we directly identify the rewards at the same time and calculate the average value as our baseline.

2) ADVANTAGE AC

The advantage AC [32] is composed of a policy function π with parameter θ and an estimate of the value function $V(s)$ with parameter Θ. The policy function executes an action according to the state, and the value function estimates the average score in this stage by training another network. The advantage function involves determining the benefit to be gained from selecting the action compared with the average value. An estimation A_t of the discounted advantage function $A(s_t, a_t)$ is used to construct a policy estimator of the following form [33]:

$$\nabla_\theta J(\theta) \leftarrow E_\pi[\sum_t^n \nabla_\theta \log \pi_\theta(a_t|s_t)A_t)] \tag{4}$$

3) PROXIMAL POLICY OPTIMIZATION

Schulman*et al.* [34] proposed a new family of PG methods for RL called PPO. The essence of PPO is to transform the online training process in PG into an offline training process through importance sampling. Unlike the PG approach, PPO allows multiple epochs of minibatch updating, which leads to faster convergence. Different from the AC approach, PPO introduces a new objective function, which is called the clipped surrogate objective function for policy update. The objective function is expressed as follows:

$$L^{\text{Clip}}(\theta) = E_t[min(r_t(\theta)A_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t] \tag{5}$$

where A_t is advantage function and $r_t(\theta)$ denote the probability ratio between the new and old policies. This objective function limits the upper limit of the update amount and establishes a trust region around. The hyperparameter θ_{old} defines the range of maximum improvement.

F. ADAPTIVE SAMPLING STRATEGY

In this section, we describe our adaptive sampling strategy for selecting what type of portfolio is worth learning for the current agent. As mentioned in the previous section, we aim to optimize the general rules of trading; thus, we train only one portfolio. We sample numerous portfolios simultaneously and allow the agent to explore and obtain knowledge from those portfolios. The agent operates on these portfolios according to the current policy: some portfolios may perform well and others may not. The object of the comparison function can be any rule-based portfolio benchmark, and in our experiment, it is the constant rebalanced portfolio (CRP) strategy. We employ five stocks from the Dow Jones Industrial Index Average to construct a portfolio. If each portfolio has its own sampling weight, millions of weights would need to be updated. Because of limitations in computing power and memory problems, our system could not sample those portfolios simultaneously. Therefore, the sampling weights adjusted are not the weights selected for each portfolio but the weights assigned to each company. This approach results in two situations. The first is that the sampling weight of each company is the same on different trading days, and the second is that each trading day has its own sampling weights for the companies (Fig. 3). Our sampling strategy is based on the Adaboost algorithm. Adaboost is an improved boosting classification algorithm. It attempts to increase the weight of the samples that represent classification errors and linearly combines them with the first few classifiers; thus, each time the new classifier is trained, it focuses on the training the samples that are difficult to classify. Each weak classifier uses a weighted voting mechanism instead of the average voting mechanism, and only weak classifiers with higher accuracy have greater weight. The portfolio comprises numerous targets. If the portfolio is favorable, then each target is also classified as favorable according to the law of large numbers. In the following two sections, the two sampling adjustment strategies are discussed: using the same or different sampling weights for each company on different trading days.

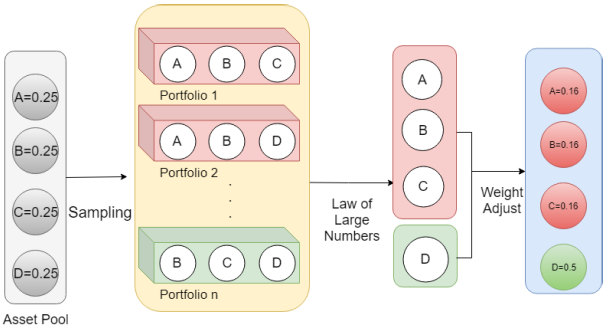


FIGURE 3. Sampling adjust schematic diagram.

1) SAMPLING WEIGHTS OF EACH COMPANY ARE THE SAME ON DIFFERENT TRADING DAYS

In Algorithm 1, suppose we have the current policy and company pool which contains m companies. Initially, the sampling weight vector for the companies is denoted as $\vec{w} = (w_1, w_2, \ldots, w_m) = \left(\frac{1}{m}, \frac{1}{m}, \frac{1}{m}, \ldots, \frac{1}{m}\right)$, a uniform distribution, which means that each company has an equal probability of being selected. The system first constructs numerous portfolios by selecting the company according to the sampling weights of each company, and it applies the policy to make decisions for those portfolios and to update the policy with the experience tuple. These steps are repeated numerous times until the policy converges. After the model converges, we simulate these portfolios and divide the companies into two categories according to the law of large numbers. The first group comprises the companies in which the agent performed well, and the second group consists of the companies in which the agent performed poorly. The performance of each company is denoted as $\vec{y} = (0,0,0,\ldots,0)$. The system assigns a value of 1 for the companies in which the agent exhibits poor performance. We calculate the ϵ rate, which is used to recalculate the sampling weights of the companies. When the agent constructs the portfolios according to the new sampling weights, the agent may perform half of the portfolio well and half of the portfolio poorly, where $\epsilon = \sum_{i=1}^{m} w_i y_i$. Subsequently, we update the sampling weights of

each company by using different equations. For the companies in which the agent performed poorly, the following equation is used to update the sampling weights: $w_i = w_i \exp(a)$; for the companies in which the agent performed well, the following equation is used to update the sampling weights $w_i = w_i \exp(-a)$. In the preceding equations $a = \ln\sqrt{\frac{1-\epsilon}{\epsilon}}$.

In sum, we first initialize the sampling weights and construct numerous portfolios on the basis of these sampling weights. After determining the performance of the portfolios, the system identifies which companies the agent managed well and which it managed poorly; that information is used to calculated the ϵ rate. Finally, the system uses the ϵ rate to adjust the sampling weight of each company. We summarize our adaptive sampling strategy in Algorithm 1.

Algorithm 1 Sampling Weight Adjust Algorithm (1)

Input: Given company number m and sampling weight vector $\vec{w}$ for each company, initial the sampling weights of each company by $w_i = 1/m$, for $i = 1,\ldots,m$

Choose a number of portfolio according to the current sampling weights vector $\vec{w}$
Stimulate the portfolios and get each portfolio performances
Initial $\vec{y} = (y_0, y_1, y_2, \ldots, y_m) = (0, 0, 0, \ldots, 0)$
Set element y_b of vector $\vec{y}$ to 1, for $b \in$ those company that the agent perform not well based on the Law of large number
Calculate $\epsilon = \sum_{i=1}^{m} w_i y_i$
for n $\leftarrow$ 1 to m **do**:
  if $y_m == 1$ **then**:
    $w_m = w_m \exp(a)$, where $a = \ln\sqrt{\frac{1-\epsilon}{\epsilon}}$
  else
    $w_m = w_m \exp(-a)$, where $a = \ln\sqrt{\frac{1-\epsilon}{\epsilon}}$

2) SAMPLING WEIGHTS OF EACH COMPANY DIFFERS ON DIFFERENT TRADING DAYS

In this section, we apply different sampling weights on different trading days. According to Algorithm 2, we have n trading days and a company pool that contains m assets.

Hence, each trading day has its own sampling weight vector $\vec{w}_n = (w_0, w_1, \ldots, w_m)$. In the stage of forming a portfolio, our system first selects a trading day according to the uniform distribution; the system then forms a portfolio on the basis of the sampling weight vector for that day. After repeating the aforementioned steps, a variety of portfolios are obtained, and the policy can be applied to make a decision for those portfolios and update the policy with the experience tuple. We repeat these steps numerous times until the policy converges. After the model converges, each day has its performance vector $\vec{y}_n = (y_0, y_1, \ldots, y_m) = 0$. Hence, we must obtain the performance vector separately. After obtaining the performance vector $\vec{y}$ we must calculate each day's epsilon ϵ_n rate and then use it to update each day's sampling weight vector. We summarize our adaptive sampling strategy in Algorithm 2.

Algorithm 2 Sampling Weight Adjust Algorithm (2)

Input: Given m company number, n trading days, initial the sampling weight's matrix $w_{n,m} = 1/m$, which is the $n \times m$ matrix.

repeat
  Select a trading day n and choose a number of portfolio according to current company weight w_n
until Select enough portfolio
Stimulate the portfolios and get each portfolio performances
Initial the performance matrix $y_{n,m} = 0$, which is the $n \times m$ matrix
for day $\leftarrow$ 1 to n **do**:
  Set element $y_{day,b}$ of matrix y to 1, for $b \in$ those company perform not well in that day based on the Law of large number
  Calculate $\epsilon = \sum_{i=1}^{m} w_{day,i} y_{day,i}$
  for n $\leftarrow$ 1 to m **do**:
    if $y_{day,n} == 1$ **then**:
      $w_{day,n} = w_{day,n} \exp(a)$, where $a = \ln\sqrt{\frac{1-\epsilon}{\epsilon}}$
    else
      $w_{day,n} = w_{day,n} \exp(-a)$, where $a = \ln\sqrt{\frac{1-\epsilon}{\epsilon}}$

G. MARKET REPRESENTATION NETWORK

As noted in the previous section, the portfolio optimization process is divided into two phases. The first step is to predict each asset's future trend, and the second phase is to rebalance each asset according the trend exceptions and the current portfolio allocation. To enable the agent to rapidly acquire skills in predicting potential risk and future trends, we use the market representation network to model the relationship between current market price trends and future market price trends. First, we train an autoencoder network to determine the latent space of stock price streams. The autoencoder consists of two parts: the encoder layer and the decoder layer, and the purpose of the latent space is to extract crucial information on stock price relations among various stocks (see step 1 in Fig. 4) and future price trends. Finally, we can calculate VAR according to the future price. Thus, we train a network, the embedding layer, that allows the model to accurately predict the future price's latent space and VAR on the basis of the predicted future values (see step 2 in Fig. 4).

VAR is the maximum possible loss of a financial asset or portfolio value over a specific period at a certain probability level. After designing the embedded network, we integrate our embedded network with our policy network.

Some changes are necessary because of the integration of the embedded network with the policy network. In the original design, each asset fits directly into the policy network through the fully connected neural network. When the

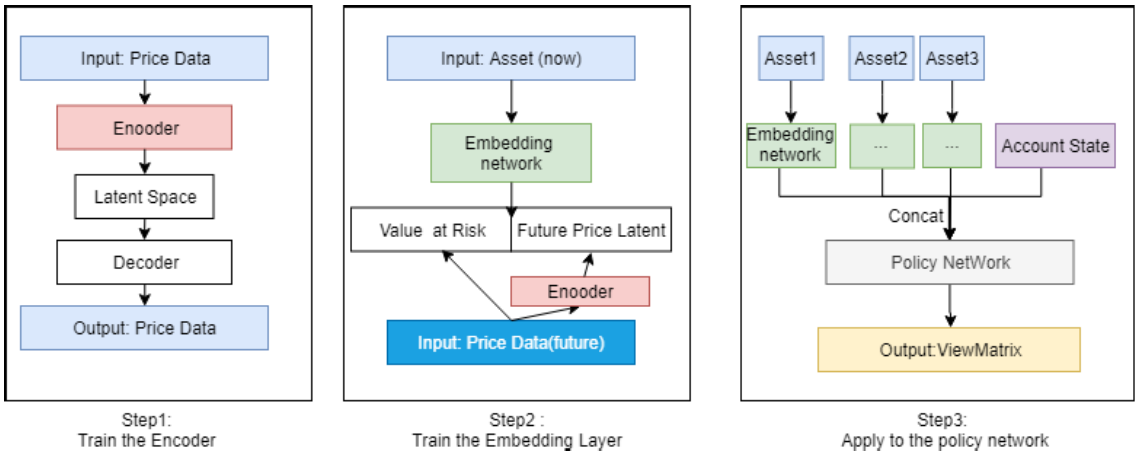


FIGURE 4. Market representation network.

embedded network is integrated into our policy network, each asset fits separately into the embedded network; thus, the parameters of the embedded network are shared. The embedded network outputs each asset's latent space. Thus, we concatenate each asset's latent space and use it to fit the policy network through the fully connected neural network. The aforementioned process is how our market representative network fits into our policy network. Two points should be considered here: whether the parameters of the embedded network should be updated and when the policy network should be updated. The question of whether the parameters of the embedded network should be updated is similar to the problem faced during the initial preparation of the policy network. If the parameters of the embedded network are fixed, fewer parameters can be updated; thus, the ability to fit the portfolio may be limited.

H. ADVERSARIAL LEARNING

Deep RL has the following advantages: it can capture nonlinear features, has low prior assumptions, and is similar to human financial investments. A study [35] that highlighted the high performance of RL plus adversarial training in mountain car and hopper environment applications served as inspiration for our approach. Because financial data have high-noise and nonstationary characteristics, we employ adversarial learning to make the model more robust. Adversarial learning adds noise to data to enhance model robustness. More specifically, adversarial learning uses incorrect information to fool the agent. As a result of the incorrect information, the agent performs relatively uncommon actions. Google DeepMind and Open AI have proven that adding noise to network parameters helps the algorithms explore their environments more effectively.

In our work, adversarial learning is implemented by adding small perturbation signals to the input features when the model is learning from experience. In the subsequent experiments, our price perturbation will be restricted at most 1 % of the original vibration amplitude:

$$s_t\prime = s_t \cdot (1 + \epsilon) \tag{6}$$

The perturbation rate epsilon is sampled from [−0.01, 0.01]. Before each state st was fed into the policy network, a sampled noise is added into the observed historical price data. We expect the agent will make action decisions according to the state $st\prime$ under perturbation. It can help to train agents more robustly and let the decision making focuses on real market trends than random noise of financial markets. However, while updating the RL agent, we calculated the policy gradient according to real state s_t. Detailed algorithm was illustrated in Algorithm 3.

By incorporating adversarial training, the versatility and robustness of the model will increase, and the agent would be less sensitive about noise disturbance of the stock price change. Therefore, undesirable transaction costs caused by frequent re-balancing will not reduce the final portfolio returns.

Algorithm 3 Adversarial Learning

Input: initialize θ as the parameters of policy $\pi(s, a\,|\theta)$
for $t \leftarrow 1$ to T **do**:
  Receive state s_t
  Add noise into the price data, the state become s'_t
  Execute action a_t According to s'_t and get s_{t+1}
$R_t \leftarrow$ total discounted rewards from t to T
Update the policy $\theta \leftarrow \theta + \eta\nabla_\theta \log \pi(s_t, a_t|\theta)R_t$

IV. EXPERIMENTS

We used stock price data from Wharton Research Data Services (WRDS) and conducted four experiments. We used the Trade and Quote database, which contains intraday transaction data for all securities posted on the New York Stock Exchange, American Stock Exchange, and Nasdaq National and Market System.

A. EXPERIMENTAL DATASETS

We collected the data of constituent stocks in the Dow Jones Industrial Average because this index is regarded as an accurate representation of the US stock market. Furthermore, the companies in the Dow Jones have a particular influence and popularity in their domain according to the Global Industry Classification Standard. Data were extracted every second and were collected from January 3, 2017, to December 29, 2017, for a total of 251 trading days. Daily trading hours were from 9:30 AM to 4:00 PM.

B. DATA PROCESSING

After collecting the transactions data, we resampled the data frequency into 30-second blocks and converted the data into open-high-low-close (OHLC) charts. OHLC charts express a stock's opening, high, low, and closing prices for a specific period, thus describing the status of the market during a specific period. Some individual stocks had missing data on a specific trading day because of trading interruptions for major events or insufficient information disclosure. We filled the missing price data (resulting from weekends and holidays) with the latest closing price to maintain time series consistency. Crucially, for standardization of data preprocessing, this window cannot contain future information. Hence, to obtain a customary agent that exhibits robust performance with various stocks, we subtracted the opening price, closing price, high price, and low price from the opening price at the beginning of the period. To improve computing efficiency, we applied the moving average (MA) line to represent past trends. In the following experiments, we used a 5-, 20-, 30-, 60-, 120-, 900-, 1800-, 3600-, 7200-, and 10800-second MA. The preprocessing procedure for MA indicators is the same as that for the OHLC price.

C. EXPERIMENT TRADING SETTING

This section includes a description of the proposed system parameters for experimental settings. The first parameter, the asset number m is set to 5, which means that 5 individual assets should be picked and allocated to build the portfolio. The parameter was determined empirically by considering the trade-offs between the large unsystematic risk (i.e. large portfolio volatility) caused by a small portfolio size and the large rebalancing costs caused by a large group of trading targets. For the optimization of the complete domain, we collected numerous portfolios when we updated our models. The number of portfolios is N which was set to 500. Since our agent is a stochastic policy, meaning it has many variances, we repeated each portfolio numerous times to reach a more accurate expected value.

After selecting the portfolios, we have P_n repetitions to collect the performance of the agent; we have P_n identical workers to gather experience. Since the interaction between the agent and the market is stochastic, the experience gathered by each worker would be different. Given our agent may not be able to master the selected portfolios within only one update, the number of update times U_t represents the process where an agent collects experience and updates the model. After U_t updates, the portfolios are re-selected. Both U_t and P_n are hyperparameters, and they are set to 10 in our system. Detailed learning Process of trading Agent is displayed in Algorithm 4.

TABLE 1. Hyperparameters settings.

Hyperparameters	notation	value
asset number	m	5
portfolio number	N	500
repeat number	P_n	10
update times	U_t	10
max trajectory number	$Max(\tau)$	10,000
initial cash	None	1,000,000
transaction cost	None	0.025%
learning rate	η	1e-6
discounting factor	γ	0.99

Algorithm 4 Reinforcement Learning Process of Trading Agent

Input: initialize θ as the parameters of policy $\pi(s, a\,|\theta)$

Input: Maximum trajectory number$Max(\tau)$, Current trajectory number $Current(\tau) = 0$, update times U_T, repeat number P_n

```
while Current(τ) < Max(τ) do
    randomly choose N groups portfolios and each portfolio contain m assets.
    for number ← 1 to U_t do
        repeat those portfolios P_n times
        get trajectories (s_{1,1}, a_{1,1}, r_{1,1}, ..., s_{N,T}, a_{N,T}, r_{N,T})
        under the policy π
        for time t ← 1 to T do
            R_t ← calculate total discounted rewards from t
            to episode end T with discount factor γ
        Current(τ) ← Current(τ) + N × P_n
        Update the policy network with experience tuple
```

After the manipulation mentioned above, we expect our agent can learn a general trading strategy within a limited trading time. Thus, the maximum trajectory $Max(\tau)$ was set to 1 million. The rest of the parameters of our simulations are as follows: The amount of cash owed by the investor was 100,000. We averaged the assets in each target in the initial state. The transaction cost was set to 25 %. In the subsequent experimental settings, the comparison was based on these settings. Our actor's learning rate was set to 0.000001, and the batch size was set to 1024. The reward function included the discount factor, which was set to 0.99. Table 1 summarizes our hyperparameter settings.

Before introducing the experiment, we briefly describe the distinction between training and testing. Because our portfolio consists of hundreds of possibilities, the agent cannot explore all options with RL in a short period. If the agent achieves excellent performance in the training interval, we believe the agent is sufficiently powerful to manage

TABLE 2. Sharpe and profit for the standard portfolio optimization's algorithm in 2017.

	Sharpe						Profit					
Test set	PG	AC	PPO	EIIE CNN	CRP	UBH	PG	AC	PPO	EIIE CNN	CRP	UBH
0	**0.31**	0.12	0.1	0.19	0.17	0.12	**831.66**	283.97	226.54	422.4	351.82	255.35
1	**0.49**	0.32	0.32	0.38	0.31	0.32	**1098.7**	762.88	674.39	789.04	619.73	649.02
2	-0.01	0.06	0.02	**0.11**	0.09	0.02	-36.37	159.92	44.42	**258.97**	215.89	43.98
3	0.41	0.31	0.28	0.34	**0.36**	0.29	**851.29**	568.59	474.55	556.97	586.12	494.51
4	**0.47**	0.41	0.46	**0.47**	0.44	0.44	**1032.65**	795.07	896.66	968.6	854.36	885.74
5	0.22	0.25	0.29	**0.33**	0.26	0.29	592.27	507.21	**594.69**	703.86	501.73	581.5
6	**0.23**	**0.23**	0.18	0.23	0.2	0.16	**780.88**	500.95	363.92	495.8	418.34	327.24
7	0.11	**0.16**	0.07	0.12	0.13	0.06	**348.09**	373.39	152.75	251.27	255.49	133.36
8	0.02	0.08	0.05	**0.1**	0.05	0.03	59.43	**176.89**	114.44	245.3	123.37	77.05
9	0.01	**0.19**	0.06	0.11	0.09	0.05	15.72	**304.91**	95.67	190.55	142.39	89.88
10	-0.13	**0.08**	0.05	0.07	0.03	0.04	-356.85	**200.56**	103.8	152.16	55.34	90.49
11	-0.03	**0.29**	0.19	0.19	0.19	0.17	-71.9	**614.82**	353.5	373.18	345.62	317.47
12	**0.16**	0.02	0.03	0.08	0.08	0.03	**427.26**	50.17	50.44	166.85	150.53	65.25
13	-0.04	**0.11**	0.01	0.02	-0.02	-0.02	-124.8	**216.58**	17.62	35.95	-34.55	-35.59
14	**0.19**	0.15	0.04	0.12	0.09	0.04	**511.27**	305.56	85.5	245.54	160.31	76.32
15	**0.37**	0.04	0.04	0.14	0.07	0.07	**937.39**	95.66	121.23	332.55	176.8	178.23
16	-0.16	**0.0**	-0.1	-0.05	-0.09	-0.11	-450.05	**7.53**	-199.68	-89.18	-174.3	-203.3
17	0.04	**0.2**	0.14	0.14	0.17	0.1	130.89	**406.38**	303.36	313.37	344.4	223.15
18	0.14	**0.23**	0.12	0.14	0.09	0.08	387.72	**497.37**	221.74	293.47	176.15	159.42
19	**0.06**	**0.06**	-0.01	0.05	0.04	-0.01	**182.26**	120.24	-23.86	103.8	78.74	-20.47
20	0.29	**0.34**	0.24	0.27	0.27	0.22	**723.57**	640.02	421.95	471.99	448.32	385.22
21	0.2	0.27	0.22	**0.28**	0.22	0.23	539.78	668.33	537.08	**685.38**	515.61	549.22
22	-0.06	**0.16**	0.02	0.07	0.05	-0.01	-244.24	**372.15**	57.86	181.2	132.23	-14.36
23	0.0	**0.11**	-0.03	0.07	0.01	-0.05	13.67	**210.5**	-52.38	147.36	27.53	-98.44
24	0.07	**0.16**	0.06	0.08	0.03	0.04	215.74	**344.11**	134.44	170.61	66.79	86.93
25	0.02	**0.21**	0.05	0.15	0.09	0.05	51.1	**355.45**	99.66	297.14	162.81	94.48
26	-0.06	**0.15**	0.02	0.05	0.02	-0.01	-244.45	**349.79**	46.25	133.63	46.89	-14.63
27	0.07	0.17	0.13	**0.18**	0.13	0.12	223.59	**341.91**	243.03	335.09	244.95	235.82
28	-0.03	**0.18**	0.03	0.12	0.05	0.0	-87.76	**337.65**	64.78	242.93	113.08	0.92
29	**0.14**	0.0	-0.0	0.08	0.09	0.02	**337.01**	3.88	-10.56	163.69	197.48	33.36
Average	0.12	**0.17**	0.1	0.15	0.12	0.09	289.18	**352.41**	207.13	321.32	243.47	188.24

any portfolio optimization problem even if the portfolio is selected in the training interval. Therefore, our testing period is the same as the training period. We randomly select 30 portfolios, and each portfolio consists of five assets traded for 60 trading days from January 3 to March 27, 2017. In the following experiments, we use the aforementioned settings to compare our results.

D. PORTFOLIO PERFORMANCE MEASUREMENT

We use the following indicators to measure portfolio measurement performance. The first indicator, which is of great importance to most people, is the profit value. In addition to the investment return, the risks that investors are exposed to during investment should also be considered. Therefore, the portfolio can be measured by indicators of profit and risk. In the following section, we describe the common risk indicators when measuring portfolio performance. The Sharpe ratio is widely used to measure the performance of an investment by considering its risk; it is defined as follows:

$$Sharpe = \frac{\mathbb{E}_t[\rho_t - \rho_f]}{\sqrt{var(\rho_t - \rho_f)}} \tag{7}$$

where ρ_t is the periodic return, and ρ_f is the risk-free rate of return. A higher Sharpe ratio indicates higher returns earned per unit of risk. Profit factor is the ratio of the net profit versus the net loss, and is defined as follows:

$$ProfitFactor = \frac{GrossProfit}{GrossLost} \tag{8}$$

Profit factor refers to how much an investor can gain based on the risk of losing 1 dollar; thus, a higher value is desirable. The last risk indicator is MDD, which is the maximum loss from the peak of a portfolio to its trough before a new peak is attained. MDD is an indicator of downside risk over a specified time period. MDD is expressed in percentage terms and computed as follows:

$$MDD = \max_{\tau \in (0,T)} \left\{ \max_{t \in (0,\tau)} \frac{p(t) - p(\tau)}{p(t)} \right\} \tag{9}$$

where time T is the maximum of the drawdown over the history of the variable $p(t)$ which refers to the closing price at time t and $T > \tau$. In the first experiment, we compare our reinforcement algorithm with two portfolio benchmarks and EIIE topology. The first portfolio benchmark, uniform buy and hold (UBH), involves investing in each asset evenly at the beginning and retaining the portfolio until the end of the trading session. The second benchmark, uniform CRP, is a passive strategy that uniformly rebalances the proportion of assets at each moment. The EIIE convolutional neural network (CNN) method, which was discussed in Section III, applies a deterministic policy. We implement the AC, PG, and PPO reinforcement algorithms. All of our policies were stochastic. The performance outcomes are

TABLE 3. Profit factor and maximum drawdown for the standard portfolio optimization's algorithm in 2017.

	Profit Factor						Maximum DrawDown					
Test set	PG	AC	PPO	EIIE CNN	CRP	UBH	PG	AC	PPO	EIIE CNN	CRP	UBH
0	**2.3**	1.35	1.31	1.69	1.59	1.38	244.23	280.17	310.24	**232.1**	255.17	276.8
1	**4.3**	2.31	2.54	3.04	2.32	2.47	**103.93**	152.82	115.33	109.79	121.08	117.72
2	0.97	1.18	1.05	1.33	**1.27**	1.05	392.57	458.26	457.64	**333.01**	355.72	428.42
3	**2.65**	2.28	2.04	2.34	2.41	2.1	**147.45**	175.79	186.48	156.7	160.87	183.14
4	**3.34**	2.94	3.23	3.37	3.13	3.12	**83.09**	137.19	125.79	116.2	127.35	129.1
5	1.86	1.95	2.18	**2.45**	1.97	2.16	181.7	156.92	149.71	**145.27**	176.56	153.85
6	1.75	1.78	1.55	**1.8**	1.67	1.47	275.36	153.12	218.05	**151.61**	151.52	222.45
7	1.35	**1.53**	1.2	1.36	1.37	1.18	358.4	301.67	280.09	**237.64**	249.91	252.36
8	1.06	1.24	1.15	**1.33**	1.16	1.1	371.73	**186.79**	285.68	294.83	303.59	315.81
9	1.02	**1.61**	1.16	1.33	1.24	1.15	341.96	**176.56**	201.74	179.57	196.15	190.08
10	0.73	**1.23**	1.14	1.21	1.07	1.12	529.17	261.81	**240.87**	249.92	248.89	255.23
11	0.93	**2.09**	1.57	1.61	1.61	1.5	293.51	149.04	180.48	170.58	**120.55**	182.53
12	**1.57**	1.06	1.07	1.23	1.21	1.09	**255.01**	364.58	347.66	334.11	287.79	348.18
13	0.89	**1.31**	1.02	1.05	0.95	0.95	481.32	**223.05**	312.07	317.57	342.92	333.19
14	**1.59**	1.46	1.12	1.37	1.26	1.1	322.71	256.33	314.06	**251.56**	276.84	309.58
15	2.7	1.1	1.13	**1.42**	1.19	1.2	**166.1**	287.33	275.45	203.55	227.69	237.59
16	0.67	**1.01**	0.78	0.89	0.79	0.77	579.75	257.6	302.86	**214.0**	267.58	302.27
17	1.12	**1.63**	1.45	1.46	1.54	1.31	403.83	**226.56**	258.45	267.64	242.11	300.31
18	1.43	**1.79**	1.34	1.45	1.26	1.23	382.08	246.67	**245.61**	258.01	269.07	268.16
19	**1.18**	1.16	0.97	1.15	1.11	0.97	384.23	**208.71**	272.94	221.46	254.45	270.41
20	2.2	**2.34**	1.85	2.01	1.93	1.75	189.32	195.02	158.23	163.25	**141.39**	174.22
21	1.68	2.04	1.83	**2.17**	1.78	1.87	398.03	311.85	347.4	**306.43**	342.55	330.82
22	0.86	1.52	1.07	**1.21**	1.16	0.98	550.37	377.6	327.93	**270.36**	302.38	342.07
23	1.01	**1.35**	0.93	1.2	1.04	0.88	323.95	212.82	262.29	**211.04**	230.33	288.4
24	**1.21**	1.5	1.17	**1.21**	1.08	1.1	408.12	296.0	314.78	**278.23**	268.67	307.81
25	1.05	**1.7**	1.14	1.48	1.25	1.13	528.53	**173.63**	292.96	238.41	237.66	291.66
26	0.86	**1.44**	1.05	1.15	1.05	0.98	512.53	233.36	264.81	**236.52**	229.06	276.32
27	1.19	**1.55**	1.38	1.54	1.38	1.36	318.93	191.02	185.11	**174.11**	185.09	192.33
28	0.92	**1.58**	1.08	1.34	1.15	1.0	314.39	**143.95**	256.47	221.91	216.89	282.33
29	**1.46**	1.01	0.99	1.22	1.28	1.04	340.96	450.51	492.66	410.43	**339.12**	456.16
Average	1.53	**1.6**	1.38	1.58	1.44	1.35	339.44	241.56	266.13	**231.86**	237.63	267.31

recorded in Tables 2 and 3. The rows in the tables represent 30 portfolios, and each column represents a distinct method for portfolio optimization. We use four technical indicators to measure the performance of the trading strategy. To facilitate our presentation, we divide the four indicators into two tables. The first table lists the Sharpe ratio and net profit for each portfolio, and the second table presents the profit factor and MDD. Thus, Table 2 presents the performance of the first portfolio in various portfolio solutions. The optimal performance (i.e., the largest Sharpe ratio, profit, and profit factor, and the lowest value MDD) in each row is presented in boldface.

1) SHARPE RATIO

The proposed method outperforms the baseline methods, CRP and UBH, in terms of the Sharpe ratio (Table 2). Moreover, to evaluate the robustness of our model, we calculated the number of times each model outperforms the portfolio baseline CRP. The results indicate that our RL algorithms surpass the baseline CRP under most circumstances in terms of the Sharpe ratio. When we scrutinize the performance of various RL algorithms; overall, the AC model has the highest Sharpe ratio, followed by the EIIE CNN; and the baseline UBH has the lowest score. This outcome is reasonable because, unlike the other policy-based methods (EIIE CNN and PG), AC adopts a critic network and thus can perform robustly.

2) NET PROFIT

Next, in terms of net profit, our proposed RL-based models also outperform the baseline methods. However, among the RL-based algorithms, PG displays relatively unstable performance: although it often outperforms other methods, it also frequently has lower performance than other methods. By contrast, AC displays both robustness and stability throughout various test cases. Finally, of all the PPO RL-based methods, PPO exhibits the lowest performance.

3) PROFIT FACTOR

Regarding the profit factor, the RL-based methods still exhibit robust performance in comparison with the baseline methods. AC again exhibits optimal performance, followed by EIIE CNN; PPO again exhibits a relatively poor result.

4) MAXIMUM DRAWDOWN

For MDD, the RL-based methods outperform the baseline methods. EIIE CNN exhibits the highest performance, and PPO and PG have relatively poor performance, indicating that these two methods are less adept at controlling risks.

Overall, RL-based methods perform more favorably than the baseline methods. Specifically, EIIE CNN and AC exhibit the highest performance among all the evaluation methods. However, the general RL methods (RL-based methods) are clearly deficient in terms of profit stability and risk control. For example, although the overall average of the PG model

TABLE 4. Sharpe and profit for the training technique in 2017.

	Sharpe				Profit			
Test set	PG_orginal	PG_adversarial	PG_RP	PG_RPadv	PG_orginal	PG_adversarial	PG_RP	PG_RPadv
0	**0.305**	0.328	0.181	0.298	831.656	**899.622**	416.384	691.673
1	0.491	**0.558**	0.519	0.419	1098.703	**1238.533**	1200.194	1006.923
2	-0.01	0.019	0.088	**0.137**	-36.371	71.827	257.437	**385.475**
3	0.412	0.421	0.315	**0.435**	**851.286**	776.751	572.844	796.124
4	0.472	**0.565**	0.431	0.411	1032.646	**1269.411**	990.164	950.788
5	0.224	0.4	**0.345**	0.333	592.266	**999.832**	820.701	809.653
6	0.234	0.316	0.362	**0.396**	780.879	884.726	789.761	**1050.535**
7	0.11	0.187	0.195	**0.225**	348.088	**582.106**	484.627	528.099
8	0.021	0.132	0.165	**0.206**	59.431	389.813	419.172	**519.285**
9	0.007	0.014	**0.186**	0.156	15.718	33.44	**349.454**	321.586
10	-0.125	-0.047	0.03	**0.12**	-356.846	-124.248	75.725	**308.701**
11	-0.03	0.02	0.17	**0.249**	-71.9	47.473	424.005	**633.822**
12	0.163	**0.299**	0.16	0.117	427.258	**758.913**	340.095	279.409
13	-0.044	**0.069**	-0.061	0.068	-124.797	**198.019**	-127.372	168.831
14	0.19	**0.265**	0.188	0.256	511.274	**700.545**	420.924	581.654
15	**0.365**	0.39	0.138	0.167	937.389	**994.406**	349.739	465.375
16	-0.159	-0.108	0.034	**0.065**	-450.047	-282.296	77.59	**127.824**
17	0.042	0.153	0.208	**0.283**	130.887	483.98	446.874	**627.299**
18	0.142	0.223	0.214	**0.272**	387.716	**591.928**	455.168	588.321
19	0.063	0.154	**0.156**	0.129	182.255	**421.669**	339.223	294.376
20	0.294	**0.319**	0.208	0.257	723.57	**804.87**	461.728	539.723
21	0.2	0.256	0.308	**0.395**	539.776	731.679	757.103	**992.545**
22	-0.057	-0.018	0.177	**0.196**	-244.243	-75.021	482.057	**515.353**
23	0.004	0.042	**0.189**	0.13	13.674	145.453	**428.401**	332.17
24	0.072	0.144	0.049	**0.185**	215.743	419.681	134.142	**409.674**
25	0.016	0.033	0.138	**0.207**	51.102	110.831	306.896	**426.969**
26	-0.058	-0.014	**0.143**	0.15	-244.455	-57.764	387.828	**437.601**
27	0.071	0.089	**0.238**	0.193	223.593	242.503	**499.02**	437.728
28	-0.03	0.034	**0.189**	0.163	-87.765	110.976	**459.175**	354.747
29	0.135	**0.191**	0.18	0.092	337.011	**517.232**	432.552	202.333
Average	0.117	0.181	0.195	**0.224**	289.183	462.896	448.387	**526.153**

is higher than that of the CRP approach, the PG model only obtains high returns for specific portfolio assets but suffers losses in multiple portfolio targets. Therefore, further techniques for controlling risks and enhancing the generalizability of these models are required.

E. PERFORMANCE OF THE STANDARD PORTFOLIO SOLUTIONS

We incorporate adversarial learning and a market representation network into our framework. We use adversarial learning to add noise to the price data during our agent's exploration phase. The noise added ranges from -1 % to 1 % of the original value. The purpose of this price perturbation is to enhance the robustness of reinforcement learning and avoid overfitting. In addition, market representation network is applied to provide sufficient information of incoming trends and potential risk to the agent. This network is trained with the historical price data of the component companies of Dow Jones industrial average index from 2016; this period does not overlap with the training period of the portfolio agent in reinforcement learning. The performance of the models is presented in Tables 4 and 5..

The performance of the models is presented in Tables 4 and 5. In this experiment, instead of performing comparisons with the portfolio baseline methods, we compare the models with the original RL-based algorithms. We test four models: (1) the original PG indicated in the previous experiment, (2) a model trained with PG and adversarial learning, (3) a model trained with PG and the market representation network, and (4) a model trained using PG plus the aforementioned two techniques (i.e., adversarial learning and market representation). Each row in the aforementioned tables represents the performance of a particular portfolio over 60 trading days, and each column represents one of the model with specific training skills. *RP* represents the use of the market presentation network, and *adv* refers to the application of adversarial learning. *RPadv* denotes the application of both techniques.

1) SHARPE RATIO

Overall, the modified *PG_RPadv* outperforms the other methods, including the baseline *PG_original*. In the 30 portfolio tests, the adv model outperforms the original model; the Sharpe ratio of the *RP* and *RPadv* models is 21 and 23 times that of the original models, respectively. As long as the models that applied the market representation network have a higher Sharpe ratio, the *RP* and *RPadv* models perform well. The results reveal that the method is effective for reducing risks. Notably, the performance of the adversarial learning model improves considerably in terms of the Sharpe ratio; however, its ability for risk control is less than the market representation network *PG_RP*.

TABLE 5. Profit factor and maximum drawdown for the training technique in 2017.

	Profit Factor				Maximum DrawDown			
Test set	PG_orginal	PG_adversarial	PG_RP	PG_RPadv	PG_orginal	PG_adversarial	PG_RP	PG_RPadv
0	2.299	**2.399**	1.602	2.298	244.231	**226.799**	263.708	247.276
1	4.298	**5.238**	4.051	3.194	103.933	**103.397**	104.239	115.899
2	0.973	1.052	1.24	**1.426**	392.57	**310.431**	330.702	415.567
3	2.649	2.77	2.206	**3.062**	147.451	**119.759**	164.874	131.552
4	3.343	**4.538**	2.939	2.888	83.092	**79.009**	119.674	109.354
5	1.855	**2.972**	2.469	2.385	181.695	**148.726**	201.411	200.617
6	1.752	2.201	2.508	**2.712**	275.362	203.977	**135.961**	171.147
7	1.349	1.683	1.608	**1.799**	358.397	272.407	318.636	**216.818**
8	1.057	1.416	1.566	**1.717**	371.732	334.733	370.084	**179.217**
9	1.017	1.035	**1.603**	1.478	341.96	319.574	157.145	**134.276**
10	0.732	0.893	1.081	**1.366**	529.166	337.502	296.768	**185.244**
11	0.928	1.051	1.525	**1.86**	293.51	259.538	**183.85**	209.941
12	1.57	**2.226**	1.509	1.347	255.01	**220.594**	252.893	380.032
13	0.889	**1.197**	0.844	1.195	481.321	307.313	327.867	**304.412**
14	1.59	1.895	1.615	**1.923**	322.71	231.243	**156.999**	183.306
15	2.698	**2.805**	1.412	1.512	166.101	**148.043**	192.155	223.57
16	0.674	0.768	1.091	**1.181**	579.747	425.338	**217.456**	169.701
17	1.12	1.537	1.684	**2.07**	403.828	348.967	**179.661**	196.798
18	1.431	1.78	1.7	**2.008**	382.084	291.946	**218.181**	247.062
19	1.177	**1.538**	1.514	1.435	384.231	310.735	168.919	**162.57**
20	2.199	**2.421**	1.705	1.965	189.322	**161.652**	168.608	266.135
21	1.679	1.963	2.189	**2.78**	398.028	354.519	245.566	**188.684**
22	0.859	0.955	1.574	**1.693**	550.37	442.254	196.259	**154.175**
23	1.012	1.118	**1.641**	1.382	323.946	304.648	**193.138**	264.13
24	1.205	1.479	1.136	**1.655**	408.12	331.234	346.383	**199.957**
25	1.046	1.097	1.43	**1.65**	528.528	568.762	**219.995**	221.961
26	0.857	0.965	**1.438**	**1.438**	512.534	364.572	**194.595**	222.864
27	1.185	1.243	**1.809**	1.666	318.926	270.52	180.181	**133.13**
28	0.924	1.092	**1.595**	1.519	314.385	277.767	274.884	**228.271**
29	1.457	**1.721**	1.625	1.271	340.961	364.32	**282.901**	415.812
Average	1.527	1.835	1.73	**1.862**	339.442	281.343	222.123	**215.983**

2) NET PROFIT

In terms of net profits, the *RPadv* model again has the most favorable results of all the models. According to the net profit statistics, the performance of the *adv*, *RP*, and *RPadv* models is 29, 21, and 22 times higher than that of the original models, respectively. Notably, although *PG_adv* sometimes exhibits favorable net profit results, *PG_RPadv* still markedly outperforms *PG_adv* on average because it is more stable. This finding indicates that controlling risks and enhancing generalizability are essential for earning long-lasting net profits.

3) MAXIMUM DRAWDOWN

Regarding the average loss of investors, the *RPadv* model has the lowest average MDD of all the models, which is consistent with our initial assumptions. The market representation network PG_RP has higher performance in preventing capital loss than the original model and the adversarial learning model.

4) PROFIT FACTOR

Concerning the profit factor, our modified *PG_RPadv* still outperforms the other methods. The adversarial learning model is more adept at maximizing profits than the representation network.

In conclusion, our proposed *PG_RPadv* is effective overall in various evaluation methods. The sole implementation of adversarial learning may lead to high profits (i.e., profit factor and profit), but this approach has limited ability in controlling risks (i.e., MDD and Sharpe ratio). By contrast, solely implementing market representation techniques may lead to effective risk control but lower profits. Nevertheless, both of these approaches still outperform the original PG method.

F. PERFORMANCE OF ADAPTIVE SAMPLING STRATEGY

The goal of our adaptive sampling strategy is to spend more time learning with the portfolios that the agent is not familiar with or in which it has relatively poor performance. In Section III-F, the company's sampling weight design is divided into two categories according to whether different or same sampling weights are applied for the companies on different trading days. In the first case, the company's sampling weight remains constant on different trading days; this group is represented by "Sample Strategy 1" in the following description and in Tables 6 and 7. For "Sample Strategy 2," the company's sampling weight differs on different trading days. To verify our sampling mechanism, we employ two RL-based methods, AC and PG algorithms, with our sampling mechanism. This experiment employs the same testing data as the previous experiment. The results are listed in Tables 6 and 7. We use six models in this experiment, namely PG, AC, PG Sample Strategy 1, AC Sample Strategy 1, PG Sample Strategy 2, and AC Sample Strategy 2.

TABLE 6. Sharpe and profit for sampling method in 2017.

	Sharpe Ratio						Net Profit					
	Original		Sample Strategy 1		Sample Strategy 2		Original		Sample Strategy 1		Sample Strategy 2	
Test set	PG	AC	PG	AC	PG	AC	PG	AC	PG	AC	PG	AC
0	0.31	0.12	0.3	0.41	0.19	**0.48**	831.66	283.97	672.48	759.09	490.14	**979.81**
1	**0.49**	0.32	0.35	0.41	0.37	0.35	**1098.7**	762.88	790.29	734.6	1064.61	714.83
2	-0.01	0.06	0.12	**0.19**	0.2	0.29	-36.37	159.92	277.57	389.26	555.78	**576.06**
3	0.41	0.31	0.43	**0.46**	0.36	0.44	**851.29**	568.59	801.77	751.78	714.19	835.46
4	0.47	0.41	0.37	**0.48**	0.3	0.41	**1032.65**	795.07	753.76	792.49	724.55	920.26
5	0.22	0.25	0.3	0.16	**0.3**	0.23	592.27	507.21	658.01	305.67	**638.81**	529.6
6	0.23	0.23	0.22	0.4	0.25	**0.34**	780.88	500.95	574.56	**885.71**	597.9	787.82
7	0.11	0.16	0.2	**0.23**	0.09	0.13	348.09	373.39	**447**	431.13	240.1	270.59
8	0.02	0.08	0.01	0.11	0.04	**0.15**	59.43	176.89	31.26	227.96	106.41	**357.85**
9	0.01	0.19	0.23	**0.33**	0.25	0.25	15.72	304.91	381.49	**561.22**	518.39	435.37
10	-0.13	0.08	0.01	0.03	0.11	**0.14**	-356.85	200.56	36.45	63.11	**283.53**	258.9
11	-0.03	**0.29**	0.25	0.19	0.25	0.27	-71.9	614.82	531.12	332.7	**629.3**	519.18
12	0.16	0.02	0.04	**0.07**	**0.07**	0.05	**427.26**	50.17	84.68	133.03	172.14	113.61
13	-0.04	**0.11**	0.05	-0.06	-0.02	-0.11	-124.8	**216.58**	105.48	-111.8	-55.99	-262.46
14	0.19	0.15	0.15	0.3	0.13	**0.27**	511.27	305.56	329.36	**556.29**	298.8	552.2
15	**0.37**	0.04	0.13	0.18	0.04	0.19	937.39	95.66	305.69	**369.28**	114.01	441.42
16	-0.16	0	**0.03**	-0.02	-0.06	0.02	-450.05	7.53	**74.82**	-55.04	-143.82	43.69
17	0.04	0.2	**0.16**	0.12	0.15	0.1	130.89	**406.38**	324.71	231.78	398.09	204.14
18	0.14	**0.23**	0.16	0.13	0.11	0.1	387.72	**497.37**	405.35	277.31	283.48	253.71
19	0.06	0.06	0.12	0.15	**0.16**	0.15	182.26	120.24	274.31	274.88	**352.2**	310.66
20	0.29	0.34	0.37	**0.49**	0.32	0.39	723.57	640.02	**808.57**	752.29	710.29	735.18
21	0.2	0.27	0.32	0.41	0.37	**0.46**	539.78	668.33	781.43	805.03	**1062.29**	978.13
22	-0.06	**0.16**	0.1	0.02	0.07	0.05	-244.24	**372.15**	278.47	40.51	186.48	115.54
23	0	0.11	0.16	**0.33**	0.16	0.24	13.67	210.5	338.48	**546.95**	383.21	457.94
24	0.07	0.16	0.14	**0.18**	0.12	0.03	215.74	**344.11**	334.46	363.28	297.75	68.67
25	0.02	0.21	0.3	**0.31**	0.22	0.32	51.1	355.45	573.25	457.31	483	**641.66**
26	-0.06	0.15	**0.07**	0.02	0.09	0.04	-244.45	**349.79**	213.84	47.97	237.17	97.4
27	0.07	0.17	0.24	0.27	0.22	**0.27**	223.59	341.91	602.45	**608.03**	557.74	581.09
28	-0.03	0.18	**0.25**	0.24	0.19	0.21	-87.76	337.65	481.11	357.37	**487.31**	417.73
29	0.14	0	0.16	0.29	0.2	**0.33**	337.01	3.88	303.78	493.52	448.53	**653.86**
Average	0.12	0.17	0.19	**0.23**	0.18	0.22	289.18	352.41	419.2	412.76	427.88	**453**

TABLE 7. Profit factor and maximum drawdown for sampling method in 2017.

	Profit Factor						Maximum Drawdown					
	Original		Sample Strategy 1		Sample Strategy 2		Original		Sample Strategy 1		Sample Strategy 2	
Test set	PG	AC	PG	AC	PG	AC	PG	AC	PG	AC	PG	AC
0	2.3	1.35	2.14	2.94	1.63	**3.39**	244.23	280.17	201.76	**157.33**	245.93	168.34
1	**4.3**	2.31	2.58	3.24	2.57	2.68	**103.93**	152.82	134.74	121.58	168.63	141.87
2	0.97	1.18	1.36	1.64	**1.66**	2.14	392.57	458.26	321.42	**168.1**	338.65	106.56
3	2.65	2.28	2.91	**3.25**	2.42	3.09	147.45	175.79	138.34	160.3	**134.43**	183.47
4	3.34	2.94	2.57	**3.37**	2.23	2.7	83.09	137.19	149.55	**66.05**	261.6	110.4
5	1.86	1.95	**2.18**	1.5	2.16	1.77	181.7	156.92	**152.23**	210.16	268.84	272.97
6	1.75	1.78	1.74	**2.75**	1.86	2.27	275.36	153.12	199.15	**131.87**	188.95	152.16
7	1.35	1.53	1.66	**1.77**	1.25	1.38	358.4	301.67	263.59	**197.25**	347.53	210.55
8	1.06	1.24	1.03	1.33	1.11	**1.48**	371.73	**186.79**	282.62	210.61	286.37	346.13
9	1.02	1.61	1.74	**2.57**	1.93	1.91	341.96	176.56	144.41	145.44	**142.24**	192.24
10	0.73	1.23	1.03	1.09	1.33	**1.42**	529.17	261.81	321.14	258.56	328.7	**191.47**
11	0.93	2.09	1.83	1.63	1.88	**1.92**	293.51	149.04	**121.68**	209.65	146.88	130.38
12	**1.57**	1.06	1.1	1.18	1.19	1.15	255.01	364.58	356.01	**235.57**	351.06	250.57
13	0.89	**1.31**	1.13	0.85	0.94	0.74	481.32	**223.05**	285.36	308.7	354.65	487.81
14	**1.59**	1.46	1.46	2.08	1.38	2.02	322.71	256.33	315.26	**184**	275.09	213.84
15	**2.7**	1.1	1.38	1.57	1.11	1.6	166.1	287.33	210.75	**126.55**	267.6	161.43
16	0.67	1.01	**1.09**	0.94	0.85	1.05	579.75	**257.6**	269.98	327.05	405.87	243.65
17	1.12	**1.63**	1.49	1.33	1.44	1.27	403.83	226.56	213.4	**172.21**	281.38	227.26
18	1.43	**1.79**	1.52	1.41	1.35	1.29	382.08	246.67	314.82	**311.22**	468.51	333.14
19	1.18	1.16	1.36	1.48	**1.52**	1.47	384.23	208.71	231.46	179.75	198.48	**160.38**
20	2.2	2.34	2.65	**3.69**	2.3	2.7	189.32	195.02	164.87	**99.91**	162.24	161.25
21	1.68	2.04	2.29	2.89	2.56	**3.27**	398.03	311.85	272.52	258.49	252.93	**139.36**
22	0.86	**1.52**	1.3	1.05	1.23	1.13	550.37	377.6	331.24	413.49	353.14	**312.52**
23	1.01	1.35	1.52	2.3	1.52	**1.85**	323.95	212.82	267.74	**157.38**	318.22	177.97
24	1.21	1.5	1.39	**1.53**	1.37	1.08	408.12	296	286.59	**222.89**	252.39	263.91
25	1.05	1.7	2.27	2.17	1.79	**2.31**	528.53	173.63	137.96	**99.23**	175.42	110.52
26	0.86	**1.44**	1.2	1.05	1.24	1.1	512.53	**233.36**	323.03	340.44	262.31	290.78
27	1.19	1.55	1.82	1.95	1.76	**1.96**	318.93	191.02	178.39	**143.36**	174.62	145.97
28	0.92	1.58	**1.93**	1.86	1.64	1.71	314.39	143.95	141.4	**72.7**	208.73	190.64
29	1.46	1.01	1.52	**2.11**	1.68	2.43	340.96	450.51	342.58	**197.9**	375.54	132.75
Average	1.53	1.6	1.71	**1.95**	1.63	1.88	339.44	241.56	235.8	**196.26**	266.56	207.01

1) SHARPE RATIO

The average Sharpe ratio of the 30 environments using our sampling mechanism is approximately 7 % higher than that of environments using random sampling (Table 6). PG with Sampling Strategy 1 has slightly higher performance than PG with Sampling Strategy 2. Moreover, AC with Sampling Strategy 1 has the highest performance, followed by AC with Sampling Strategy 2 and the naïve AC.

2) NET PROFIT

In terms of net profit, both AC and PG with Sampling Strategy 2 substantially outperform the other methods. This finding indicates that distinct sampling weights on different trading days can enhance profit earning ability.

3) MAXIMUM DRAWDOWN

In terms of MDD, for both the PG or AC RL algorithms, Sampling Strategy 1 outperforms the original method and Sampling strategy 2. In terms of MDD, although the loss suffered by investors during the period is reduced, the degree of the reduction is not as high as is the case with the market representation network. This indicates that the sampling strategy can increase overall performance but only slightly reduce risk.

4) PROFIT FACTOR

The PG with Sampling Strategy 1 exhibits substantial improvement—its profit factor is nearly 23 times that of the original PG. Furthermore, AC with Sampling Strategy 2 exhibits a slightly lower result than AC with Sampling Strategy 1.

Overall, both our modified sampling methods provide more favorable results compared with our baseline, random sampling. Specifically, Sampling Strategy 1 provides higher results in terms of the Sharpe ratio, profit factor, and MDD. Thus, employing the same company weights on different trading days is more appropriate than employing different values. Finally, AC still outperforms PG in most circumstances, which is consistent with our first experiment (see Tables 2 and 3).

V. CONCLUSION

We proposed a comprehensive RL framework that can improve generalizability under limited computational resources by applying adversarial learning, a market representation network, and an adaptive sampling strategy. First, we designed an adaptive sampling strategy to determine which data are worth learning by observing the learning condition. This strategy enabled the agent to learn the general trading strategy more effectively within a limited period. We also used adversarial learning during the RL process to enhance the model's robustness. By adding perturbations to our highly stochastic stock price streams, we enhanced the generalizability of our model. Moreover, to help our agent control risks, we pretrained an autoencoder network to obtain the latent space of the price and the VAR of the stock streams. This embedded network, namely the market representation network, helped our RL agent minimize risks and maximize returns throughout the portfolio management period. While the proposed RL learning system still displays less capability in risk control, in our future work, multi-agent framework should be adopted and different agents could be designed to deal with different portfolio management objectives. In this multi-objective machine learning framework, some agents focus on portfolio profit maximization, while others might aim to minimize the risk of the decision making. The agents can collaboratively find an optimal strategy for portfolio management which balance return and risk. In addition, with the use of auto-ML techniques, the hyperparameters of each experimental settings may be optimized automatically without manually tuning processes.

Our experimental results indicated that compared with the naïve learning strategy, the model with our sampling strategy had 6 %–7% higher performance in terms of the Sharpe ratio and 45 % higher performance in terms of the profit value. Furthermore, the model employing adversarial learning and the market representation network reduced the MDD by 40 % and increased the Sharpe ratio and profits by 22 times. The goal of this study was to enable the agent to learn the general rules of trading by exploring and learning from a wide variety of environments containing diverse portfolios. Many recent studies have employed similar approaches to address the so-called lifelong learning problem. Lifelong learning is achieved through continuous learning of new knowledge and with the aim of achieving excellent results in various tasks. To enable this process, we must confirm that the knowledge of the model is transferable, and that the model does not forget the experience it has acquired. In reality, if the model learns to perform task 1 and then learns task 2, the model is highly likely to forget the knowledge acquired in task 1. Thus, we must incorporate multiple tasks into the model and learn simultaneously to limit knowledge loss. However, the ability of models is limited. As tasks continue to increase, we cannot expect a model to address all the problems. Therefore, in future work, we hope to apply the notion of metalearning. The purpose of metalearning is to learn how to learn something. By continually learning tasks, the model can become more proficient at learning and can subsequently learn faster when a new task is presented.Also, we expect to adopt multi-agent systems to realize multi-objective learning. This would increase the agent's ability in incorporating important information, and achieve robustness and generalizability of proposed reinforcement learning-based trading system.

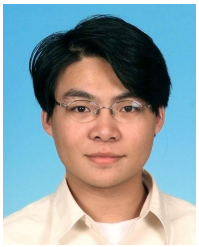
REFERENCES

[1] J. Moody and M. Saffell, "Learning to trade via direct reinforcement," *IEEE Trans. Neural Netw.*, vol. 12, no. 4, pp. 875–889, Jul. 2001.

[2] 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.

[3] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017, *arXiv:1706.10059*. [Online]. Available: http://arxiv.org/abs/1706.10059

[4] P. Yu, J. S. Lee, I. Kulyatin, Z. Shi, and S. Dasgupta, "Model-based deep reinforcement learning for dynamic portfolio optimization," 2019, *arXiv:1901.08740*. [Online]. Available: http://arxiv.org/abs/1901.08740
[5] F. Feng, H. Chen, X. He, J. Ding, M. Sun, and T.-S. Chua, "Enhancing stock movement prediction with adversarial training," 2018, *arXiv:1810.09936*. [Online]. Available: http://arxiv.org/abs/1810.09936
[6] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Comput. Sci.*, vol. 147, pp. 400–406, Jan. 2019.
[7] H. Ouyang, X. Zhang, and H. Yan, "Index tracking based on deep neural network," *Cognit. Syst. Res.*, vol. 57, pp. 107–114, Oct. 2019.
[8] S. Kim and S. Kim, "Index tracking through deep latent representation learning," *Quant. Finance*, vol. 20, no. 4, pp. 639–652, Apr. 2020.
[9] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica, "RLlib: Abstractions for distributed reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 3053–3062.
[10] H. M. Markowitz, "Portfolio selection/harry Markowitz," *J. Finance*, vol. 7, no. 1, pp. 77–91, 1952.
[11] R. O. Michaud, "The Markowitz optimization enigma: Is 'optimized'optimal," *Financial Analysts J.*, vol. 45, no. 1, pp. 31–42, 1989.
[12] H. Hong and J. C. Stein, "A unified theory of underreaction, momentum trading, and overreaction in asset markets," *J. Finance*, vol. 54, no. 6, pp. 2143–2184, Dec. 1999.
[13] K. G. Rouwenhorst, "International momentum strategies," *J. finance*, vol. 53, no. 1, pp. 267–284, 1998.
[14] S. Koyano and K. Ikeda, "Online portfolio selection based on the posts of winners and losers in stock microblogs," in *Proc. IEEE Symp. Ser. Comput. Intell. (SSCI)*, Nov. 2017, pp. 1–4.
[15] F. Z. Xing, E. Cambria, and R. E. Welsch, "Intelligent asset allocation via market sentiment views," *IEEE Comput. Intell. Mag.*, vol. 13, no. 4, pp. 25–34, Nov. 2018.
[16] L. Malandri, F. Z. Xing, C. Orsenigo, C. Vercellis, and E. Cambria, "Public mood–driven asset allocation: The importance of financial sentiment in portfolio management," *Cognit. Comput.*, vol. 10, no. 6, pp. 1167–1176, Dec. 2018.
[17] G.-Y. Ban, N. El Karoui, and A. E. Lim, "Machine learning and portfolio optimization," *Manage. Sci.*, vol. 64, no. 3, pp. 1136–1154, 2018.
[18] J. Branke, B. Scheckenbach, M. Stein, K. Deb, and H. Schmeck, "Portfolio optimization with an envelope-based multi-objective evolutionary algorithm," *Eur. J. Oper. Res.*, vol. 199, no. 3, pp. 684–693, Dec. 2009.
[19] N. Hachicha, B. Jarboui, and P. Siarry, "A fuzzy logic control using a differential evolution algorithm aimed at modelling the financial market dynamics," *Inf. Sci.*, vol. 181, no. 1, pp. 79–91, Jan. 2011.
[20] L. L. Macedo, P. Godinho, and M. J. Alves, "Mean-semivariance portfolio optimization with multiobjective evolutionary algorithms and technical analysis rules," *Expert Syst. Appl.*, vol. 79, pp. 33–43, Aug. 2017.
[21] Z. Jiang and J. Liang, "Cryptocurrency portfolio management with deep reinforcement learning," in *Proc. Intell. Syst. Conf. (IntelliSys)*, Sep. 2017, pp. 905–913.
[22] Q. Kang, H. Zhou, and Y. Kang, "An asynchronous advantage actor-critic reinforcement learning method for stock selection and portfolio management," in *Proc. 2nd Int. Conf. Big Data Res. (ICBDR)*, 2018, pp. 141–145.
[23] S. Almahdi and S. Y. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," *Expert Syst. Appl.*, vol. 87, pp. 267–279, Nov. 2017.
[24] Z. Liang, H. Chen, J. Zhu, K. Jiang, and Y. Li, "Adversarial deep reinforcement learning in portfolio management," 2018, *arXiv:1808.09940*. [Online]. Available: http://arxiv.org/abs/1808.09940
[25] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 1282–1289.
[26] J. Oh, S. Singh, H. Lee, and P. Kohli, "Zero-shot task generalization with multi-task deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2661–2670.
[27] A. Zhang, N. Ballas, and J. Pineau, "A dissection of overfitting and generalization in continuous reinforcement learning," 2018, *arXiv:1806.07937*. [Online]. Available: http://arxiv.org/abs/1806.07937
[28] J. Farebrother, M. C. Machado, and M. Bowling, "Generalization and regularization in DQN," 2018, *arXiv:1810.00123*. [Online]. Available: http://arxiv.org/abs/1810.00123
[29] C.-H. Kuo, C.-T. Chen, S.-J. Lin, and S.-H. Huang, "Improving generalization in reinforcement learning–based trading by using a generative adversarial market model," *IEEE Access*, vol. 9, pp. 50738–50754, 2021.
[30] Y.-Y. Chen, C.-T. Chen, C.-Y. Sang, Y.-C. Yang, and S.-H. Huang, "Adversarial attacks against reinforcement learning-based portfolio management strategy," *IEEE Access*, vol. 9, pp. 50667–50685, 2021.
[31] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.
[32] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.
[33] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015, *arXiv:1506.02438*. [Online]. Available: http://arxiv.org/abs/1506.02438
[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*. [Online]. Available: http://arxiv.org/abs/1707.06347
[35] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," in *Proc. 17th Int. Conf. Auto. Agents MultiAgent Syst.*, 2018, pp. 2040–2042.



SZU-HAO HUANG (Member, IEEE) received the B.E. and Ph.D. degrees in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2001 and 2009, respectively. He is currently an Assistant Professor with the Department of Information Management and Finance, and the Chief Director of the Financial Technology (FinTech) Innovation Research Center, National Yang Ming Chiao Tung University, Hsinchu. He is also a Principal Investigator of the MOST Financial Technology Innovation Industrial-Academic Alliance and several cooperation projects with leading companies in Taiwan. He has authored more than 50 articles published in the related international journals and conferences. His research interests include artificial intelligence, deep learning, recommender systems, computer vision, and financial technology.

YU-HSIANG MIAO received the B.S. degree in information management from Yuan Ze University, Taoyuan, Taiwan, in 2017, and the M.S. degree in information management from National Chiao Tung University, Hsinchu, Taiwan, in 2019. His research interests include artificial intelligence, deep learning, and financial technology (FinTech).

YI-TING HSIAO received the B.S. degree in interdisciplinary program of engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2020. She is currently a Researcher with the Financial Technology (FinTech) Innovation Research Center, National Yang Ming Chiao Tung University, Hsinchu. Her research interests include reinforcement learning, deep learning, and financial technology (FinTech).