Received March 11, 2021, accepted March 20, 2021, date of publication March 24, 2021, date of current version April 7, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3068768

Adversarial Attacks Against Reinforcement Learning-Based Portfolio Management Strategy

YU-YING CHEN[1], CHIAO-TING CHEN[2], CHUAN-YUN SANG[1], YAO-CHUN YANG[1], AND SZU-HAO HUANG[3], (Member, IEEE)

[1]Institute of Information Management, National Chiao Tung University, Hsinchu 30010, Taiwan
[2]Department of Computer Science, National Chiao Tung University, Hsinchu 30010, Taiwan
[3]Department of Information Management and Finance, National 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 Many researchers have incorporated deep neural networks (DNNs) with reinforcement learning (RL) in automatic trading systems. However, such methods result in complicated algorithmic trading models with several defects, especially when a DNN model is vulnerable to malicious adversarial samples. Researches have rarely focused on planning for long-term attacks against RL-based trading systems. To neutralize these attacks, researchers must consider generating imperceptible perturbations while simultaneously reducing the number of modified steps. In this research, an adversary is used to attack an RL-based trading agent. First, we propose an extension of the ensemble of the identical independent evaluators (EIIE) method, called enhanced EIIE, in which information on the best bids and asks is incorporated. Enhanced EIIE was demonstrated to produce an authoritative trading agent that yields better portfolio performance relative to that of an EIIE agent. Enhanced EIIE was then applied to the adversarial agent for the agent to learn when and how much to attack (in the form of introducing perturbations).In our experiments, our proposed adversarial attack mechanisms were $> 30\%$ more effective at reducing accumulated portfolio value relative to the conventional attack mechanisms of the fast gradient sign method (FSGM) and iterative FSGM, which are currently more commonly researched and adapted to compare and improve.

INDEX TERMS Reinforcement learning, adversarial attack.

I. INTRODUCTION

Portfolio management is a long-standing field of research. The use of rule-based algorithms (e.g., in automatic trading) has enabled effective performance in portfolio management. With the increasing prevalence of deep learning, many researchers have leveraged neural network–based models to obtain better performance. Reinforcement learning (RL) has recently been used to solve portfolio management tasks. These methods perform excellently, but they carry vulnerabilities under certain conditions. Researchers associate automated trading with high volatility. Automated trading systems can use preprogrammed rules when a stock price falls below a predefined threshold. Because these algorithms make decisions based on historical data, they may be more sensitive to movements in prices, which means that a given strategy may let investors quickly move in and out of the market due to sudden price movements.

Besides, deep neural network (DNN) models have been proven to be vulnerable to adversarial samples. Adversarial samples generated by attackers can cause serious malfunctions in DNN models. Such malfunctions occur from the addition of malicious perturbations on the input data; such additions have been widely studied in object recognition tasks. A well-trained DNN model will be misled into misclassifying an image with a high level of confidentiality on account of the presence of imperceptible adversarial samples. Many researchers have applied attack methods to image classification tasks. The loss of the victim model entails a loss on applications after images are misclassified. If the victim model constitutes a trading strategy, the adversary aims to make the trading strategy perform poorly (e.g., to yield

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

low returns). Thus, the corresponding loss incurred by the trading strategy is larger than that incurred from classification models.

Recent studies have increasingly applied attack mechanisms to advanced neural network–based models, such as RL models. Most studies on attacks have investigated the application of RL models through the direct use of traditional attack mechanisms, such as the fast gradient sign method (FGSM) [1]. However, a critic network must be present to evaluate how much reward is reduced by perturbations made at a current time step after a sequence of interactions; such evaluation allows the model to be updated and to learn well. If a critic network is absent, then the adversary must attempt to attack in the absence of an evaluation, which makes the adversary liable to adopting a suboptimal attack method. In addition, scholars have identified many other challenges in attack RL models due to these models' differences in architecture and training processes compared with traditional adversarial attacks on supervised classification algorithms.

First, the traditional supervised model aims to output a specific answer according to a given input. The goal of the attack is thus to dupe the supervised model into predicting a wrong answer with high confidence. By contrast, the RL agent interacts with the environment: the agent's actions change the state of an environment, and the agent subsequently receives a sequence of observations. The tasks of RL algorithms are much more complicated than those of traditional supervised models, and the training process is based on interactions. Thus, the goal of the attacker is to mislead the agent not only to act inappropriately but also to act with less efficiency than the agent otherwise would in the absence of an attack. Second, a key requirement for the attacker is that perturbations from the attack be as slight as possible for the attack to be imperceptible to a human user in supervised tasks. Similarly, in relation to an RL agent, the attacker should not only produce small adversarial samples but must also reduce the number of modified steps. This is because the victim agent more easily detects an attack with a larger number of modified steps.

In this paper, we aimed to prove the vulnerability of RL models in portfolio management. Because previous works on attack RL models have failed to provide suitable solutions to addressing RL model vulnerabilities, we propose using an RL-based adversarial attack method against a victim model; this method yields a well-trained RL trading system for portfolio management. We believe that both classifiers and well-performing RL agents can be vulnerable to artificial intelligence attackers. The main architecture of our method is identical to that of multi-agent RL, where an attack agent is adopted to attack the victim agent. Specifically, the adversary places fake orders for $< 0.0001\%$ of the price of the original clean orders, causing the victim model to perform poorly with respect to the standard three evaluation metrics. The figure 1 shows that an adversary decides on the timing and intensity of the attack based on the data that it examines

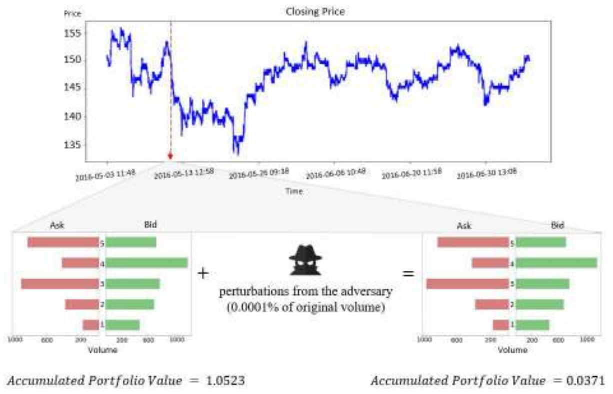

FIGURE 1. Example of our proposed RL-based adversarial trading agent.

during a given period. Consider the case of a single evaluation metric, where the accumulated portfolio value (APV) is defined as the difference between the final and initial portfolio values. After negligible perturbation is added to the original volume data, the APV of the victim agent decreases (from 1.0523 to 0.0371 in the example illustrated in Fig. 1). Our victim model was inspired by the ensemble of identical independent evaluators (EIIE) networks proposed in 2017, which is well known for its ability to yield high returns compared with traditional trading methods [2]. We extended this framework by adding volume information to improve performance, and we termed this modified method enhanced EIIE. Furthermore, we ensured that the environment in enhanced EIIE is as similar as possible to a real financial market. Therefore, we actively selected assets with the largest trading volumes in the market as the expected components of the investment portfolio. Every state in enhanced EIIE contains two elements: the price and volume of the five best ticks. We aimed to execute the least possible number of attacks on states in enhanced EIIE. We did so by adding ''fake'' orders to limit orders within the five best ticks, which were to be canceled at the next time step. Consequently, enhanced EIIE malfunctions, which causes the loss of capital.

Our major contributions are as follows:

- We propose a cutting-edge method for simultaneously analyzing adversarial attacks on RL models and the trading strategy adopted.
- We proved that even a well-performing portfolio management agent has some security vulnerabilities.
- To our knowledge, we are the first to attack an RL trading system with RL-based adversarial attack algorithms.
- We proved that attacks on an RL trading strategy result in a greater loss than do attacks on strategies based on classification models. This is because, in the case of attacks, investors in an RL trading system lose their capital, whereas investors in a classification model system only incur a loss from the misclassification of images.

II. RELATED WORK

In this section, we review the abundant literature related to our work. We first discuss traditional methods and deep learning algorithms that are used to solve portfolio management problems in finance. Subsequently, we review studies on traditional adversarial attack techniques that are used to solve classification tasks.

A. ADVERSARIAL ATTACKS IN CLASSIFICATION TASKS AND RL MODELS

Machine learning and deep learning techniques have recently been adopted in various fields, such as natural language processing and computer vision. Although deep learning models perform well in many fields, weaknesses remain in DNNs when they are applied to image classification tasks; these weaknesses were first found in 2013 [3]. Although these models are highly accurate, they are susceptible to very small perturbations to images that are almost imperceptible to the human user. This phenomenon has attracted the attention of other researchers working on adversarial attacks and deep learning security. After these discoveries were made, researchers discussed the fragility of machine learning and deep learning models.

Pioneering attack methods all feature a one-step attack, meaning that the victim model must be successfully attacked in only one step. For example, Szegedy *et al.* introduced the box-constrained L-BFGS. In this method, the aim is to find an adversarial sample that is similar to the input data, where this adversarial sample leads to the model making poor decisions [3]. The FGSM is a nontargeted gradient-based attack algorithm that is designed to attack classifiers by computing the sign of the gradient [1], [4], [5]. Until now, many studies have used the FGSM architecture as an experimental control group or have attempted to invent algorithms based on this architecture [6], [7]. A study revealed that one-step attacks are unlikely conducted to yield an optimal decision; thus, one-step-attack methods have been refined into methods where many steps are iterated to obtain the optimal perturbation [8]. These basic attack algorithms have been extended in newer attack algorithms, such as Deepfool [9], the Carlini–Wagner attack [10], and JSMA [11]; in these extended methods, the adversary can allocate perturbations in multiple steps.

In addition, most of the aforementioned traditional adversarial attack mechanisms have focused on classification tasks. However, the applicability of adversarial attacks extends far beyond classification tasks alone. The recurrent neural network, autoencoder (AE), variational autoencoder (VAE), and generative adversarial network were designed to undertake different tasks. Many studies have also proven that these models are vulnerable to adversaries despite their outstanding performance [12]–[14]. Besides, all current models based on neural network architectures are vulnerable to adversaries.

RL, in which the best action is found through interactions with the environment, has been applied to many tasks. The literature on RL security has increased with the widening of RL applications. In RL, the adversarial attack is any perturbation that potentially results in the agent receiving less reward or being more likely to undertake the worst possible action. The adversary can target any component of the Markov decision process (MDP). First, the adversary may choose to perturb rewards by either attacking rewards directly [15]–[17] or attacking other indirect parts of the RL training process [18], [19]. Second, the adversary can target the agent of RL models. Indirect methods include having the adversary execute the following tasks: adding perturbations to states or the environment [20], [21], controlling another adversarial agent in the same environment as the one the original agent was under [22], attacking observations by modifying the sensory data [23], introducing high-confidence perturbations to the environment, and modifying the environment by perturbing states or by manipulating environmental dynamics [21], [24]. Although the RL agent can be attacked in many ways, most studies have applied traditional but sub-optimal attack mechanisms against the agent. In our literature review, we noted a paucity of research on adversarial attacks against RL. Such paucity has been due to the difference between RL and other neural networks with respect to training processes. Therefore, the literature on attacks against RL models does not provide a solution to our problem.

In this section, we provide the detail description of our problem definition in digital domain attack, and then extend the formulation in order to generate a robust adversarial examples for physical domain attack. Let x denote an clean input image and y_{true} denote the corresponding ground-truth label, given a classifier C: $C(x) = y$ that outputs the prediction label of the input. To generate an adversarial example $x_{adv} = x + \rho$, which is human-imperceptible from x but deceives the classifier, i.e. $C(x_{adv}) \neq y_{true}$, the objective function can be written as the following optimization problem:

$$\arg\min_{x_{adv}} \mathcal{L}(C(x_{adv}), y_{target}), s.t. \ \|x_{adv} - x\|_p \leq \epsilon \tag{1}$$

where $J(.)$ is the cross-entropy loss of the classifier, and y_{target} is the target label which differs from the original label. In general, the distance between the adversarial example and the corresponding clean image needs to satisfy the L_p-norm bound, where p can be $0, 1, 2$ or ∞. The above optimization can be relaxed by using the corresponding Lagrangian-relaxed form:

$$\arg\min_{\rho} \mathcal{L}(C(x_{adv}), y_{target}) + \lambda|\rho\|_p \tag{2}$$

where λ is a hyper-parameter controlling the strength of the distance penalty term $|\rho\|_p$.

In the case of anti-spoofing, the C mentioned above becomes a simple binary classifier and the goal of the attacker is to fool the anti-spoofing model with the adversarial spoofing image, i.e. $y_{target} = true$. For the attacker, it's natural to craft the target images by adding perturbations on the spoofing images, since the real image doesn't have sufficient clues with the information of the decision of the victim model. In the digital domain, let $\hat{x}$ denote a spoofing image and the adversarial spoofing image $x_{adv} = \hat{x} + \rho$ is fed into the victim model directly, where adversarial noise ρ can be found by solving the optimization problem in Eq. (2). In the physical domain attack, the adversarial spoofing images need to go through another rebroadcast-recapture procedure before the model's final decision. For the purpose of synthesizing robust adversarial example in the physical domain, we followed the Expectation Over Transformation (EOT) algorithm [12] to generate the adversarial examples that remain adversarial over a chosen transformation distribution T. Therefore, the new optimization problem can be rewritten as:

$$\arg\min_{\rho} \; \mathbb{E}_{\mathrm{r}\sim\mathbb{T}}[\mathcal{L}(C(r(x_{adv})), y_{target})] + \lambda|\rho\|_p \tag{3}$$

where $r(.)$ denotes the rebroadcast-recapture procedure and T denotes the set of transformations such as random rotation, translation, or addition of noise, which the adversarial perturbation ρ needs to be robust to.

In the above equation, the face detection and face recognition module are not considered and the crafted adversarial spoofing image may not be accepted in the whole face authentication procedure. Thus, we added extra constraints in the above equation to generate the adversarial perturbation which considered both the face recognition and anti-spoofing module, the optimization problem can be rewritten as:

$$\begin{aligned}&\arg\min_{\rho} \; \mathbb{E}_{\mathrm{r}\sim\mathbb{T}}[\mathcal{L}(f_s(r(x_{adv})), y_{target})] + \lambda|\rho\|_p,\\ &s.t. \;\; f_d(r(x_{adv})) = 1, f_r(r(x_{adv})) = l_{\hat{x}}\end{aligned} \tag{4}$$

where f_s denotes the output of the spoofing module, f_d denote the output of the face detection module, f_r denotes the output of the face recognition module. For the face detector, $f_d(.) = 1$ stands for the real face (and 0 for non-face). For the face recognition module, $f_r(.) = l_{\hat{x}}$ stands for the correct identity for the image $\hat{x}$. In the anti-spoofing module, $f_s(.) = 1$ denotes the input image is a real person (and 0 for spoofing image). The attack is successful when $f_s(r(x_{adv})) = 1$ subject to the two constraints.

White-box Adversarial Morphing Attack In the morphing attack, the target of the attacker is to generate displacement fields for the images and remap the vectorized images and fields back to form the morphing attack images, which can be expressed as:

$$x_d = remap(x, (\Delta x, \Delta y)) \tag{5}$$

where $(\Delta x, \Delta y)$ denotes the corresponding displacement field of the clean image x, and the $remap(\cdot)$ function generate the deformed image x_d according to the displacement field and input image.

Under the white-box adversarial morphing attack settings, we assume that the attacker has complete knowledge of the victim model and can get access to the model parameters. Therefore, the attack can generate a unique displacement field for each image by back-propagating the classifier errors through the model parameters.

To generate the displacement field for each individual attack image, we propose a model architecture which is shown in figure(). The architecture of our model is similar to U-Net [38], it can be decomposed as an extraction path on the left and reconstruction path on right. In detail, each convolution layer has a kernel size of 3 × 3 with stride 2 for downsampling, following by a rectified linear unit (ReLU) or hyperbolic tangent (Tanh) activation function. In the extraction path, each component or block extracts the essential and complicated features of the input image. In the reconstruction path, the components or blocks aim to construct the displacement field according to the extracted information of the input image. However, much information has lost due to the dimensionality reduction of each layer in the extract path. Hence, the feature map of each level in the extract path concatenates with its corresponding level in the reconstruct path, and then as the input of the next layer. At the end of our architecture, since we found that the displacement field generated from our model will decompose the structure of the image, therefore, the Gaussian Smoothing layer will be applied to the displacement field before adding to the image.

Similar to adversarial noise attack, in order to solve the optimization problem in the anti-spoofing scenario by adapting the proposed morphing attack method, the deformed image should be human-imperceptible and deceives the spoofing detection model. Therefore, a constraint is applied to limit the movement of pixels for the optimization problem in Eq. (1), which can be reformulated as:

$$\arg\min_{x_d} \; \mathcal{L}(C(x_d), y_{target}) + \|(\Delta\mathbf{x}, \Delta\mathbf{y})\|_2^2 \tag{6}$$

In the end, the extra constraints of the face detection and face recognition model are needed to be consider, the optimization problem can be written as:

$$\begin{aligned}&\arg\min_{x_d} \;\; \mathcal{L}(C(x_d), y_{target}) + \|(\Delta\mathbf{x}, \Delta\mathbf{y})\|_2^2,\\ &s.t. \;\; f_d(r(x_d)) = 1, f_r(r(x_d)) = l_{\hat{x}}\end{aligned} \tag{7}$$

By limiting the classification loss and the amount of movement made by the morphing process, the model can learn to generate the appropriate displacement field for the morphing attack image, and the constraints of face detection and face recognition can also be solved easily by limiting the amount of displacement field.

B. PORTFOLIO MANAGEMENT TASKS WITH TRADITIONAL AND DEEP LEARNING METHODS

In this section, we review the literature on portfolio management. Portfolio management has been studied for decades. Fundamental to finance, portfolio management involves optimizing the allocation of a holder's wealth across a large set of assets. Scholars have formulated various techniques to improve portfolio management. The portfolio management problem can be formulated as a sequential decision problem, and existing methods for solving this problem draw either on mean-variance theory [25] or capital growth theory [26]. The mean-variance theory aims to achieve a balance between risk and expected return, and capital growth theory aims to maximize the expected return of the portfolio.

The aim of algorithms is typically to maximize cumulative return; such algorithms can be classified into several categories: traditional strategies, momentum strategies, contrarian strategies, pattern-finding algorithms, meta learning methods, and machine learning and deep learning mechanisms. Specifically, traditional strategies involve no knowledge: this strategy merely involves holding on to the best assets until the end [27]. Momentum strategies tend to follow the expected return of the optimal strategy, and these strategies often correspond to capital growth theory [28], [29]. Pattern-finding algorithms extract information from financial historical data and predict the market distribution, with the ultimate aim of optimizing the portfolio value based on the predicted distribution [30]. These aforementioned algorithms only employ one strategy. However, meta learning methods employ many strategies [31]. Because of the impressive advances in neural networks, machine learning, and deep learning, these methods have broadly replaced traditional feature extraction optimization algorithms in solving financial problems [32], [33]. However, basic algorithms leveraging machine learning and deep learning learn the task using supervised data, which are likely unable to capture market uncertainty. Researchers have thus used RL to solve this problem [34]–[36]. In RL, the trading agent acts after observing historical financial data. Subsequently, the agent receives a reward from the environment that reinforces good decisions and penalizes bad ones, thus helping the agent learn. To improve the performance of combinatorial learning in reinforcement learning, in addition to achieving investment goals, effective information must be collected. Many studies have attempted to add trading strategies or conditions to models, such as enabling agents to sell short [37], capture more market news [38], or decline trading risks [39].

[37] proposed a method that allows agent short selling in addition to long buying and selling. Apart from basic market data, [38] used financial data and market news to solve the problems of data heterogeneity and environmental uncertainty. Our proposed enhanced EIIE added volume of information, which affects the trading decisions of agents. When selecting portfolios in real life, investors must also consider risk management. The impact of risk on actual transactions must be considered. Unlike our enhanced EIIE, which focuses on investment gains and losses, [39] added a risk control item to the reward function to ensure that risks when trading are considered in the model. Our proposed attack method can affect the portfolio management model, cause losses, or even cause the victim model to lose the ability to manage risk.

To increase performance in portfolio management problems, some studies have proposed algorithms for comparison [40]. [40] applied three advanced reinforcement learning algorithms, namely Deep deterministic policy gradient (DDPG), proximal policy optimization (PPO), and policy gradient (PG), to portfolio management. They proposed the adversarial training method, which not only improves training efficiency but also proves that the agent based on the PG algorithm can outperform UCRP. In the present research, we proposed a related algorithm based on PG to attack the portfolio management model. In a later section, we discuss portfolio management strategies that are employed as the victim model and that also takes the form of an online stochastic batch learning scheme [2].

The findings of these studies collectively suggest that the portfolio management problem is not too difficult to solve; this means that algorithmic portfolio allocation can yield profits with relative ease. However, we believe that the training process has intrinsic deficiencies. This belief motivated us in terms of proving that trading systems are (easily) rendered useless when an adversarial attack adds imperceptible perturbations to the trading information.

III. PRELIMINARIES

In this paper, we aimed to attack a financial market by introducing a narrow range of perturbations to the volume of a given set of investment objectives. These perturbations cause a collapse in most automatic trading systems. The ultimate objective is for our attack costs to be much lower than those of other investors; under the best circumstances, our method can even seize the opportunity to conduct inverse operations. The entire system comprises two parts: the victim agent and our proposed attack agent (Figure 2). The goal of the enhanced EIIE victim model is to maximize portfolio value with respect to 11 fixed underlying assets; the model does so by assigning weights to these assets, where these weights determine how much is invested in a given asset. In opposition to the victim model, the adversary tries to add perturbations on the states of the victim agent by learning from the evaluations made by the victim model.

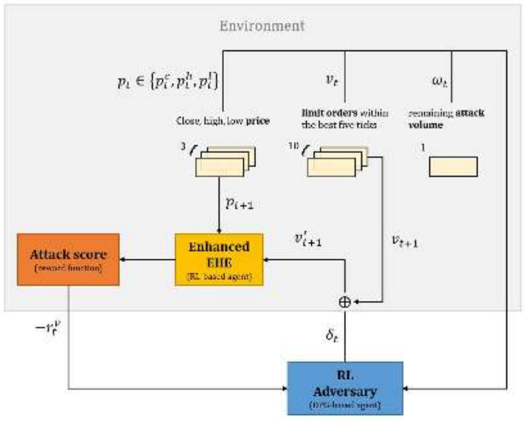
Environment
$p_t \in \{p_t^c, p_t^h, p_t^l\}$
v_t
ω_t
Close, high, low price
limit orders within the best five ticks
remaining attack volume
3
10
1
p_{t+1}
v'_{t+1}
v_{t+1}
Attack score (reward function)
Enhanced EIIE (RL based agent)
$-r_t^V$
δ_t
RL Adversary (DPG-based agent)

FIGURE 2. System overview.

A. PROBLEM DEFINITION

The attack process can be modeled as an MDP with finite time steps and a deterministic policy, which is adopted by the attacker. The overall framework is similar to that of multi-agent RL, but the two policies are not in a zero-sum game. We denote the adversary and victim as A and V respectively. The MDP $M = (S, (A^A, A^V), T, (R^A, R^V))$ comprises the following: a state set S, joint state transition function T, victim model action set A^V, victim model reward R^V, adversary action set A^A, and reward for the adversary R^A. The state set $S \in \{S^A, A^V, R^V\}$ comprises states from the financial market S^A, the actions of the victim model, and the reward of the victim model. $T : S^A \times A^A \to \Delta(S^A)$, where $\Delta(S^A)$ is a probability distribution on S^A. The reward function $R_t^A : S_t^A \times A^A \times S_{t+1}^A \to \mathbb{R}$ for time step t depends on the current state, the action of the adversary, and the subsequent state. At each time step t, the adversary selects an action $a_t^A \in A$ according to both the observation of state $s_t^A \in S$ and a policy. The policy $\pi : S \times A \to [0, 1]$ represents the probability distribution of state-action pairs where $\pi(a|x)$ is defined as the probability that the agent under a given policy chooses action a when observing state x.

We did not include states from the victim model S^V because these states comprise a subset of S^A. States from the financial market $S^A \in \{p^c, p^h, p^l, v, \omega\}$ are represented in terms of the closing price p^c, highest price p^h, lowest price p^l, volume v of the five best bids and asks, and the remaining attack volume ω of the 11 assets within 1 minute. The first four elements of S^A are states of the victim model, and each action of the victim model $A_t^V = w_t \in \{0, 1\}$ constitutes the portfolio weight of a given asset. $w_t = \pi^V(s_t, w_{t-1})$ denotes the action made by policy π^V given a set of observations and given the previous action at time step t. The victim model produces a portfolio vector w_t at the end of period t. The job of the victim model is to maximize the final portfolio value, which is equivalent to maximizing the average accumulated return R. Thus, we can also formulate the portfolio management problem as an MDP, where wealth is continually relocated across numerous assets.

In summary, the victim model constitutes the environment of the attack agent. Given current observations of a market and a trading strategy that reflects the state s_t^A, our goal is to learn a policy $\pi^A(s^A)$ by minimizing the reward from the victim agent; this policy governs which action is taken a_t^A (attack or do not attack).

- **Environment**: The attack environment is constructed by the victim agent and is detailed in the following section.
- **State** $s_t^A \in S^A$: The state comprises the actions and rewards of the victim agent and the market information. $\tau : S \times A \to S$ is the state transition function that exists to generate the next state from the current state-action pair.
- **Action** $a_t = \delta_t$: Let $a_t^A \in A$ denote the action at time step t. After observing information from the market and the actions of the victim model, the attack agent decides whether and how aggressively to attack.
- **Reward** r^A: The reward function is defined as a negative value of the reward from the victim agent. Specifically, where $\gamma \in [0, 1)$ is the discount factor.

$$r_t^A = -\sum_{i=0}^{T} \gamma^{(i-t)} r(s_i', a_i) \tag{8}$$

We made some basic assumptions for the enhanced EIIE victim model and for the adversaries. These assumptions are required to make the attack technique realistic. The assumptions for the enhanced EIIE victim model are as follows:

- **Assumption 1**: The capital that the enhanced EIIE victim decides to put into the market exerts no influence on the market. Thus, because the agent's actions have no market impact, other investors do not alter their decisions upon observing the actions of the enhanced EIIE victim. We assumed this because we used historical data, which cannot be changed, to train the enhanced EIIE victim model.
- **Assumption 2**: For all assets, liquidity is sufficiently high for every order to be fulfilled immediately at the last price for there to be no unsettled position. In addition, buy or sell orders remain unchanged; these orders form the portfolio weight and are placed in the market by the enhanced EIIE victim.

The assumptions for adversaries are as follows.

- **Assumption 1**: When the adversary is trained, we assume that the enhanced EIIE victim follows a fixed stochastic policy π_V and that this policy produces decisions that are consistent with the victim model stored with the best-performing weights. In some systems where safety is paramount, the standard research process is to validate and subsequently freeze the attack model to ensure that the victim model does not change its decision process due to retraining. Thus, having attacks be directed at a fixed model is a realistic design for both research and real-world applications.
- **Assumption 2**: The perturbations, which are fake orders, are not matched at the next time step. In other words, these orders are canceled before they are matched. Adversaries face a trade-off between the cost and success rate of attacks. If these orders are matched, then attack costs are large.

IV. ENVIRONMENT AND VICTIM MODEL

In this section, we describe our enhanced EIIE victim model, which draws on that of Jiang *et al.* [2]. The elements of the model are the learning target, design of states, actions, reward function, and policy network.

A. INPUT DATA AND LEARNING TARGET

We actively selected the expected components of the portfolio, and the input data fed into the victim model pertain to historical prices and the volume of the five best bids and asks. The input data is processed into an input tensor X of the shape (f, m, n) where $f = 13$ is the total feature number, $m = 15$ is the window size, and $n = 11$ is the number of non-cash assets. As mentioned, price data are used to calculate y, and doing so can lead to the change in total portfolio value for a given period. We used the closing price of all underlying assets. This price is denoted as p_t^i, $i \in A$, $t \in T$, which represents the price of the ith asset at the period tth that together comprise y.

Cash is a special asset; the agent decides how much cash to invest in assets. Because y is important for the agent to act, we defined y as the relative change in prices, as follows.

$$y_t = \frac{p_t^c}{p_{t-1}^c} = [1, \frac{p_{1,t}^c}{p_{1,t-1}^c}, \cdots, \frac{p_{11,t}^c}{p_{11,t-1}^c}]^T \tag{9}$$

Our aim was for the victim agent to learn how to maximize the final portfolio value by observing how the prices p^c, p^h, and p^l map onto y. Besides, the portfolio changes are calculated as follows, where P_t is the current portfolio value and the portfolio value at the initial time step in the period is set to be one $P_0 = 1$. Specifically,

$$P = P_0 \prod_{t=1}^{T} y_t \cdot w_{t-1} \tag{10}$$

where w_{t-1} is the portfolio weight after rebalancing.

Rebalancing is a necessary step for the financial trading process because price changes with time. Thus, the allocated original weight w'_{t-1} at time step $t-1$ necessarily changes at the subsequent time step t due to changes in price. For the weight to remain unchanged, some assets must be bought or sold. Specifically,

$$P'_t = P_t \cdot c \sum_{i=1}^{N} |w'_{t-1} - w_{t-1}| \tag{11}$$

where c is the transaction cost, which we set to 0.25%.

B. RL ENVIRONMENT AND AGENT

Scholars have believed that the information required to aid the agent in decision-making is concealed in asset prices. In particular, the scholar who formulated the EIIE believed that historical prices roughly represent the environment. However, raw price data are still difficult for the agent to process. Thus, the scholar who formulated the EIIE adopted subsampling and a history cut-off mechanism that included feature extraction, asset selection, and partial data from a recent number of periods. However, we believe that the volume of best bids and asks also constitutes rich data. We formulated a new feature extraction method based on this belief. In this method, we added volume data, allowing the expectations of other investors in the market to be observed. To make the victim model under the attack mechanism credible, we decided to add this volume of data into the training feature to improve the performance of the victim agent.

The agent's goal is to reallocate the wealth across assets through buying or selling these assets; such reallocation depends heavily on the difference between the portfolio weights w_{t-1} and w_t. Because w_{t-1} is the unchangeable historical decision, the agent's action pertains only to the allocation of w_t.

Because of the RL architecture, after a current action $a_t^V = w_t$ is executed, the agent receives a reward r_t^V for the present time step; this action affects the subsequent state s_{t+1}^V, and this reward affects the subsequent action $a_{t+1}^V = w_{t+1}$. This influence is due to w_{t-1} being considered as part of the environment, where w_{t-1} is added to the states for the implied impact factor to be included; this addition helps the agent make better decisions. Thus, the state at time step t includes a pair comprising X_t and w_{t-1}, which represent the external state and internal state, respectively. X_t comprises the price data set (p_t^c, p_t^h, p_t^l) and the volume v_t of the five best bids and asks. Specifically,

$$s_t^V = (X_t, w_{t-1}), \; where \; X_t = ((p_t^c, p_t^h, p_t^l), v_t) \tag{12}$$

C. REWARD FUNCTION

The goal of the victim agent is to maximize the final portfolio value PV at the end of the time step; this maximization is equivalent to the maximization of the logarithmic accumulated return R^V. Specifically,

$$\begin{aligned} R^V(s_1, a_1, \cdots, s_T, a_T) &= \frac{1}{T} ln \frac{PV_T}{P_0} \\ &= \frac{1}{T} \sum_{t=1}^{T} ln(y_t \cdots w_{t-1}) \\ &= \frac{1}{T} \sum_{t=1}^{T} r_t \end{aligned} \tag{13}$$

D. POLICY NETWORK

1D-CNN is the main type of the neural network for forming the policy function, where, as defined in Equation (12), a portfolio vector w is output given some input data. As mentioned, the input data include those on the volume of the five best bids and asks. In a real financial market, the trading strategies of investors depend not only on prices in the previous few minutes but also on other indicators, such as the volume of the five best bids and asks. Thus, we used the concept of the original paper, but our trading agent acts closer to how a real trader does. In this architecture, the last hidden layer is the scores for 11 asset and cash positions. Subsequently, the cash position and the asset scores are concatenated to output portfolio weights through the softmax activation function layer. Having the same design as the policy network in [2], the network for n assets functions independently while parameters are shared among these streams. These streams can thus be construed as being independent, and they behave like a host of many identical smaller networks separately observing and allocating the assets. The only interactions between these small networks occur through the softmax function because it is necessary to ensure that all allocation weights are not smaller than zero and that the sum of these weights does not exceed one. The formulator of this method called this mechanism identical independent evaluators and termed its topology feature EIIE.

Our enhanced EIIE is based on the aforementioned architecture and has been experimentally demonstrated to perform better than the EIIE does. The architecture shared by EIIE and enhanced EIIE is illustrated in Figure 3. This network is implemented mainly by convolution neural networks.

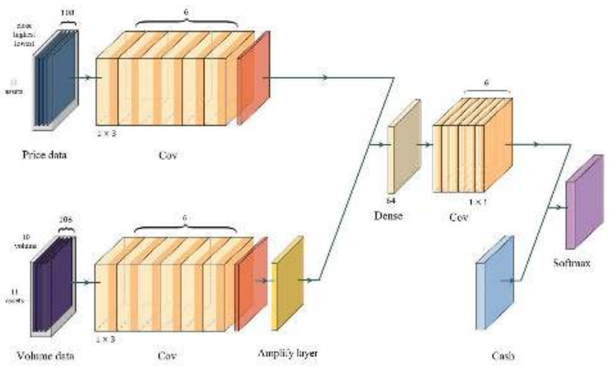

FIGURE 3. **The architecture of Enhanced EIIE.**

The input of the network can be divided into two parts. The first part is the price data represented by the tensor $3 \times n \times m$. This tensor comprises the closing, highest, and lowest prices of $n = 11$ assets over the past $m = 15$ periods. The second part is volume data, represented by the tensor $10 \times n \times m$. This tensor holds the volume of the five best bids and asks. Price data and volume data are fed into the 1D-CNN, in turn, to have the agent learn their different representations. Also, the weight of the stream from the volume data goes through an amplifying layer before concatenation. In the amplifying layer, we aim to strengthen the influence of volume data. Many traditional trading strategies make decisions that are based only on historical prices, and the agent must thus precisely predict the future price to yield a profit. By contrast, in our method, the trading agent makes decisions using the aforementioned volume data; in doing so, the agent can observe the actions of potential investors and can make rapid decisions as a result. This is because the volume of the five best bids and asks reflects the future expectations of investors. Subsequently, volume data and price data are concatenated, and the concatenated stream is fed into the dense layer and 1D-convolution layer. The cash is then added before a softmax layer is applied to output weights. In the entire feature map, the first dimension of the neural network layer is set to be 1 to isolate each row up until the softmax layer.

V. ADVERSARY

In this section, we detail the workings of a greedy adversary and an RL-based adversary. The learning target, reward function, actions, and policy network are identical for both adversaries; only their state spaces slightly differ.

A. PERTURBATIONS

In our work, we define the noisy state observation $s_t^{V'}$ of the perturbed enhanced EIIE victim model under attack as the sum of the clean state s_t^V and the perturbation pattern δ_t.

$$s_t^{V'} = s_t^V + \delta_t \tag{14}$$

The enhanced EIIE state that can be perturbed comprises price data and the volume of the five best bids and asks. However, price data cannot be reasonably perturbed because real-time changes in prices stem from different matched transaction results. Lower movements in prices indicate more capital put into the market. Thus, an investor that perturbs price data incurs a much higher cost relative to other investors in the market. In the subsequent subsections, we detail why we chose to perturb volume data.

TABLE 1. **Basic adversary attributes.**

Classification Rule	Attributes of Proposed Adversaries
Specificity	non-targeted attack
Objective of Adversaries	final accumulated reward reduction
Accessibility of Adversaries	black-box attack
Attack Frequency	multiple steps

B. SYSTEM DEFINITION

As mentioned in the literature review, attacks can be classified into several categories depending on the goal. Table 1 details the attributes of our adversary architecture. Because the adversary can decide whether a certain class should be output in a classification problem, traditional adversarial attacks can be classified into two main types by specificity: targeted attacks and non-targeted attacks. In our case, our problem is a continuous portfolio management problem where no finite actions can be chosen as the target. Thus, the objective of the adversary is to confuse the agent's decision-making process to minimize the agent's reward. Because our adversary has no right to obtain the parameters of the victim model, this attack is a black-box attack. The adversary designs an attack method using what little available knowledge it has and evades the EIIE based on the transferability property of adversarial samples. Such an attacker behavior implies that an adversarial sample that attacks a model successfully can also attack other similar models even when different data sets are used. By contrast, most studies on attacks have focused on classification tasks where the goal is either to elicit misclassification or reduce the confidence level of the victim model. In our study, because our victim model is an RL agent, the goal of the adversary is simply to reduce the final expected accumulated reward. Because the adversary is an RL-based algorithm and our data are continuous trading data, the adversary must undergo many steps, depending on the attack frequency, that culminates in the decision of whether to attack. This approach by the adversary has advantages and disadvantages: the adversary is helped by its ability to diversify attacks in multiple steps, which may increase attack performance, but is hindered by the substantial difficulty in learning the optimal attack strategy.

As for RL classification, the adversary has many choices as to its method of attack, which can be any element in the

MDP (s, a, r, p). Hence, the adversary can target states, rewards, the environment, or the policy. In our study, we target rewards by perturbing the state space.

C. ENVIRONMENT AND AGENT

In our study, the environment is an enhanced EIIE victim model. The adversary is allowed black-box access to states and actions sampled from the victim model π_V, and the victim model is not allowed to obtain information pertaining to, for example, model weights or activation functions. The goal is to find an attack policy that can maximize the expected reward over a sequence of state–action pairs $S = \{s_t^A, a_t^A\}$. In our setting, T was set to be 15. For the RL architecture, an adversary that can perturb more steps is more likely to greatly reduce the reward of the victim model. If the adversary perturbs the state at time step t, where $s_t^V = s_t^{V'}$, the action a_t^V of the victim agent becomes $a_t^{V'}$ due to different states.This action may then reduce the instantaneous reward r_t^V; such actions from a sequence of interactions decrease the final accumulated reward. Specifically,

$$\pi_A^* = argmax_{\pi_A}\{\mathbb{E}_{S\sim p(S|\pi_A)} \sum_{t=1}^{T} r(s_t^A, a_t^A)\} \tag{15}$$

At a higher level, the accumulated reward $\sum_{t=1}^{T} r(s_t^A, a_t^A)$ weighs the terms of the sum, and the parameters of the policy are updated to have a policy that better produces state-action pairs that yield a greater accumulated reward. Greedy attacks and RL attacks are the two ideas underlying our formulation of attacks on the victim agent. Table 2 presents a comparison between enhanced EIIE and the adversary in RL attacks.

TABLE 2. Interactions between enhanced EIIE and the attack agent.

	Enhanced EIIE	Adversary
State	price data and volume data	states and actions of Enhanced EIIE
Action	asset allocation weight	how much volume to perturb
Reward	periodic logarithmic returns	negative value of Enhanced EIIE reward

For both greedy attacks and RL attacks, the state is unchanged if the adversary chooses to leave the state unperturbed. If the adversary decides to attack, the state changes to $s^{A'}$, which is equivalent to the original state with the perturbation added. Specifically,

$$\pi(s^A, a^A, r^A) = \begin{cases} s^A, & if\ a^A = 0 \\ s^{A'}, & if\ a^A > 0 \end{cases}$$
$$s^{A'} = s^A + arg\min_{\delta_s} \|\delta\| \tag{16}$$

where $a^A = 0$ and 1 represent a nonattack and attack, respectively.

The two attack methods slightly differ in their states, which we detail as follows.

D. STATES AND ACTIONS

In a greedy attack, the attacker decides when and how much perturbation is to be added; this decision depends only on price and volume data. As mentioned, for adversarial attacks to be applied on an RL agent, some conditions must be fulfilled. First, the perturbation must be imperceptible for the adversary to remain undetected by the victim agent. Thus, we set the two constraints of (1) a maximum attack volume ρ and (2) a total attack volume of $\sum_{t=1}^{T} \delta_t$ at each time step δ_t. If the total perturbation exceeds the constraint value, all the perturbations that exceed the constrained value will be set as 0.

$$s_t^{A'} = \begin{cases} s_t^A + \delta_t, & if\ \sum_{t=1}^{T} \delta_t <= \rho \\ s_t^A, & if\ \sum_{t=1}^{T} \delta_t >= \rho \end{cases} \tag{17}$$

Because of this design, the greedy adversary learns the optimal attack strategy with difficulty. The greedy adversary decides on its perturbations based only on price data and volume data; the adversary does not know how much more perturbations it can add. This adversary is, as its name implies, greedy because the adversary only cares about minimizing the instantaneous portfolio value. The adversary's decision-making process does not take a global view, which means that it may miss a possible optimal attack strategy of unleashing all perturbations at the last moment. Thus, we propose an RL attack strategy with more state information, which is the remaining attack volume ω. The total maximum attack volume is denoted as μ. Thus, the states are price data, volume data, and remaining attack volume; specifically, $s_t^A = (X_t, w_{t+1}, \omega)$, $where\ X_t = ((p^c, p^h, p^l), v)$. Per the setting described in the previous paragraph, perturbations are still clipped when the limitation ρ is exceeded. The action of both adversaries is the extent of perturbations $\delta = a^A \times max(v)$, where $a^A \in (0, 1)$. With the information of the remaining attack volume, the adversary's decision-making process is a continuous one, where the adversary takes a global view when allocating the attack volume.

E. REWARD FUNCTION AND POLICY NETWORK

The objective of the victim model is to maximize the final APV at the end of period T. The objective of the adversary is to elicit the worst performance in the victim model (i.e., to minimize the victim's portfolio value). Intuitively, the adversary can be trained by allowing it to learn from evaluations of the victim model. In such a training process, the reward of the adversary r^A can be defined as the negative absolute value of the reward r^V obtained by the victim model.

$$\begin{aligned} R^A(s_1^A, a_1^A, \cdots, s_T^A, a_T^A) &= -R^V(s_1^V, a_1^V, \cdots, s_T^V, a_T^V) \\ &= -\frac{1}{T}\sum_{t=1}^{T}(r_t^V - b) \end{aligned} \tag{18}$$

Intuitively, $E[R^A|s, a^A, \pi^A]$ can be maximized by estimating ∇E. The adversary agent can generate τ from $p(\theta|\phi)$ by differentiating the expected return with respect to ϕ, which is the parameter determining the distribution over θ.

$$\nabla_\phi E_\phi = \frac{1}{N}\sum_{n=1}^{N} \nabla_\phi(\log p(\theta|\phi))R(\tau) \tag{19}$$

Because the victim policy remains fixed, the two-policy Markov game is reduced to a one-player MDP $M_a = (S, A_a, T_a, R_a)$, which the attacker must solve. The state space of the adversary includes states, actions, and the rewards of the victim. The action space holds those adversarial perturbations that differ from the state and action spaces of the victim policy. In other words, the victim can be construed as the environment of the attacker policy.

$$T^A(s^A, a^A) = T(s, a^A, a^V) \tag{20}$$
$$R^A(s^A, a^A) = -R^V(s^V, a^V) \tag{21}$$

The training process of the adversary is detailed in Algorithm 1. For the training process of the adversary, some necessary input variables are predefined, and the training variable θ is initialized. At every iteration, states of the victim model and the adversary are initialized, and the remain attack volume is set as the maximum remaining attack volume μ. Subsequently, at each time step, the adversary may decide whether to add perturbations δ at the next state v by observing the current clean state. Before adding these perturbations at the next state, the adversary checks whether their attacks have exceeded the maximum attack volume, as stated in Equation (17). After the remaining attack volume is calculated, perturbations that the adversary has made will be added to the next state. The enhanced EIIE victim then allocates portfolio weights a^V by observing the perturbed states. The negative value of enhanced EIIE rewards constitutes the adversary's reward, which will be fed into the model. After the aforementioned series of interactions, the tuple (s_t^A, a_t^A, r_t^A) is stored in the buffer. This process is repeated several times for the adversary agent to attempt various strategies, and the corresponding samples are used to update the agent.

The adversary faces some limitations in our method, which pertain to the accessibility of adversaries, how much adversaries can affect the victim model, and the attacking ability of the adversary. These limitations are detailed as follows.

- We defined the attack method as a black-box attack, which means that the attacker can interact with the target model, with some limitations.
- The adversary cannot change the functioning of the victim's policy, which means that the victim model functions only with the pre-trained weight.
- The attacker can only add perturbations on those states that interact with the agent and the environment; the attacker cannot change the values of actions and rewards.

We crafted adversarial samples by adding imperceptible adversarial perturbation to maximize the loss of the original model. Attackers can choose from many construction techniques to make the most appropriate trade-off between attack success rate and computational cost. For our task, the system to be attacked is a trading strategy. Such an attack carries a very high computational cost compared with attacks in other image classification tasks. Thus, we aimed to lower the computational cost. Specifically, in our setting, after the adversary decides the attack volume δ_t, this attack volume is added to the volume of the five best bids and asks at the

Algorithm 1 Deterministic Policy Gradient-Like Agent for Victim Policy and RL Attack Policy

```
Input: state of victim model $s^V$, state of adversary $s^A$, training iterations $M$, attack magnitude constraint $\epsilon$, remain attack volume $\omega$, experience buffer $\mathcal{D}$
Output: attack policy $\theta$
1:  Initialize training variables $\theta$
2:  for $i = 1, 2, \ldots, M$ do
3:      Receive initial state of victim model $s_t^V = \{p_t^c, p_t^h, p_t^l, v_t\}$
4:      Receive initial state of adversary $s_t^A = \{p_t^c, p_t^h, p_t^l, v_t, \omega_t\}$
5:      Initialize the remain attack volume $\omega$ as the maximum remain attack volume $\mu$
6:      for $t = 1$ to T do
7:          Adversary select action $a_t^A = \pi_A(s_t^A)$
8:          Check if attack volume exceeds maximum attack volume
                $a_t^A = clip(a_t^A, a_t^A, \epsilon \cdot max(v_t))$          (22)
9:          Calculate the remain attack volume $\omega_t$ based on the action of the adversary and add it to the state
                $\omega_t = \omega_{t-1} - a_{t-1}^A$          (23)
10:         Add perturbations that the adversary made to clean state
                $S_{t+1}^{V'} = S_{t+1}^V + a_t^A$          (24)
11:         victim policy select action $a_t^V = \pi_V(S_{t+1}^{V'})$
12:         Evaluate victim policy and get reward $r_t^V$
13:         Feed negative reward of victim policy as adversary's reward $r_t^A = -r_t^V$
14:         Store $(s_t^A, a_t^A, r_t^A)$ in experience buffer $\mathcal{D}$
15:         if i mod N = 0 then
16:             Update the network:
                $\nabla_\theta J \approx \frac{1}{N}\sum_j \nabla_\theta \pi_A(s^{Aj}) \nabla_{a^A} R^A(s^j, a^A, a^V)|_{a^A=\pi(s^{Aj})}$          (25)
17:             Set $\mathcal{D}$ as empty
18:         end if
19:     end for
20: end for
```

subsequent time step v_{t+1}. The total volume is then canceled before the volume is matched to eliminate attack costs. This process can feasibly be applied under the 2016 trading mechanism but not under the present-day continuous trading mechanism. In the present-day mechanism, there a strong probability exists that limit orders with a reasonable price will be matched. The agent must then bear risks to be matched where the orders with prices near to the strike price have more opportunities to be matched.

F. ADVERSARIAL ATTACKS ON RL MODELS

Adversarial attackers against an RL model have employed the FGSM on deep Q-networks or the asynchronous advantage actor-critic (A3C) method. These methods are applied because they all have a function for estimating the most suitable action for the attacker to add perturbations with an explicit target. However, our victim network has no estimation network. Thus, we applied an RL agent as the attacker, and this attack agent observes the victim policy for a period to decide whether to attack.

The main idea underlying the FGSM is finding the weakness in the model gradient to conduct a one-step attack. The FGSM has been widely applied to computer vision classification models. The perturbation η is generated to attack fragile linear models. The perturbation obeys the following identity.

$$\eta = \epsilon(\nabla_x J(\psi, x, y)) \tag{26}$$

where ψ, x, y are the parameters, input, and the training target of the victim model, respectively. ϵ controls how much perturbations are produced, and $J(\psi, x, y)$ is the cost of training the victim model.

However, the FGSM is not always effective in all models. Scholars have thus proposed I-FGSM as an alternative iterative approach. I-FGSM generates perturbations in several steps to increase the likelihood of a successful attack; the total perturbation in I-FGSM is equal to that in a one-step FGSM. Recently, researchers have successfully used the FGSM and I-FGSM to attack RL agents. Thus, we used these algorithms as baselines in our experiments.

If an investor wishes to mislead other investors by placing fake orders, the quantity of fake orders may be potentially large. Generally, in the settings of attack classification models, only one or very few pixels are perturbed; however, trading models are impossible to attack by placing only one or very few orders. Thus, we limited the total perturbed volume of the five best bids and asks made by the adversary to be 0.0001% of the total trading volume. An adversary is likely to succeed if it attacks when prices fluctuate drastically because it will be less abnormal when the action of the victim model changes. Thus, the extent of perturbations is much larger (albeit still small) during periods when prices fluctuate drastically. The portfolio value of the victim model then decreases after a series of attacks.

VI. EXPERIMENTAL RESULTS

To demonstrate the performance of our model, we designed five experiments. In the first experiment, we demonstrated the attack performance of our proposed attack mechanisms by evaluating it against traditional attack methods; these attacks were applied to three portfolio management strategies, whose pre- and post-attack portfolio performance levels were compared. In the second experiment, we observed the relationship between attack scale and attack performance. We also applied the same attack architecture to three data sets because we expected adversaries to perform well on various data sets. To supplement the verification of our proposed method, we conducted the experiment with two additional data sets. They were the stock market data of the top 20 companies in the S&P 500 from Quandl and the data of euro cross rates with six other major currencies from Forex Academy.

A. DATA SETS AND EXPERIMENTAL SETTINGS

Before introducing the experiments, we describe the experimental settings, the data sets used, and the evaluation metrics used.

1) ASSET SELECTION

In the first and second experiments, assets were selected from all 11 underlying assets listed in Table 3. To meet the risk management requirements for major holdings in mutual funds, we selected the assets with the largest trading volumes in 2016, where a larger trading volume indicates greater market liquidity. Of the 11 assets, two belonged to exchange-traded funds (ETFs), and the remaining nine were stocks. In the fourth experiment, we selected the top 20 companies from the S&P 500 index. To prove that our proposed attack model can be used in the foreign exchange market, we performed the fifth experiment.

TABLE 3. Asset selection in the first experiment.

Code	Category	Name
00632R	ETF	YUANTA FUNDS/ETF
00633L		FBAM/ETF
2303	Stocks	United Microelectronics Corp.
2330		Taiwan Semiconductor Mfg. Co. Ltd.
2409		AU Optronics Corp
2448		EPISTAR Corporation
2881		Fubon Financial Holding Co Ltd.
2882		Cathay Financial Holding Co., Ltd.
2884		E.Sun Financial Holding Co Ltd
2891		CTBC Financial Holding Co Ltd
3481		Innolux Corp

2) DATA DESCRIPTION AND PREPROCESSING

The experimental data set comprised data drawn from the trade books and display books published by the Taiwan Stock Exchange (TWSE). Trade books mainly record trading information, and we used them to calculate price data and present book list information pertaining to the buy and sell side; furthermore, we extracted volume data from them. The fourth experimental data set was daily equity data from Quandl, which records a large amount of transaction information. We selected the stock market data of the top 20 companies in the S&P 500 index to be the assets in the experiment. We collected the daily foreign exchange market data from Forex Academy for the fifth experiment. These data contain cross exchange rates between the euro and the other six currencies.

We preprocessed the raw data. Price data were normalized to a narrow range by dividing the first closing price in the training window, per Equation 27.

$$p^c = \frac{p^c}{p_0^c}, \quad p^h = \frac{p^h}{p_0^c}, \quad p^l = \frac{p^l}{p_0^c} \tag{27}$$

Volume data were obtained by first summing the volume of five best bids and asks within 1 min for all matched and unmatched orders that we selected and then normalizing this sum by dividing the sum with the maximum volume in the training window, per Equation 28.

$$v = \frac{v}{\max(v)} \tag{28}$$

3) EXPERIMENTAL SETTINGS

Because trends vary between periods, we conducted a rolling-train test to ensure that the enhanced EIIE model performs well under various conditions. The rolling windows were set to be 3 months for training and 2 months for testing. The present rolling window was then shifted to 1 month to obtain the subsequent rolling window until the end of the test. For the adversary, the rolling window was set to be the training period. Also, we set the rolling windows to 3 years for training and 2 years for testing in the fourth experiment. The rolling window in the final experiment was set to 2 years for training and 1 year for testing.

4) EVALUATION METRICS

To reveal the performance of the adversaries and enhanced EIIE model, we used three evaluation metrics that are commonly used for financial investments. The first is APV, whose definition is written in Equation 29. The initial value for each asset was set to be 1.

$$APV_t = \frac{P_t}{P_0} \tag{29}$$

The second is maximum drawdown (MDD), which indicates downside risk over a given period. The MDD is defined as the maximum observed loss from a high point V_h to a low point V_l of the portfolio before a new peak is attained; this definition is written in Equation 30.

$$MDD = \frac{V_h - V_l}{V_h} \tag{30}$$

The third is the Sharpe ratio (SR), which indicates the risk–return relationship. The SR is defined as the average return R_p that was earned in excess of the risk-free rate R_f per unit of volatility σ_p; this definition is written in Equation 31.

$$SR = \frac{\mathbb{E}[R_p - R_f]}{\sigma_p} \tag{31}$$

B. OUR ADVERSARIES COMPARED WITH TRADITIONAL ATTACK MECHANISMS

In the first part of this experiment, we demonstrated that enhanced EIIE outperforms EIIE under the original setting stipulated by the EIIE's formulators. These two methods were compared with the traditional uniform constant rebalanced portfolios (UCRP) management method. The UCRP holds a portfolio that consists of a bucket of equally-weighted assets until the end of time. The differences between enhanced EIIE and EIIE are their features and model architectures. To ensure an impartial comparison between the two models, we trained EIIE with only price data and tuned the hyperparameters while maintaining the original settings for window size and training epochs. Enhanced EIIE was trained with both price and volume data.

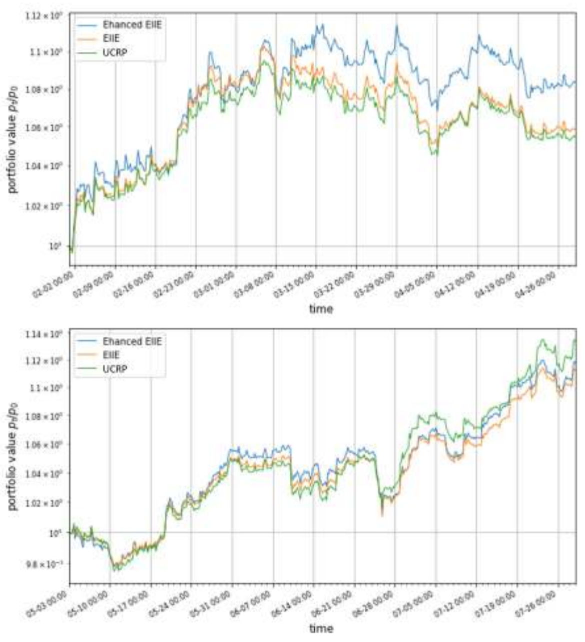

FIGURE 4. Comparison of the UCRP, EIIE, and enhanced EIIE. (a) One case where enhanced EIIE outperforms its counterparts. (b) One case where enhanced EIIE outperforms EIIE and is outperformed by the UCRP.

The figure 4 presents a comparison of the UCRP, EIIE, and enhanced EIIE. To make the price trend more obvious, we summed minute-level portfolio values into a 30-min portfolio value. Of the eight rolling models in total, we selected two that performed most differently to illustrate our findings. As indicated in Figure 4 (a), EIIE and the UCRP performed similarly and were outperformed by enhanced EIIE. Because the performance was indicated by accumulated return, the agent can maintain an excellent performance if it profits greatly in the middle periods and does not make too many losses during the later period. Most agents did so in the

TABLE 4. APV of trading agents and adversaries.

Period	UCRP	EIIE ($c = 0.25\%$)	Enhanced EIIE ($c = 0.25\%$)	FGSM ($c = 0.25\%$)	I-FGSM ($c = 0.25\%$)	Proposed Method-Greedy Attack ($c = 0$)	Proposed Method-Greedy Attack ($c = 0.25\%$)	Proposed Method-RL Attack ($c = 0$)	Proposed Method-RL Attack ($c = 0.25\%$)
Apr.- May	0.9593	**0.9550**	0.9801	0.6711	0.6469	0.9042	0.0413	0.9041	0.0411
May- June	0.9994	1.0028	1.0523	0.6421	0.6223	0.9003	0.0366	**0.9007**	**0.0371**
June- July	1.0120	1.0832	**1.0345**	0.6360	0.6193	0.9237	0.0358	0.9226	0.0356
July- Aug.	1.0200	1.0391	1.0398	0.6290	0.6178	0.9259	0.0312	0.9246	0.0309
Aug.- Sep.	1.0017	1.0150	1.0337	0.6463	0.6215	0.9177	0.0346	0.9178	0.0344
Sep.- Oct.	0.9804	1.0575	**0.9722**	0.6579	0.6362	0.9344	0.0401	0.9342	0.0395
Oct.- Nov.	0.9804	**0.9772**	0.9813	0.5939	0.5760	0.9302	0.0311	0.9301	0.0313
Nov.- Dec.	1.0212	1.1521	**1.1487**	0.6645	0.6561	0.9106	0.0376	0.9105	0.0376

eight rollings. The second case is indicated in Figure 4 (b), the UCRP outperformed enhanced EIIE and EIIE at few instances in all eight rolling results. Thus, enhanced EIIE still outperformed EIIE overall. Although these cases are presented with the 30-min portfolio value, both the enhanced EIIE and the adversary act every minute.

Because enhanced EIIE performed well, we applied various attack mechanisms to it. The left side of Table 4 shows the performance of trading agents and the UCRP, which act every minute. EIIE performed worse in the first and seventh rolling models than the UCRP did, but enhanced EIIE also failed to outperform EIIE in the three rolling models. Although enhanced EIIE has more features fed to it, it did not perform the best in all periods. Such performance suggests that it is well trained, meaning it can be a victim model of adversaries.

Thus, we compared four attack mechanisms, as indicated on the right side of Table 4. As mentioned in Chapter 3, to prove that our attack mechanisms outperform traditional attack methods, such as the FGSM and I-FGSM, we conducted experiments for every rolling model. The attributes of the FGSM and I-FGSM are such that both methods have a 100% attack success rate, meaning that attacks necessarily succeed if weaknesses are present in a neural network. I-FGSM, because it can add perturbations iteratively, reduced portfolio value more than the FGSM did. Our proposed adversaries also reduced portfolio value to approximately 0.92 without incurring transaction costs. However, in a real trading scenario, every single movement in position incurs a transaction cost. Also, our adversaries have learned to let the victim agent change portfolio weights drastically over many time steps for the agent to incur large transaction costs. Thus, our proposed two adversaries could reduce the portfolio value of the enhanced EIIE victim to almost the minimum value of 0, at a 0.25% transaction cost. For all rolling periods, the RL adversary performed worse in only one training period relative to the greedy adversary. This was because the RL adversary knows the attack volume that is left, which allows it to allocate the remaining amount. By contrast, the greedy adversary, having no access to such knowledge, only realizes that it has run out of attack volumes when the interaction concludes, making the greedy adversary allocate attacks less effectively than its RL counterpart does. The FGSM and I-FGSM reduced the portfolio value by 0.3842 and 0.4023 on average, respectively. Greedy and RL attacks performed similarly, reducing the portfolio value by 0.99 on average.

In the whole training period, we recorded the current portfolio value Pt at each time step. The focus in the portfolio management problem is the APV at the last time step. Thus, we multiplied all values in the series, per Equation 32, to compare the final performance of the four attack methods.

$$P_f = \prod_{t=1}^{n} P_t, \; P_0 = 1 \tag{32}$$

Figure 5(a) illustrates the APVs associated with enhanced EIIE, the FGSM, I-FGSM, greedy attack, and RL attack. Enhanced EIIE had a final portfolio value of > 1, which decreased to almost 0.2 under traditional attacks and to almost 0 (the lower bound of the portfolio value) under greedy and RL attacks. Furthermore, throughout the entire period, I-FGSM outperformed the FGSM, and greedy attack outperformed the RL attack. Because RL-based attacks performed much better than traditional attacks did, we also presented the results for only enhanced EIIE under traditional attacks in Figure 5(b). Specifically, the FGSM and I-FGSM made the enhanced EIIE victim agent lose considerable capital when the victim agent allowed the portfolio value to drop; the FGSM and I-FGSM could do so because they learn how to add perturbations based on the gradient of the target model.

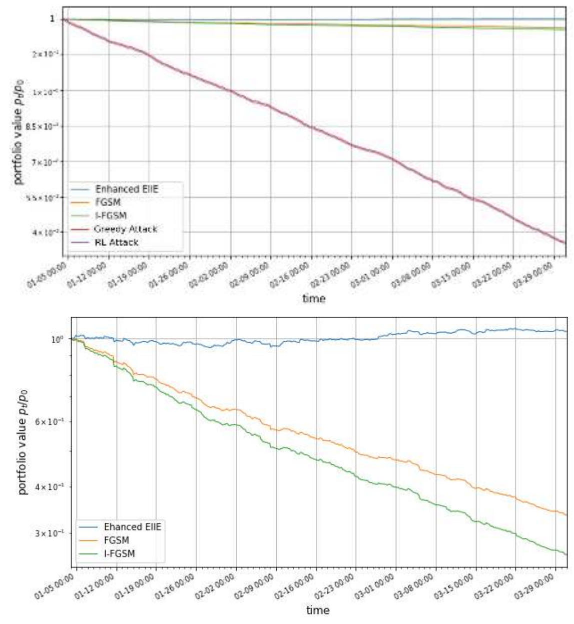

FIGURE 5. Example of APV comparison at first rolling. (a) APVs of enhanced EIIE, traditional attacks, and RL-based attacks. (b) Partial-observation APVs of only enhanced EIIE and traditional attacks.

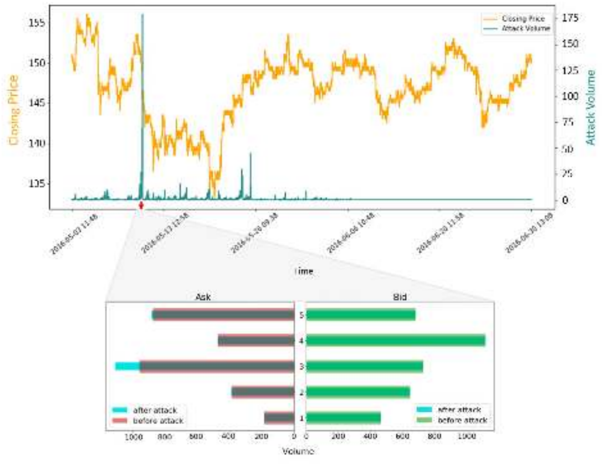

FIGURE 6. Relation between price and perturbation at second rolling.

We subsequently used a case to observe how the adversary decides on when and how fiercely to attack. Figure 6 presents the perturbations made by the RL adversary as well as the closing price. The x axis represents the time step, the left y axis represents the closing price of the given attacked target, and the right y-axis represents the number of fake orders being placed in the market. The figure next to the main figure represents the volume of the five best asks and bids before and after the attack. Figure 6 illustrates the relationship between price trend and perturbations for asset code 2884 from May 1 to June 30. In this case, the adversary only adds perturbations in ask orders. On May 5, 2016, when the closing price dropped sharply, the adversary placed a large quantity of ask orders in the first half of time steps, strongly indicating that someone believes that the asset price will start rising. This large quantity was also attributable to an adversary choosing to add perturbations to other assets, where the adversary aims to deceive investors into making a suboptimal decision. The adversary thus induced the trading agent to change its decision. When the price of the asset begins falling at this time step, the enhanced EIIE victim makes a loss due to this attack, and more fake orders with small lots are placed in the following period of decrease. During this period, the enhanced EIIE victim also changes its decision when the closing price is close to the global minimum value in the entire time step. It changes the asset weight to approximately 0.2. Due to the attack, the enhanced EIIE victim sets relatively high price weights on this asset and sells lots at a relatively low price. Nevertheless, the enhanced EIIE victim is still influenced to set different weights with nonperturbed noise; this is because the performance of every asset changes not only depending on the perturbations added to that asset but also on changes in the weights of other assets.

C. RELATIONSHIP BETWEEN ATTACK PERFORMANCES AND THE EXTENT OF PERTURBATIONS

This experiment demonstrated the relationship between attack volume and portfolio value. In a classification problem, the adversary is tasked with attacking the victim model with very perturbations. Greater perturbations necessarily result in a less accurate victim model. Specifically, because the trading volume is highly influential in a financial market, attacks are more likely to succeed when more limit orders are placed.

Our strategy was to place a small quantity of limit orders and subsequently cancel them before they are matched. Despite the fact that we only needed to take on the risk of being matched, if the limit orders being placed in the market were too large, the attacker would be detected by investors. This complicated our task. For the classification task, the adversary's only objective is to deceive the classifier; however, we needed to attack the market without being detected by the trading agent and by other investors.

We limited the total maximum attack volume to 0.0001%, 0.00005%, 0.000025%, and 0.000001% of the training set (Table 5). For example, if, in a period, the total volume of the five best bids and asks is 60 billion, the maximum attack volume is 60,000. In other words, the average maximum attack volume for each batch, each asset, each training window, and for each bid and ask was 1 lot of stocks.

The three attack scales performed similarly (difference $\approx$ 0.0001). From Table 5, the perturbations generated by the FGSM and I-FGSM at different attack scales were such that victim model accuracy was inversely related to perturbation size. Our proposed adversaries could reduce the portfolio value to approximately 0.9185 at no transaction cost and to 0.0361 at a 0.25% transaction cost.

D. ATTACKS ON VARIOUS DATA SETS – STOCK DATA FROM TWSE

In this experiment, we applied greedy and RL attacks to different data sets. As mentioned, the assets used in

TABLE 5. Average attack performance at perturbation limitations of 0.0001%, 0.00005%, 0.000025%, and 0.000001%.

Attack Scale	Enhanced EIIE ($c = 0.25\%$)	FGSM ($c = 0.25\%$)	I-FGSM ($c = 0.25\%$)	Proposed Method-Greedy Attack ($c = 0$)	Proposed Method-Greedy Attack ($c = 0.25\%$)	Proposed Method-RL Attack ($c = 0$)	Proposed Method-RL Attack ($c = 0.25\%$)
0.000100%	1.0278	0.6426	0.6426	0.9184	0.0360	0.9183	0.0359
0.000050%	1.0278	0.6427	0.6427	0.9185	0.0361	0.9184	0.036
0.000025%	1.0278	0.6428	0.6428	0.9186	0.0362	0.9184	0.036
0.000001%	1.0278	0.6431	0.6431	0.9264	0.0429	0.9262	0.0428

experiment 1 and 2 were selected from all underlying assets. To prove that the attack architecture can be applied to various portfolios, we selected two more buckets of assets to form the portfolio; all assets belonged to stocks. All assets in the first and second buckets were electronic and semiconductor stocks, respectively. Because the electronic and semiconductor industries dominate Taiwan's economy, many investors invest in these stocks on the TWSE.

We also evaluated performance in terms of APV. However, portfolio value does not reflect risk, whereas the MDD and SR do. In general, a good portfolio has a high APV, low MDD, and high SR. Thus, an effective adversary not only reduces portfolio value but also induces large fluctuations in different portfolios. Thus, in this experiment, we analyzed the APV, MDD, and SR of the portfolio comprising assets from electronic stocks and semiconductor stocks.

Table 6 lists the MDD and SR but not APV of five models in eight periods. Enhanced EIIE outperformed EIIE and UCRP with respect to APV in 20 of 24 stocks, and the greedy attack performed better than the RL attack did in only one period for harming APV. We trained enhanced EIIE to perform best with respect to APV but not MDD or SR. Thus, enhanced EIIE may yield poorer performance in MDD and SR than in APV. In our experiments, enhanced EIIE had a fairly poor MDD in the absence of attacks; this means that the fluctuation and drawdown are small and that the return at each time step is relatively steady. Attacks result in large fluctuations in APV. Both greedy and RL attacks reduced MDD performance to almost its worst possible level (MDD $= 1$), reducing MDD performance more than the traditional attack methods did. Specifically, when MDD $= 1$, the lowest portfolio value before a new high is approximately 0. This result indicates that adversaries adopted the strategy of letting prices drop. Specifically, after fake orders are placed in the market, we expect that investors will be influenced into making suboptimal decisions. Through these transactions, the price of assets change, and the portfolio value decreases while investors are unaware of the attack.

A larger (i.e., better performing) SR means that profit is yielded under more acceptable risks. The average SR of UCRP and EIIE in eight rollings were 0.0223 and 0.0311, respectively (not presented in Table 6). Enhanced EIIE outperformed EIIE and UCRP in 18 of the 24 rollings. The attacks considerably reduced SR in all periods due to the damage done to APV and MDD. Specifically, attacks increase risk and lower returns, making SR low (i.e., very undesirable). Similar to their performance in the aforementioned experiments, the FGSM and I-FGSM consistently hurt SR (average SR after the attack: -0.0563) and performed much better than our two adversaries did (average SR after the attack: -0.5099).

E. ATTACKS ON VARIOUS DATA SETS – STOCK DATA FROM QUANDL

To examine the vulnerability of RL models in portfolio management in front of the RL-based adversarial attack method in other major stock markets, we further tested them by using the daily data of the top 20 S&P 500 stocks from 2010 to 2017. The data fields include the opening, highest, lowest, and closing prices and the daily trading volume. We used this data set to conduct the experiment, which was evaluated using the three metrics described on p.11, section VI A, subsection 4.

We formed our testing portfolios over 2013–2017 with five nonoverlapping 1-year periods. Table 7 summarizes the results of the UCRP (the baseline method), Enhanced EIIE (the victim model), and the victim model after RL attack in terms of the three evaluation metrics.

As indicated in Table 7, the RL-attack of the top 20 S&P 500 portfolio yielded pronounced results for all three metrics in every test period. All the APVs decreased to < 1 (i.e., had negative returns) after the attack. The effects of the portfolios' value were large across all testing periods. The MDD results provide insight into the success of the attack. For example, the MDD of 0.9920 in 2014 was the key reason for the portfolio losing nearly all its value. The Sharpe ratios were all negative after the attack, and the smaller absolute values

TABLE 6. MDD and SR of trading agents and adversaries with different data sets.

		MDD					SR				
Dataset	Period	Enhanced EIIE	FGSM	I-FGSM	Greedy Attack	RL Attack	Enhanced EIIE	FGSM	I-FGSM	Greedy Attack	RL Attack
All	1	0.0503	0.2708	0.3031	0.9796	**0.9846**	0.0585	0.0024	0.0024	-0.2615	**-0.3239**
	2	0.0271	0.2994	0.3287	0.9709	**0.9813**	0.1014	0.0024	0.0023	-0.5100	**-0.5987**
	3	0.0378	0.1807	0.2182	0.8766	**0.8801**	0.0316	0.0012	0.0009	-0.6192	**-0.6287**
	4	0.0284	0.2477	0.2551	0.9786	**0.9798**	0.0934	0.0020	0.0020	-0.3681	**-0.5136**
	5	0.0214	0.2704	0.2977	0.9988	0.9843	0.0836	0.0018	0.0011	-0.3310	**-0.4193**
	6	0.0382	0.2178	0.2398	0.9828	**0.9871**	0.0840	0.0019	0.0018	-0.3168	**-0.3525**
	7	0.0302	0.2716	0.2965	0.9523	0.9428	0.0554	0.0023	0.0020	-0.7031	-0.7016
	8	0.0359	0.2293	0.2326	0.9287	**0.9291**	0.0691	0.0016	0.0017	-0.6002	**-0.7097**
	Avg	0.0337	0.2485	0.2713	0.9585	0.9606	0.0721	0.0020	0.0018	-0.4637	-0.5310
Electronic Stocks	1	0.0542	0.0651	0.1051	0.9865	**0.9889**	-0.0073	-0.0375	-0.0417	-0.7106	**-0.7191**
	2	0.0575	0.4576	0.4875	0.9815	0.9809	0.0073	-0.4137	-0.4216	-0.5106	**-0.5284**
	3	0.0466	0.2815	0.3343	0.9746	**0.9807**	0.0013	-0.0816	-0.1020	-0.4290	**-0.4314**
	4	0.0771	0.3403	0.3794	0.9683	0.9566	-0.0169	-0.0717	-0.1024	-0.3284	**-0.3695**
	5	0.2003	0.4131	0.4323	0.8921	0.8137	-0.0062	-0.0082	-0.0245	-0.4295	-0.4123
	6	0.0422	0.4522	0.4622	0.9184	0.9142	0.0016	-0.1346	-0.1543	-0.5251	-0.5136
	7	0.0213	0.3117	0.3518	0.9632	**0.9652**	0.0026	-0.0913	-0.1115	-0.3593	**-0.3780**
	8	0.0355	0.4295	0.4451	0.9645	**0.9693**	0.0058	-0.1098	-0.1222	-0.2119	**-0.3081**
	Avg	0.0668	0.3439	0.3742	0.9561	0.9462	-0.0015	-0.1185	-0.1350	-0.4231	-0.4226
Semi-conductor Stocks	1	0.0578	0.3136	0.3507	0.9621	**0.9701**	-0.0154	-0.1803	-0.1917	-0.6092	**-0.6468**
	2	0.0575	0.1842	0.1831	0.9843	0.9799	0.0073	-0.1016	-0.1215	-0.6269	**-0.6532**
	3	0.0391	0.0692	0.1092	0.9658	0.9543	0.0055	-0.0004	-0.0009	-0.5989	**-0.5998**
	4	0.0365	0.0962	0.0984	0.9881	0.9832	0.0049	-0.0031	-0.0058	-0.5622	**-0.5962**
	5	0.0838	0.1638	0.1838	0.9951	**0.9997**	0.0047	0.0000	0.0000	-0.5623	-0.5517
	6	0.0436	0.3107	0.3238	0.9032	**0.9104**	-0.0090	-0.0097	-0.0099	-0.5487	-0.5363
	7	0.0483	0.1489	0.1512	0.9504	**0.9532**	-0.0053	-0.0304	-0.0418	-0.5433	-0.5124
	8	0.0052	0.0551	0.0563	0.9701	**0.9703**	0.0286	-0.0015	-0.0065	-0.5521	**-0.5992**
	Avg	0.0464	0.1672	0.1821	0.9649	**0.9651**	0.0027	-0.0409	-0.0473	-0.5754	**-0.5870**

TABLE 7. Experimental results of top 20 stocks S&P 500 portfolios (c = 0.25%).

	APV			MDD			SR		
Testing period	UCRP	Enhanced EIIE	RL attack	UCRP	Enhanced EIIE	RL attack	UCRP	Enhanced EIIE	RL attack
2013	1.2098	1.7502	**0.7976**	0.2230	0.2861	**0.4042**	0.0671	0.1231	**-0.0340**
2014	2.7588	2.6251	**0.0086**	0.0818	0.0827	**0.9920**	0.1719	0.1673	**-0.0645**
2015	1.4053	1.4076	**0.7643**	0.1080	0.1081	**0.4786**	0.1191	0.1197	**-0.0080**
2016	1.3068	1.3195	**0.4184**	0.2215	0.2224	**0.7728**	0.0881	0.0897	**-0.0113**
2017	1.3704	3.7791	**0.5235**	0.1404	0.4390	**0.6197**	0.1190	0.1550	**-0.0657**
Avg.	1.6102	2.1763	**0.5025**	0.1550	0.2277	**0.6535**	0.1130	0.1309	**-0.0367**

reflected the high volatility of the portfolios. Overall, our RL attack was effective and yielded clear results.

F. ATTACKS ON VARIOUS DATA SETS – FOREIGN EXCHANGE DATA

In addition to the equity portfolios, we investigated foreign exchange portfolios. This experiment involved the daily euro cross rates with six other major currencies, such as the US dollar, Australian dollar, and Canadian dollar, from 2008 to 2020. The data fields, control methods, and observed indicators were the same as those of the top 20 S&P 500 experiment.

For the foreign exchange portfolios, the decreases in APVs were not as severe as those in the equity portfolios. However, the APVs decreased in all testing periods after the attack.

TABLE 8. Experimental results of foreign exchange portfolios (c = 0.25%).

	APV			MDD			SR		
Testing period	UCRP	Enhanced EIIE	RL attack	UCRP	Enhanced EIIE	RL attack	UCRP	Enhanced EIIE	RL attack
2011/1/1-2012/12/31	0.9371	0.9390	**0.9363**	0.1136	0.1138	**0.1340**	-0.0261	-0.0251	-0.0183
2012/1/1-2013/12/31	1.0809	1.0823	**1.0435**	0.0786	0.0781	0.0502	0.0411	0.0418	**0.0323**
2011/1/1-2012/12/31	0.9371	0.9390	**0.9363**	0.1136	0.1138	**0.1340**	-0.0261	-0.0251	-0.0183
2012/1/1-2013/12/31	1.0809	1.0823	**1.0435**	0.0786	0.0781	0.0502	0.0411	0.0418	**0.0323**
2013/1/1-2014/12/31	1.0211	1.0240	**0.9773**	0.0813	0.0818	0.0472	0.0154	0.0168	**-0.0193**
2014/1/1-2015/12/31	0.9427	0.9635	**0.9267**	0.1217	0.1476	**0.1588**	-0.0238	-0.0038	-0.0195
2015/1/1-2016/12/31	1.0335	1.0345	**0.9721**	0.0795	0.0793	0.0654	0.0218	0.0222	**-0.0133**
2016/1/1-2017/12/31	1.0636	1.0643	**1.0289**	0.0823	0.0826	**0.0904**	0.0368	0.0370	**0.0193**
2017/1/1-2018/12/31	1.0727	1.1262	**0.9765**	0.0805	0.1114	0.0884	0.0434	0.0517	**-0.0083**
2018/1/1-2019/12/31	0.9598	1.0213	**0.9417**	0.0551	0.0865	**0.0921**	-0.0365	0.0150	**-0.0413**
2019/1/1-2020/12/31	1.0130	1.0153	**0.9979**	0.0523	0.0517	**0.0620**	0.0129	0.0146	**0.0011**
Avg.	1.0030	1.0086	**0.9712**	0.0949	**0.1001**	0.0911	0.0057	0.0104	**-0.0076**

Although the MDD results were not as perfect as the APV results, the post-attack MDDs of more than half of the testing periods increased. Similarly, the Sharpe ratios of the portfolios decreased in the victim model in all the testing periods except the first one. In summary, the RL attack was effective overall on the foreign exchange portfolios, although the effects were not as noticeable as those on the equity portfolios.

The RL attack was effective in general, with clear results on the equity portfolios. The different natures of the two asset classes may have caused the differences in effectiveness. Further investigation of the relationship between the properties of asset classes might improve model effectiveness.

VII. CONCLUSION

We proposed two RL-based attack methods, greedy and RL attacks, to attack an RL portfolio management model. Many researchers have formulated attacks on classification models, but few have investigated the vulnerability of RL models. With our attack architecture, the adversary can learn when and to what extent perturbations should be added. Our architecture also results in no costs being incurred from attacking the portfolio management model. To make perturbations imperceptible, we limited the total attack volume to 0.0001% of the total trading volume. Thus, the adversary learns how to precisely allocate perturbations to most effectively minimize the final portfolio value.

To prove that our adversaries perform well, we used 2016 trading data provided by the TWSE to train both the enhanced EIIE model and the adversaries. The experimental results demonstrated that our proposed approaches performed better than traditional attack methods did with respect to the APV, SR, and MDD. Compared with the FGSM and I-FGSM, both greedy and RL attacks performed 40%, 50%, and 80% better with respect to APV, SR, and MDD, respectively. The last two experiments proved that our attack method can be applied to the S&P 500 and foreign exchange markets.

Our main contributions are listed as follows.

- We demonstrated the vulnerability of an RL agent in portfolio management. An attack may cause the trading strategy to collapse, constituting an opportunity for the attacker to profit.
- Because of the RL architecture, our greedy attack, and RL attack systems can learn when and to what extent perturbations should be added; traditional attack systems cannot learn these two pieces of knowledge (on scale and timing) simultaneously.
- As far as we know, ours is the first study to use RL-based attack algorithms to attack a portfolio management model in the financial market.

Further research is warranted, especially considering the catastrophic consequences of such attacks. In this study, we only focused on the severity of attacks in the financial market. In the future, researchers should make the trading model more robust by having the model learn how to defend against such attacks. In addition, the performance levels of our models only depend on the evaluation of the victim model. If other attack evaluation methods are considered, the performance can be improved. Furthermore, different matching mechanisms may require different attack methods. The 2016 matching mechanism features the call auction, but the present-day mechanism features continuous trading. If the same attack architecture is applied to continuous trading, fake orders are highly likely to be matched.

REFERENCES

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*. [Online]. Available: http://arxiv.org/abs/1412.6572

[2] 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

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*. [Online]. Available: http://arxiv.org/abs/1312.6199

[4] M. A. A. Milton, "Evaluation of momentum diverse input iterative fast gradient sign method (M-DI2-FGSM) based attack method on MCS 2018 adversarial attacks on black box face recognition system," 2018, *arXiv:1806.08970*. [Online]. Available: http://arxiv.org/abs/1806.08970

[5] J. Chen, Y. Wu, X. Xu, Y. Chen, H. Zheng, and Q. Xuan, "Fast gradient attack on network embedding," 2018, *arXiv:1809.02797*. [Online]. Available: http://arxiv.org/abs/1809.02797

[6] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," 2017, *arXiv:1705.07204*. [Online]. Available: http://arxiv.org/abs/1705.07204

[7] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.

[8] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*. [Online]. Available: http://arxiv.org/abs/1607.02533

[9] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.

[10] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.

[11] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.

[12] M. Sadeghi and E. G. Larsson, "Physical adversarial attacks against end-to-end autoencoder communication systems," *IEEE Commun. Lett.*, vol. 23, no. 5, pp. 847–850, May 2019.

[13] G. Gondim-Ribeiro, P. Tabacof, and E. Valle, "Adversarial attacks on variational autoencoders," 2018, *arXiv:1806.04646*. [Online]. Available: http://arxiv.org/abs/1806.04646

[14] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proc. MILCOM IEEE Mil. Commun. Conf.*, Nov. 2016, pp. 49–54.

[15] Y. Han, B. I. P. Rubinstein, T. Abraham, T. Alpcan, O. De Vel, S. Erfani, D. Hubczenko, C. Leckie, and P. Montague, "Reinforcement learning for autonomous defence in software-defined networking," in *Decision and Game Theory for Security*, L. Bushnell, R. Poovendran, and T. Ba ar, Eds. Cham, Switzerland: Springer, 2018, pp. 145–165.

[16] Y. Huang and Q. Zhu, "Deceptive reinforcement learning under adversarial manipulations on cost signals," in *Decision and Game Theory for Security*, L. Bushnell, R. Poovendran, and T. Ba ar, Eds. Cham, Switzerland: Springer, 2019, pp. 217–237.

[17] V. Gallego, R. Naveiro, and D. R. Insua, "Reinforcement learning under threats," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 9939–9940.

[18] 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.

[19] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," 2019, *arXiv:1909.02583*. [Online]. Available: http://arxiv.org/abs/1909.02583

[20] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," 2017, *arXiv:1702.02284*. [Online]. Available: http://arxiv.org/abs/1702.02284

[21] L. Hussenot, M. Geist, and O. Pietquin, "CopyCAT: Taking control of neural policies with constant attacks," 2019, *arXiv:1905.12282*. [Online]. Available: http://arxiv.org/abs/1905.12282

[22] A. Gleave, M. Dennis, C. Wild, N. Kant, S. Levine, and S. Russell, "Adversarial policies: Attacking deep reinforcement learning," 2019, *arXiv:1905.10615*. [Online]. Available: http://arxiv.org/abs/1905.10615

[23] G. Clark, M. Doran, and W. Glisson, "A malicious attack on the machine learning policy of a robotic system," in *Proc. 17th IEEE Int. Conf. Trust, Secur. Privacy Comput. Commun./12th IEEE Int. Conf. Big Data Sci. Eng. (TrustCom/BigDataSE)*, Aug. 2018, pp. 516–521.

[24] A. Gupta and Z. Yang, "Adversarial reinforcement learning for observer design in autonomous systems under cyber attacks," 2018, *arXiv:1809.06784*. [Online]. Available: http://arxiv.org/abs/1809.06784

[25] H. Markowitz, *Portfolio Selection: Efficient Diversification Investments*, vol. 16. New York, NY, USA: Wiley, 1959.

[26] N. H. Hakansson and W. T. Ziemba, "Capital growth theory," *Handbooks Oper. Res. Manage. Sci.*, vol. 9, pp. 65–86, Jan. 1995.

[27] J. L. Kelly, Jr., "A new interpretation of information rate," in *The Kelly Capital Growth Investment Criterion: Theory and Practice*. Singapore: World Scientific, 2011, pp. 25–34.

[28] A. Agarwal, E. Hazan, S. Kale, and R. E. Schapire, "Algorithms for portfolio management based on the Newton method," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 9–16.

[29] D.-J. Huang, J. Zhou, B. Li, S. C. H. Hoi, and S. Zhou, "Robust median reversion strategy for online portfolio selection," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 9, pp. 2480–2493, Sep. 2016.

[30] B. Li, S. C. H. Hoi, and V. Gopalkrishnan, "CORN: Correlation-driven nonparametric learning approach for portfolio selection," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–29, Apr. 2011.

[31] B. Li and S. C. Hoi, "Online portfolio selection: A survey," *ACM Comput. Surv.*, vol. 46, no. 3, pp. 1–36, 2014.

[32] Y. Huang, L. F. Capretz, and D. Ho, "Neural network models for stock selection based on fundamental analysis," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.

[33] M.-Y. Day and J.-T. Lin, "Artificial intelligence for ETF market prediction and portfolio optimization," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2019, pp. 1026–1033.

[34] W. Shin, S.-J. Bu, and S.-B. Cho, "Automatic financial trading agent for low-risk portfolio management using deep reinforcement learning," 2019, *arXiv:1909.03278*. [Online]. Available: http://arxiv.org/abs/1909.03278

[35] F. Soleymani and E. Paquet, "Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath," *Expert Syst. Appl.*, vol. 156, Oct. 2020, Art. no. 113456.

[36] L. Weng, X. Sun, M. Xia, J. Liu, and Y. Xu, "Portfolio trading system of digital currencies: A deep reinforcement learning with multidimensional attention gating mechanism," *Neurocomputing*, vol. 402, pp. 171–182, Aug. 2020.

[37] J. Wang, Y. Li, and Y. Cao, "Dynamic portfolio management with reinforcement learning," 2019, *arXiv:1911.11880*. [Online]. Available: http://arxiv.org/abs/1911.11880

[38] Y. Ye, H. Pei, B. Wang, P.-Y. Chen, Y. Zhu, J. Xiao, and B. Li, "Reinforcement-learning based portfolio management with augmented asset movement prediction states," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 1, pp. 1112–1119.

[39] Y. Zhang, P. Zhao, B. Li, Q. Wu, J. Huang, and M. Tan, "Cost-sensitive portfolio selection via deep reinforcement learning," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 10, 2020, doi: 10.1109/TKDE.2020.2979700.

[40] 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

YU-YING CHEN received the B.S. degree in information management and finance and the M.S. degree in information management from National Chiao Tung University, Hsinchu, Taiwan, in 2018 and 2020, respectively. Her research interests include reinforcement learning, adversarial attack, and financial technology .

CHIAO-TING CHEN received the B.S. degree in information management and finance and the M.S. degree in information management from National Chiao Tung University, Hsinchu, Taiwan, in 2018 and 2020, respectively, where she is currently pursuing the Ph.D. degree in computer science. Her research interests include deep learning, artificial intelligence, knowledge graph, graph embedding, and financial technology.

CHUAN-YUN SANG received the B.S. degree from the Department of Information Management and Finance, National Taiwan University of Science and Technology, Taipei, Taiwan, in 2020. She is currently pursuing the M.S. degree in information management with National Chiao Tung University, Hsinchu, Taiwan. Her research interests include deep learning, artificial intelligence, reinforcement learning, and financial technology.

YAO-CHUN YANG received the B.S. degree from the Department of Industrial Engineering and Engineering Management, National Tsing Hua University, Hsinchu, Taiwan, in 2019. He is currently pursuing the M.S. degree in information management with National Chiao Tung University, Hsinchu. His research interests include deep learning, artificial intelligence, knowledge graph, graph embedding, and financial technology.

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 Chiao Tung University, Hsinchu. He has authored more than 50 papers published in the related international journals and conferences. He is also the Principal Investigator of the MOST Financial Technology Innovation Industrial-Academic Alliance and several cooperation projects with leading companies in Taiwan. His research interests include artificial intelligence, deep learning, recommender systems, computer vision, and financial technology.