World Wide Web
https://doi.org/10.1007/s11280-022-01112-4

Profit-based deep architecture with integration of reinforced data selector to enhance trend-following strategy

Yang Li[1] · Zibin Zheng[1,2] · Hong-Ning Dai[3] · Raymond Chi-Wing Wong[4] · Haoran Xie[5]

Received: 10 January 2022 / Revised: 19 June 2022 / Accepted: 27 September 2022

Abstract
Despite the popularity of trend-following strategies in financial markets, they often lack adaptability to the emerging varied markets. Recently, deep learning (DL) methods demonstrate the effectiveness in stock-market analysis. Thus, the application of DL methods to enhance trend-following strategies has received substantial attention. However, there are two key challenges to be solved before the adoption of DL methods in enhancing trend-following strategies: (1) how to design an effective data selector to include more related data? (2) how to design a profit-based model to enhance strategies? To address these two challenges, this paper contributes to a new framework, namely profit-based deep architecture with the integration of reinforced data selector (PDA-RDS) to improve the effectiveness of DL methods. In particular, profit-based deep architecture (PDA) integrates a dynamic profit weight and a focal loss function to obtain high profits. In addition, reinforced data selector (RDS) is constructed to select high-quality training samples and a training-aware immediate reward is designated to improve the effectiveness of RDS. Extensive experiments on both U.S. and China stock-market datasets demonstrate that PDA-RDS outperforms the state-of-the-art baseline methods in terms of higher cumulative percentage rate and average percentage rate, both of which are crucial to investment strategies.

Keywords Deep learning · Transfer learning · Reinforcement learning · Trend-following strategy · Data selection

1 Introduction

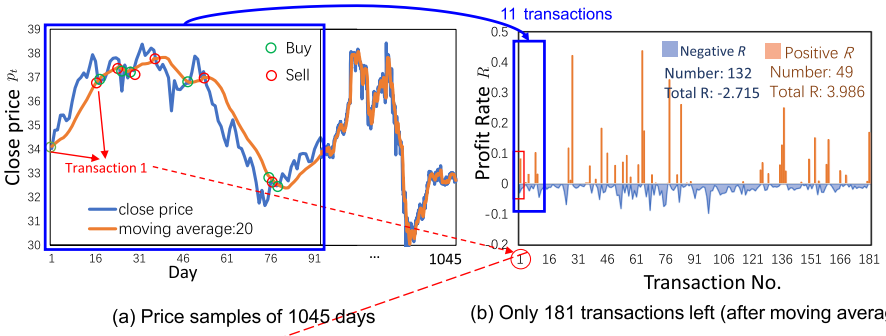
Trend-Following Strategy (TFS) [1] is one of the most typical investment strategies and may be one of the most popular strategies in financial markets. TFS attempts to obtain gains through analyzing the *momentum* of a stock (or stocks) in an up trend or a down

This article belongs to the Topical Collection: *Web-based Intelligent Financial Services*
Guest Editors: Hong-Ning Dai, Xiaohui Tao, Haoran Xie, and Miguel Martinez.

✉ Yang Li
liyang99@mail2.sysu.edu.cn

Extended author information available on the last page of the article

Published online: 14 October 2022



(a) Price samples of 1045 days

(b) Only 181 transactions left (after moving average)

Transaction No.	1	2	3	4	5	6	7	8	9	10	11	*Acc*	*Profit*
R (1/1000)	81	-7.8	4.7	-1.1	-1.7	-1.6	-1	-2	10	27	-5.6		
Ground truth	1	-1	1	-1	-1	-1	-1	-1	1	1	-1		
Classifier1	-1	-1	1	-1	-1	-1	1	-1	1	-1	-1	**0.727**	0.0137
Classifier2	1	1	-1	1	1	-1	-1	1	1	1	-1	0.454	**0.1054**

(c) Model with higher *Acc* having lower *Profit*

Fig. 1 Stock NYSE:WMB with a moving average TFS. It contains 1,045 days and 181 transactions when applying a moving-average TFS. Moreover, positive R represents positive profit and negative R represents negative profit

trend in terms of stock prices and then making decisions (i.e., purchasing or selling stocks). However, conventional TFS techniques, such as the moving average TFS [2], momentum indicator TFS [3], and trendlines & chart patterns [4] can only capture certain features from historical stock data while cannot adapt to the emerging varied stock markets.

Deep learning (DL) technologies have achieved tremendous success in many fields in the past few years including natural language processing [5], image classification [6], financial time-series data analysis [7], and other fields [8–10]. Recently, DL approaches have also been adopted in the stock-market analysis [11, 12]. Particularly, Recurrent Neural Network (RNN) and its variations show superior performance than conventional financial analytical methods, e.g., Autoregressive Integrated Moving Average (ARIMA) and Generalized Auto Regressive Conditional Heteroskedasticity (GARCH) [12]. Meanwhile, DL methods have also been used in data analysis of financial text data such as economic news [13] and social media [14] to enhance the stock prediction when integrating with the stock data analysis. Despite the above research advances, the application of DL techniques to enhance TFS in stock markets still faces two key challenges.

Challenge 1. How to design an effective data selector to include more related TFS data? DL methods achieve the outstanding performance through learning (or extracting) features from massive data. However, it is difficult to obtain enough TFS data for training DL models so as to improve TFS. Figure 1 shows an example of using a *moving-average* TFS on the stock NYSE:WMB, which is a stock in the New York Stock Exchange (NYSE). The *moving average* essentially calculates the average of prices over a period time to determine the trend of a stock (e.g., we consider 20-day moving average for the period). Generally speaking, we buy the stock if the price P_{buy} is greater than the moving average; we sell the stock if the price P_{sell} is lower than the moving average. After a moving-average TFS

is applied to original 1,045 price samples as shown in Figure 1(a), only 181 data samples (i.e., transactions) are left (i.e., 17.2% of total 1,045 samples), as shown in Figure 1(b). Different from price-prediction tasks which can use all price samples for training, *it is challenging to DL methods for TFS-enhancing tasks due to insufficient TFS samples*. The solution to this problem is to introduce more relevant data samples to the dataset so as to improve the training effect of DL methods [7, 15, 16]. However, most existing studies separate the data selection from model training processes, thereby irrelevant data samples being used to train DL models. Consequently, the separation of data selection and model training processes leads to the poor performance.

Challenge 2. How to design a profit-based model for TFS? There is also an unbalanced-sample problem when calculating the profits after executing a TFS. The profit rate R_i for transaction i is defined as follows:

$$R_i = \frac{p_{\text{sell}} - p_{\text{buy}}}{p_{\text{buy}}} - \text{TC}, \tag{1}$$

where TC represents the transaction cost including transaction fee and transaction slippage. Figure 1(b) shows that there is an unbalance between negative profit values and positive profit values, e.g., there are 132 negative profit values far greater than 49 positive profit values while 49 positive profits lead to a cumulative positive profit rate $R = 3.986 - 2.715 = 1.271$. Conventional DL methods often consider the stock prediction as a classification problem [11, 12] or a regression problem [7, 17]. However, the better performance (such as higher accuracy and smaller error) of DL methods does not necessarily bring the higher profits obtained for the real stock in contrast to the common sense. Take Figure 1(c) as an example, where we consider prices of a real stock from NYSE:WMB with different transactions, i.e., 1, 2, ..., 11. As shown in Figure 1(c), *Classifier1* gains the lower profit than *Classifier2* due to different profit values in each prediction although it obtains higher accuracy (Acc) than *Classifier2*, where the profit is the total balance of the selling price minus the buying price. This example implies that *the higher accuracy of a DL model may not lead to higher profits*. Thus, it is a necessity to design a profit-based model for DL methods.

To address the above challenges, we propose a Profit-based Deep Architecture with integration of Reinforced Data Selector (namely PDA-RDS) to enhance TFS in stock investment. To address the first challenge, we construct a reinforced data selector (RDS) to select high-quality training samples from source stocks. This data selector is jointly trained with profit-based deep architecture (PDA). To address the second challenge, we establish a PDA, which contains a sequential embedding layer, a historical attention layer, and a predictive layer. Meanwhile, instead of applying a simple cross-entropy loss in the predictive layer, we introduce a dynamic profit weight and a focal loss. Moreover, we devise a training-aware immediate reward function to mitigate the overfitting problem in the reinforcement learning. In this way, the proposed data selector method not only can select useful training samples for the target stock but also can improve the generalization of PDA.

The main contributions are highlighted as follows.

- We are the first to identify two key challenges in applying DL methods to enhance TFS in stock investment, including the challenge of designing an effective data selector and the challenge of designing the profit-based DL model.
- We propose a profit-based deep neural network with integration of a dynamic profit weight and a focal loss function to address the challenge of the profit-based DL model.

- To address the challenge of designing an effective data selector, we devise a reinforced data selector, which integrates a training-aware immediate reward function into the actor-critic reinforcement learning.
- Extensive experiments on both U.S. and China stock-market datasets demonstrate that our framework achieves the superior performance than the state-of-the-art baseline methods.

The rest of the article is organized as follows. Section 2 presents a brief literature survey on related work. Section 3 gives preliminaries on the task definitions and evaluation metrics. We then elaborate the main approach in Section 4. Section 5 next presents experimental results. We finally conclude the paper in Section 6.

2 Related work

We present an overview on related work to our study as follows.

2.1 Transfer learning

The most relevant studies to this paper belong to supervised transfer learning (STL), which is different from the partial transfer learning (PTL). PTL [18, 19] approaches are designed for the scenario, in which the number of target-domain classes are smaller than that in the source domain. It is a critical issue to identify and partially transfer the useful information from the labeled source-domain instances. Different from STL, the labels in target-domain and source-domain are the same in this paper. STL methods can be divided into two categories: feature-based STL methods and instance-based STL methods. Feature-based STL methods such as [20, 21] attempt to project both the source training data and target training data into a common feature space. However, this process may transfer negative training samples from the source domain to the target domain, thereby degrading the predictive performance. Instance-based STL methods [22] avoid transferring negative training samples through selecting or re-weighting the source training samples. However, this process is extremely time-consuming since the sub-model in the transfer learning (TL) framework may execute multiple times to find a valid weight vector. Although reinforcement learning (RL) used in TL has been evidenced effectively in studies [23, 24], it can easily get into overfitting, consequently suffering from the weak generalization, especially when the target validation data is limited. In this paper, we use a training-aware immediate reward based on reinforced transfer learning (RTL) to select useful training samples. Thus, the proposed method not only can be beneficial to the predictive target task but also improve the generalization of the predictive model.

2.2 Reinforcement learning

Another group of studies related to this paper are RL methods, which can be divided into two categories: value-based RL approaches and policy-based RL approaches. The value-based RL approaches [25–28] are based on the value function that is usually constructed by a deep neural network and trained through maximizing the total discounted reward. However, the value-based RL methods have the difficulty to handle the complex environment (e.g., continuous action space). Instead of learning a value indirectly, the policy-based RL

approaches [29] can learn the policy directly, thereby learning various complex strategies and achieving better performance than the value-based RL methods. However, policy-based RL methods often have a high variance caused by the Monte Carlo process. Actor-critic [30] RL approaches integrate the advantages of value-based RL approaches and policy-based RL approaches, consequently reducing the variance. In this paper, our RDS belongs to the category of actor-critic RL approaches. Other advanced actor-critic RL approaches include deep deterministic policy gradient [31], asynchronous advantage actor-critic [32], and proximal policy optimization [33]. Data selection based on RL has been applied in many fields such as active learning [34] , co-training [35], and text-matching [36]. To the best of our knowledge, this is the first work applying reinforced data selector to enhance the trend-following strategy in stock market investment.

3 Preliminaries

In this section, we first give the task definition of enhancing TFS and then introduce evaluation metrics.

3.1 Task definition

TFS[1] [2, 37] is one of the more popular investment strategies in the hedge fund industry. TFS considers both long positions in assets that have historically positive returns, and short positions in assets that have historically negative returns. Following either of them can earn or lose money in the market [38]. Our task is to judge whether to execute the signal generated by TFS. To the best of our knowledge, our TFS is first proposed in this paper and aims to execute the signals that lead to profits rather than those signals that lead to losses. More details are described as follows.

First , we give a general description of TFS as follows. For a stock i, p_t represents the close price at time t. Then, the upper-bound function and the lower-bound function of the close price p within a time range $[t-n, t]$ are defined as $u(p_{t-n,t})$, $l(p_{t-n,t})$, respectively. In TFS, if $p_t \geq u(p_{t-n,t})$ (i.e., no less than the upper bound within a time range $[t-n, t]$), stock i should be bought or long positioned (named "*long*" in short). Otherwise, if $p_t \leq l(p_{t-n,t})$ (i.e., no greater than the lower bound), stock i should be sold or short positioned (named "*short*" in short). Note that both upper bound and lower bound u and l are the functions of the price.

Second, the task of enhancing TFS is to learn a prediction function denoted by $f(X;\theta)$ with parameters θ, which maps the sequential features X to the label space $\hat{y}$, i.e., $\hat{y} = f(X;\theta)$. In other words, the function aims to predict whether to execute the *trading signals* generated by TFS. For instance, TFS generates a signal S at time t, the sequential features X is composed of the technical indicators (i.e., some analysis tools to help investors better understand price movement) computed through $p_{t-n,t}$. We learn the prediction function f by fitting their ground-truth labels $y \in \{-1, 1\}$, where $y = 1$ represents S is the positive signal ($R > 0$ from S to next signal), implying to execute S and the corresponding (X, y) is a positive sample; $y = -1$ represents S is the negative signal ($R \leq 0$), implying not to execute S and the corresponding (X, y) is a negative sample.

[1] https://www.investopedia.com/terms/t/trendtrading.asp

Moreover, we call a transaction as the *executed transaction* when the predictive value $\hat{y} = 1$. Therefore, the main goal of this paper is to enhance TFS rather than creating new TFS.

In summary, a task of enhancing TFS has the input and output: 1) **Input:** sequential features X; 2) **Output:** classification result $\hat{y}$.

3.2 Evaluation metrics

We adopt Cumulative Percentage Rate (CPR), which is the most standard evaluation metric to evaluate investment strategies [39]. CPR is defined as $\mathsf{CPR} = \sum_{i=1}^{\mathcal{N}} R_i$, where R_i can be obtained by (1) and $\mathcal{N}$ is the number of executed transactions. In addition, we also consider a traditional measure, Sharp Ratio (SR) [40] to evaluate the effectiveness of a strategy. SR is formulated as follows:

$$\mathsf{SR} = \frac{\mathcal{A} - \sigma}{\mathcal{V}}, \tag{2}$$

where $\mathcal{A}$ is the average profit rate of all executed transactions, σ is a risk-free profit rate (e.g., a bank's profit rate), and $\mathcal{V}$ is the volatility to measure the risk of a strategy.

The average profit rate of all executed transactions $\mathcal{A}$ can be calculated as follows,

$$\mathcal{A} = \sum_{i=1}^{\mathcal{N}} \left(\frac{1}{\mathcal{T}_i} \sum_{t=1}^{\mathcal{T}_i} K_t - \mathsf{TC} \right), \tag{3}$$

where $K_t = d \times (p_t / p_{t-1} - 1)$, $d = 1$ represents "*long*" holding and $d = -1$ represents "*short*" holding of transaction i within $\mathcal{T}_i$ holding periods.

The volatility $\mathcal{V}$ is given by

$$\mathcal{V} = \sqrt{\frac{\sum_{i=1}^{\mathcal{N}} \sum_{t=1}^{\mathcal{T}_i} (K_t - \bar{K}_t)^2}{\sum_{i=1}^{\mathcal{N}} \mathcal{T}_i}}, \tag{4}$$

where $\bar{K}_t = \frac{\sum_{i=1}^{\mathcal{N}} \sum_{t=1}^{\mathcal{T}_i} K_t}{\sum_{i=1}^{\mathcal{N}} \mathcal{T}_i}$.

On the one hand, SR is determined by $\mathcal{A}$ and $\mathcal{V}$. The larger $\mathcal{A}$ or smaller $\mathcal{V}$ may cause the increment of SR. On the other hand, CPR represents how much profit rate that a method can make for all executed transactions. The higher CPR implies the better performance of an investment strategy. It is worth mentioning that a larger CPR also implies a larger SR while a larger SR may not lead to a larger CPR. This is because a larger CPR means a TFS not executing some signals whose CPR is less than zero, consequently increasing the $\mathcal{A}$ and decreasing $\mathcal{V}$. As a result, SR is increased. However, a larger SR may not lead to a larger CPR since no execution of some signals whose CPR is greater than zero may also decrease the volatility $\mathcal{V}$, thereby increasing SR while CPR is not increased.

Meanwhile, we also introduce Average Percentage Rate (APR), which is defined as $\mathsf{APR} = \frac{\mathsf{CPR}}{\mathcal{N}}$. APR represents how much profit rate a method can make for each executed transaction. Both CPR and APR are main metrics in our paper. The higher APR means that each executed transaction can be tolerant to the uncertain TC which may fluctuate due to the uncertainty of financial markets. Thus, APR can better measure the performance of two investment strategies when both of them have a close CPR value [39].

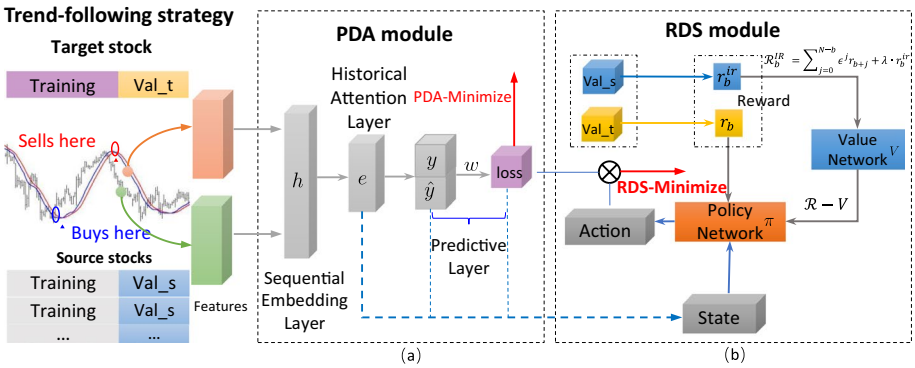


Fig. 2 PDA-RDS framework

4 Our approach

4.1 PDA module

Figure 2 depicts the PDA-RDS framework, which consists of (a) PDA module and (b) RDS module, to be illustrated as follows. As shown in Figure 2(a), PDA consists of three layers: a sequential embedding layer, a historical attention layer, and a predictive layer.

4.1.1 Sequential embedding layer

Long Short-Term Memory (LSTM) [41] has been widely used to analyze time-series data, such as natural language processing [5], video processing [42], and stock price prediction [43]. In PDA, at each time-step, LSTM learns the hidden representation h_t by jointly considering the input $x_t \in \mathbb{R}^D$ with dimension D and the hidden representation h_{t-1} (at the previous time $t-1$) to capture long sequential dependency and temporal patterns. Formally, an LSTM layer is applied to map $X = [x_1, \cdots, x_T] \in \mathbb{R}^{D \times T}$ into hidden representations $H = [h_1, \cdots, h_T] \in \mathbb{R}^{U \times T}$ with the new dimension U.

4.1.2 Historical attention layer

The attention mechanism [44] has been widely used in LSTM-based solutions for sequential-learning problems [11]. The attention scheme aims to model the effect that the data at different time-steps can contribute differently to the representation of the whole time-series data. Specifically, the historical attention [45] is described as $z = \sum_{i=1}^{T} \gamma_i h_i$, where γ_i is computed by an attention function defined as follows

$$\gamma_i = \frac{\exp(g_i)}{\sum_{j=1}^{T} \exp(g_j)}, \tag{5}$$

$$g = v^{\top} \tanh(W_1 h_t + b), \tag{6}$$

where W_1, b, $v^{\top}$ are parameters to be learned, and z is obtained through all previous hidden representations. Instead of directly applying z, we construct a final hidden representation

e, which contains an attention representation z and the last hidden representation h_T of LSTM. We have $e = [z^\top, h_T^\top]^\top$.

4.1.3 Predictive layer

As shown in Figure 1, there is an unbalance between the negative profit values and positive profit values. The solutions to the unbalanced classification problem can be divided into i) sampling-based approaches and ii) cost-sensitive approaches. Sampling-based approaches [46] address the imbalance of the input data by adjusting the minority class or the majority class in the input dataset so as to construct the balanced training dataset. Cost-sensitive approaches [47] penalize the mis-classification in the training phase via the fixed-cost matrix. However, cost-sensitive approaches do not consider the profit difference of each prediction. To solve this problem, we devise a dynamic profit weight loss function, which is defined as follows:

$$\mathcal{L}_1 = -\sum_{k=1}^{2} |w_k| \cdot 1_{y_k}^\top \cdot \log[\hat{y}_k], \tag{7}$$

where $\hat{y} = \mathsf{softmax}(e)$ and y is a one-hot 2-dimensional vector. One dimension denotes this vector corresponding to a positive sample and another dimension denotes this vector corresponding to a negative sample. Commonly, w_k is set to $\frac{q}{Q_k}$ [12], where Q_k is the number of samples that are labelled as an integer k and q is a constant parameter. In the proposed PDA, w_k can be dynamically adjusted for each training sample i. The weight for each sample i is denoted by w_k^i, which can be calculated as follows:

$$w_k^i = \begin{cases} \frac{\|R_i\|}{w_k^s} & if \|R_i\| > w_k^s \\ w_k^s & if \|R_i\| \le w_k^s \end{cases}, \tag{8}$$

where R_i is the profit rate as defined in (1). We use a fixed value w_k^s to smooth R_i to normalize the profits of different stocks with highly-varied prices, which are hard to measure together.

Furthermore, we introduce a focal loss (FL) [48] to learn the positive samples with profits less than w_k^s, which are difficult to be identified. The focal loss is defined as $FL = (1-\hat{y})^\gamma \log[\hat{y}]$, where γ is a tunable focusing-parameter. Since the mis-classification of negative samples does not have significant impacts as the positive samples and negative samples contain substantial noises, we only introduce a parameter c to the negative samples instead of the focal loss [49]. Finally, the profit-based loss function is formulated as follows:

$$\mathcal{L}_2 = -|w_1^i| \cdot 1_{y_1}^\top \cdot (1-\hat{y}_1)^\gamma \cdot \log[\hat{y}_1] - |w_2^i| \cdot \frac{1}{c} \cdot 1_{y_2}^\top \cdot \log[\hat{y}_2]. \tag{9}$$

4.2 RDS module

As is shown in Figure 1, applying TFS to the original price sequence will reduce the number of training samples since each transaction contains a time range of the original price sequence. Thus, it is very important to introduce other stocks into training samples. On the one hand, it can increase the number of training samples. On the other hand, it can

improve the generality of the model. However, not all the training samples are beneficial to the target stock. Therefore, we propose a reinforced data selector (RDS) to select high quality training samples as illustrated in Figure 2(b). In particular, we consider the process of data selection as a Markov decision process, which consists of *state*, *action*, *reward*, and *optimization* described as follows.

4.2.1 State

A set S_b of states is obtained from the environment (i.e., PDA) where b is the batch number. Each state $S_i \in S_b$ consists of three features. The first feature is a high-level signal representation learned by the PDA module while [36] has proved that the high-level (hidden) signal representation can achieve relatively excellent performance. The second feature is the predicted probability of the PDA module on training sample $X_i \in X_b$. The third feature is the training loss of the PDA module on training sample X_i. A combination of the second and the third features can measure the performance of the current PDA module on X_i.

4.2.2 Action

An action is denoted by $a_i \in A_b$, where A_b is the action set containing m actions $(i = 1, 2, ..., m)$. Each action a_i is essentially a probability of a training sample X_i being selected, thereby $a_i \in [0, 1]$. Instead of applying the selection in X_b, we conduct a selection step in the calculation of the batch loss, which is the weighted average of all the loss values of all the actions. For instance, when action a_i is chosen, its loss value is included in the batch loss; otherwise its loss value will not be included. The PDA module is then updated by minimizing the batch loss.

4.2.3 Reward

After updating the PDA module, we can obtain a *delay reward* r_b on the target validation samples X^{φ}. CPR and APR are two important metrics to measure the strategy described in Section 3.2. Here, we define a conjoint reward CR to combine the CPR and APR, which is shown as follows:

$$\mathsf{CR} = \mathsf{CPR} + \varphi \times \mathsf{APR}, \tag{10}$$

where φ is the parameter to balance the weight of CPR and APR. The value of φ is choosen from zero to the number of the validating samples in our experiments. Formally, r_b is defined as follows:

$$r_b = \frac{1}{C} \sum_{i=1}^{C} (\mathsf{CR}_i^+ - \mathsf{CR}_i^-), \tag{11}$$

where C is the number of X^{φ}, CR_i^- and CR_i^+ represents the conjoint rewards before updating the PDA module and after updating the PDA module, respectively. Thus, the future total reward r_b' for each batch in an episode can be formalized as $r_b' = \sum_{j=0}^{N-b} \epsilon^j r_{b+j}$, where N is the number of batches in an episode and ϵ is the reward discount factor.

The delay reward is only calculated from the target validation samples with the relatively small volume. the maximization of the total delay reward may lead to the overfitting of PDA module on the target validation samples, consequently suffering from weak

generalization and poor performance. To address this issue, we introduce three methods to combine the performance in the validation samples from all the source stocks.

(1) *Uniform integration* (UI). UI calculates the delay reward of validation samples from all the source stocks as reward r^{ui}. Thus, the total reward is calculated as $\mathcal{R}_b^{UI} = \sum_{j=0}^{N-b} e^j(r_{b+j} + \lambda \cdot r_{b+j}^{ui})$.

(2) *Weighted integration* (WI). UI may introduce the negative validation samples since not all source stocks can equally contribute to the target stock. Thus, we apply WI method, which is to select a subset from source stocks based on the historical cosine similarity. The term r^{wi} is the reward on the validation samples from the subset of source stocks, which contains ten stocks. The total reward is calculated by $\mathcal{R}_b^{WI} = \sum_{j=0}^{N-b} e^j(r_{b+j} + \lambda \cdot r_{b+j}^{wi})$.

(3) *Immediate reward* (IR). However, the aforementioned two methods are not training aware methods. We propose an immediate reward, IR, which is obtained on the validation samples of a specific stock, whose training samples are used to update the PDA module at the current training time. It's reasonable that more training samples are selected to update the PDA module from a specific stock, its validation samples X^v should have a larger IR. Therefore, IR is given as follows:

$$r_b^{ir} = \frac{\tilde{E}}{E}\frac{1}{G}\sum_{i=1}^{G}(\mathsf{CR}_i^+ - \mathsf{CR}_i^-), \tag{12}$$

where E represents the number of training samples, $\tilde{E}$ represents the number of selected training samples, and G is the number of X^v. Finally, the total reward of each batch is formulated as follows:

$$\mathcal{R}_b^{IR} = r_b' + \lambda \cdot r_b^{ir}, \tag{13}$$

where λ is the parameter used to control the importance of the immediate reward and delay reward. Note that, PDA-RDS uses $\mathcal{R}_b^{IR}$ as the reward function.

4.2.4 Optimization

We then apply the actor-critic algorithm [30] to optimize RDS. The main process of actor-critic algorithm is as follows. First, the policy π (i.e., two fully-connected layers) is paramterized by θ and an action A_b is obtained through $\pi(\cdot\|S_b)$ for a given S_b. After that, PDA is updated by S_b and A_b once receiving the reward R_b, which contains a delay reward r_b and an immediate reward r_b^{ir}. For each episode, θ is updated via $\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_\theta(S_b)\delta$, where α_θ is the learning rate, and δ is the total reward in the whole episode. To reduce the variance and update through steps, we introduce the advantage function network to estimate δ, which is considered to have smaller variance and faster convergence. Therefore, $\delta_b = Q(S_b, A_b) - V_\mu(S_b)$, where the value estimator V_μ is a network consisting of two fully-connected layers parameterized by μ and $Q(S_b, A_b)$ and can be calculated by (13). Finally, θ is updated as follows:

$$\theta \leftarrow \theta + \alpha_\theta \nabla_\theta \log \pi_\theta(S_b)\delta_b. \tag{14}$$

The parameter μ in value function V_μ is updated through minimizing the mean squared error between the future reward and the estimate reward,

$$\mu \leftarrow \mu + \alpha_\mu \nabla_\mu ||\mathcal{R}_b^{IR} - V_\mu(S_b)||_2^2, \tag{15}$$

where α_μ is the learning rate.

4.2.5 Fine-tuning

Fine-tuning [50] is a process to make small modifications of the pre-training model. In particular, we transfer the parameters of PDA module from pre-training process and freeze the parameters in the sequential embedding layer and historical attention layer. In this way, we only tune the parameters in the predictive layer with the target training data. Finally, the detailed learning process is described in Algorithm 1.

Algorithm 1 Learning process of Reinforced data selector

Algorithm 1 Learning process of Reinforced data selector

Input: Episode L, source training (validating) samples X^t, X^v, target training (validating) samples X^τ, X^φ.
Output: The trained PDA module.

Initial PDA module through training with X^τ;
for episode = 1 to L **do**
  for all X_b^t, X_b^v **do**
    Obtain the State S_b^t;
    Sample action A_b^t according to policy $\pi(S_b^t)$;
    Update the profit based model with X_b^t and A_b^t;
    Obtain the delay reward r_b in (11);
    Obtain the immediate reward r_b^{ir} in (12);
    Store $(S_b^t, A_b^t, r_b, r_b^{ir})$ to a history memory H;
  end for
  for all $(S_b^t, A_b^t, r_b, r_b^{ir}) \in H$ **do**
    Calculate the future total reward $\mathcal{R}_b^{\mathrm{IR}}$ in (13);
    Obtain the estimated future total reward $V(S_b^t)$;
    Update the policy network as in (14);
    Update the value network as in (15);
  end for
end for

Freeze the sequential embedding layer and historical attention layer of PDA module
for all $X_b \in X^\tau$ **do**
  Fine-tuning the prediction layer with X_b in (9);
end for

Table 1 Features in vector v_t

Features	Calculation
5-rma, 10-rma, 15-rma	$n\text{-rma} = \frac{n\times\text{price}_t - \sum_{i=1}^{n}\text{price}_i}{\sum_{i=1}^{n}\text{price}_i}$
20-rma, 25-rma, 30-rma	
1-roc, 3-roc, 5-roc	$n\text{-roc} = \frac{\text{price}_t - \text{price}_{t-n}}{\text{price}_{t-n}}$
7-roc, 9-roc, 11-roc	

rma: rate of moving average; roc: rate of change; price: p_t or $p_t - u(p_{t-n,t})$

5 Experiments

5.1 Datasets

We conduct experiments on both U.S. and China stock-market datasets. In particular, we consider S &P 500[2] constituents containing 500 stocks in U.S. stock markets and HS 300[3] constituents containing 300 stocks in China stock markets (Chinese mainland). For S &P 500 dataset, we collect price-volume data (open, close, high, low, and volume) of each stock from 2013/02/08 to 2019/06/17, which spans more than six years and contains 1,174 trading days. For HS 300 dataset, we collect price-volume data (open, close, high, low, and volume) of each stock from 2013/10/08 to 2020/06/08, which spans more than six years and contains 1,628 trading days. In the preprocessing stage, price data samples are augmented through the back-rehabilitation called *price data adjustment*. The adjusted price data can exactly reflect the stock's values accounting for some corporate actions, such as *stock splits* and *stock dividends*.

We then let the upper bound function $u(\cdot)$ and the lower bound function $l(\cdot)$ of TFS be $u(p_{t-n,t}) = l(p_{t-n,t}) = \frac{\sum_{i=t-n}^{t} p_i}{n}$, where p_t represents the close price and n is set to 20. Thus, the new time-series data is calculated by $p_t - u(p_{t-n,t})$. Meanwhile, we apply TFS and construct the feature matrix X and label y of each transaction. $x_t \in X$ represents the feature vector at time t obtained from Table 1, which contains 24 features. Finally, the dataset of each stock is divided into three sets: training set (S &P 500: 2013/02/08-2017/02/08; HS 300: 2013/10/08-2017/10/08), validation set (S &P 500: 2017/03/01-2018/03/01; HS 300: 2017/10/08-2018/10/08), and testing set (S &P 500: 2018/03/01-2019/06/17; HS 300: 2018/10/08-2020/06/08). The transaction cost TC of a complete transaction is set to 0.1% of transaction amount as in [39]. Therefore, the ratio of the number of positive and negative samples of HS 300 and S &P 500 is 39:61 and 29:71, respectively.

5.2 Evaluation metrics

Following previous studies on stock prediction [11, 12], we also adopt two standard metrics: accuracy (Acc) and F1 score (F1) to evaluate the prediction accuracy. In Section 3.2, we also introduce three metrics: CPR, APR, ASR. ASR is the annualized SR, which is

[2] https://drive.google.com/file/d/1Iowk-9946O53Okk6vDPnfgRDCGa5sJng/view?usp=sharing
[3] https://drive.google.com/file/d/1InH3nnNEE2lFbWnrrT_67jZBjiH2Hfb5/view?usp=sharing

computed as $\mathsf{ASR} = \mathsf{SR} \times \sqrt{M}$, where M is the number of holding periods in a year. Meanwhile, we use the average ASR of all stocks in the testing dataset. We next evaluate the performance of our proposed PDA-RDS method with comparison of other state-of-the-art (SOA) baseline methods. The higher CPR, APR, and ASR mean the better performance of an investment strategy.

5.3 Experimental details

All the experiments were conducted on a workstation consisting of 32-Core CPU and two Nvidia GPUs (Tesla P100) with 12 GB RAM (OS is Ubuntu 18.04 LTS). We implemented the framework on top of TensorFlow 2.0 with Python 3.7 and adopt the Adam optimizer [51]. The parameters of the Adam optimizer are fixed to a range from 0.9 to 0.999. Besides, we conduct a grid search over all hyperparameters for each method and dataset. Regarding the actor-critic method, we adopt two fully-connected networks for the policy network and the value network, respectively. We vary the number of hidden state size in $\{16, 32, 64, 128\}$ and choose λ from [0, 20]. For PDA module, the number of LSTM hidden units is chosen from set $\{16, 32, 64, 128\}$. Following [48], we tune the focusing parameter γ via grid-search of [0, 5]. Moreover, we tune parameter c, and w_k^s from $\{1, 3, 5, 10\}$ and $\{0, 0.01, 0.015, 0.02\}$. The dropout rate of our model is chosen from $\{0.2, 0.3, 0.5\}$. During the training phase, the batch size is chosen from $\{32, 64, 128\}$ and the learning rate is set from $\{0.005, 0.01, 0.05\}$. We test different hyperparameters and find the best settings for each method according to the performance on the validation dataset. Table 3 lists configurations of the main parameters in our model.

5.4 Baselines

We evaluate and compare the proposed PDA-RDS with the following baseline methods and their variants:

- *TFS* is a naive baseline method considering the moving average TFS.
- *LSTM* has an LSTM layer and a prediction layer.
- *Gated recurrent units (GRU)* has an GRU layer and a prediction layer.
- *ALSTM* is an attentive LSTM [52], which is the same as our PDA except for the loss function. ALSTM only adopts the simple cross-entropy loss function.
- *RAND* is a randomized method that randomly selects the training data from 100 source stocks to train them together for the target stock set.
- *Full Transfer Learning (TL)* is a TL method, which transfers the training data of source stocks without data selection (only PDA module is used). The PDA module is then fine-tuned by the target training data.
- *Similarity TL* [15] transfers the training data of source stocks according to the similarity to the target stock set. Particularly, the cosine similarity between the close price of the source stock and that of the target stock is calculated. Meanwhile, we select the top-100 stocks for similarity TL.
- *Ruder and Plank* [22] proposes an instance selection method with Bayesian optimization.
- *Reinforced Transfer Learning (RTL)* [36] proposes an instance selection method with reinforcement learning for transfer learning. RTL has the designated reward while our RDS has different rewards (as follows).

Table 2 Results of PDA-RDS with other baselines

Models \ Datasets	S &P 500					HS 300				
	Acc	F1	CPR	APR	ASR	Acc	F1	CPR	APR	ASR
TFS	/	/	15.05	0.0013	0.5228	/	/	18.12	0.0032	0.3348
LSTM	0.7461	0.4367	2.32	0.0044	0.3016	0.6932	0.3618	4.33	0.0044	0.2644
GRU	0.7353	0.4351	2.12	0.0043	0.3244	0.6872	0.3815	5.14	0.0046	0.2863
ALSTM	0.7472	0.4369	6.98	0.0054	0.5316	0.6674	0.3988	5.76	0.0048	0.2843
PDA (ours)	0.6632	0.5332	18.08	0.0065	0.7400	0.6321	0.4432	20.48	0.0057	0.5276
RAND	0.6692	0.5367	18.46	0.0068	0.7761	0.6647	0.4266	20.14	0.0055	0.5081
Full TL	0.6710	0.5388	18.66	0.0069	0.8002	0.6288	0.4654	20.88	0.0060	0.5443
Similarity TL [15]	0.6820	0.5460	19.55	0.0072	0.8455	0.6344	0.4699	20.96	0.0062	0.5686
Ruder and Plank [22]	0.6457	0.5206	16.32	0.0069	0.6300	0.5264	0.4322	19.43	0.0042	0.4442
RTL [36]	0.6763	0.5430	20.44	0.0070	0.9100	0.5944	0.4653	21.56	0.0066	0.6058
PDA − RDS (ours)	0.6932	0.5844	23.10	0.0080	1.0131	0.6122	0.4876	23.96	0.0075	0.6641

Table 3 Parameters in PDA-RDS

Parameters	Configurations
tensorflow random seed	123456
ALSTM	lstm(64)-lstm(64)-attention(64)-dense(2)
actor-critic	dense(64)-dense(64)
gamma in Adam optimize	0.95
dropout of lstm layer	0.5
smooth value w_k^s in Eq. (8)	0.01
γ in (9)	2
c in (9)	2
λ in (13)	7
train batchsize	64
train learning rate	0.01 for actor critic and 0.015 for pda model
training episodes	150

Note that methods TFS, LSTM, GRU, ALSTM, and PDA incorporate the training samples (also validating samples and testing samples) of all stocks into one training set (also validating set and testing set). Other data-selection methods divide all stocks into source stocks and target stocks. Target stocks essentially consists of a stock set with 10 stocks in order to avoid the insufficiency of training and validating samples (if a target stock only contains a stock). There are 50 sets on S &P 500 dataset and 30 sets on HS 300 dataset, each of which contains 10 stocks selected from the training samples sorted according to the CPR values. In order to achieve a fair comparison, the hidden layers and other parameters in LSTM, GRU, ALSTM and PDA models are the same as those in our model as given in Table 3. Moreover, parameters in Full TL, Similarity TL, and RTL are also the same as those in our PDA-RDS. Furthermore, we follow [22] to tune the parameters in Ruder and Plank.

5.5 Results

Table 2 compares our PDA-RDS with other baselines. We have the following observations from Table 2: i) PDA-RDS outperforms all the baselines in terms of CPR and APR (i.e., making more profits). Particularly, CPR, APR and ASR of PDA-RDS are 13.0%, 14.29%, 11.32% higher than those of RTL [36], respectively. ii) Among all the conventional baselines, data-selection schemes such as RAND, Full TL, Similarity TL, Ruder and Plank, and RTL outperform TFS, LSTM, GRU, and ALSTM, implying that not all source stocks contribute to the performance improvement. Although Full TL also transfers all the training data of source stocks, it performs slightly better than RAND due the fine-tuning process. RTL performs much better than other data-selection methods (except for our PDA-RDS) due to the introduction of RL in data selection. Our PDA-RDS has superior performance than other baselines mainly due to the profit-based model and the data-selection based on IR. iii) ALSTM has a higher ASR than TFS in S &P 500 dataset but the corresponding CPR is much lower than TFS. It implies that ASR may not be a good metrics in evaluating the enhanced TFS model. iv) The models incorporating the profit-based loss function (PDA, RAND, Full TL, Similarity TL, Ruder and Plank, RTL, PDA-RDS) show a better F1 score than the model without profit-based loss function (LSTM, FRU, ALSTM), which demonstrates that the proposed loss function can handle the label unbalance problem and produce a larger CPR.

5.6 Ablation studies

5.6.1 Effectiveness of PDA module

We compare PDA with ALSTM-based methods as well as variants of our PDA on S &P 500 dataset.

- *ALSTM w/ us* is a variant of ALSTM and is trained by data samples which are preprocessed via under-sampling method.
- *ALSTM w/ os* is a variant of ALSTM and is trained by data samples which are preprocessed by over-sampling method.
- *ALSTM w/ w* is a variant of ALSTM which consists of a fixed positive weight w to the cross-entropy loss function, where w is the ratio of negative samples to positive samples.
- *PDA w/o FL* is a variant of the PDA module variant without the FL loss.
- *PDA w/ FL* is a variant of the PDA module variant with the FL loss for both positive and negative samples.
- *PDA w/o SM* is a variant of the PDA module variant without the smoothing process as defined in (8).

Table 4 presents performance comparison of our PDA with other methods. We have the following observations from the experimental results: i) Regarding the unbalanced data problem, our PDA obtains the largest F1 score, which demonstrates the effectiveness of the profit-based loss function. In addition, the under-sampling method (ALSTM w /us) performs better than ALSTM w/ os and ALSTM w/ w. Moreover, ALSTM w/ w

Table 4 Performance comparison of PDA with ALSTM-based methods as well as variants of PDA

	Acc	F1	CPR	APR
ALSTM w/ us	0.5081	0.4768	17.02	0.0033
ALSTM w/ os	0.2715	0.2350	13.42	0.0012
ALSTM w/ *w*	0.7466	0.4376	0.0584	0.0047
PDA (ours)	0.6632	0.5332	18.08	0.0065
PDA w/o FL	0.5541	0.5098	17.11	0.0035
PDA w/ FL	0.5928	0.5234	18.19	0.0047
PDA w/o SM	0.5347	0.4876	13.19	0.0027

Table 5 Performance comparison of PDA-RDS with different reward schemes

	Acc	F1	CPR	APR
PDA-RDS-UI	0.6684	0.5390	19.28	0.0066
PDA-RDS-WI	0.6844	0.5531	21.08	0.0073
PDA-RDS	0.6932	0.5844	23.10	0.0080

mainly focuses on prediction instead of the profit-making strategy, thereby the CPR value of ALSTM w/ *w* is the lowest (i.e., 0.0584). ii) Compared with the ablation variants of our PDA, such as PDA w/o FL, PDA w/ FL, and PDA w/o SM, PDA can learn more information when the focal loss is introduced to the positive training samples. For example, PDA achieves the best APR 0.0065, which is 28% improvement over PDA w/FL though PDA w/ FL has slightly higher CPR than PDA (i.e., 0.6% higher) while APR plays a more important role in profit-making when the CPR is close to same (detailed in Section 5.7.4). Furthermore, the smooth function plays an important role in our PDA. In summary, PDA demonstrates its advantages in making profits.

5.6.2 Effectiveness of IR in RDS module

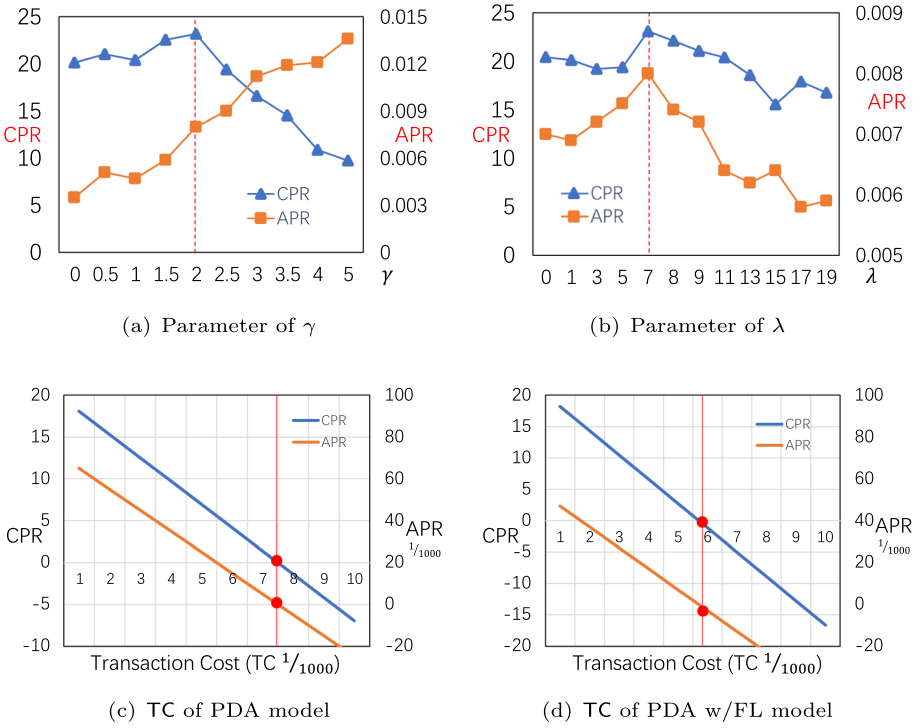
To evaluate the effectiveness of IR in RDS module, we compare variants of different reward schemes on S &P 500 dataset. Table 5 presents the results. PDA-RDS has two variants: PDA-RDS-UI and PDA-RDS-WI, which also perform excellently (aligning with Table 2). They all improve the generalization of reinforced data selector avoiding overfitting the validation dataset. We observe from Table 5 that PDA-RDS outperforms PDA-RDS-UI and PDA-RDS-WI due to the introduction of IR to the training process, thereby improving the learning effect.

5.7 Parameters analysis

The analysis of all parameters is based on S &P 500 dataset.

5.7.1 Impact of *n*

In our experimental setting, all stocks share the same value n (i.e., $n = 20$ in this paper). However, it is reasonable to choose different values of n for each stock. Therefore, we



(a) Parameter of γ (b) Parameter of λ

(c) TC of PDA model (d) TC of PDA w/FL model

Fig. 3 The impact of parameters

design a strategy, named the best-n TFS, in which n can be obtained through calculating the best CPR value from the historical training data of each stock of S &P 500. The best-n TFS has obtained CPR $= 5.90$ and APR $= 0.0003$, which are much lower than those of TFS (fixed $n = 20$) in Table 2. The main reason lies in the possible overfitting of different values of n of each stock in the best-n TFS in the historical training data. Meanwhile, we also apply PDA to enhance the best-n TFS and achieve CPR $= 10.39$ and APR $= 0.0028$, outperforming the best-n TFS; this result implies that the introduction of PDA can greatly improve the best-n TFS.

5.7.2 Impact of γ

The focusing-parameter γ defined in (9) is tunable. We investigate the impact of γ on the performance (CPR and APR) of our PDA-RDS via varying γ within range [0, 5]. Figure 3(a) illustrates the results. We observe from Figure 3(a) that CPR fluctuates not significantly when γ falls into range [0, 2] while it decreases rapidly when γ falls into range [2, 5]. Differently, APR swells significantly when γ falls into range [0, 5]. The main reason lies in the fact that PDA learns more information from positive samples with the increment of γ in range [0, 2]. However, the excessive attention to positive samples decreases the number of positive samples in prediction, especially when $\gamma > 2$. The fewer positive samples in prediction consequently lead to the decreased CPR value and the increased APR value.

Table 6 Comparison of different RL algorithms

Methods	Acc	F1	CPR	APR
(only)	0.6632	0.5332	18.08	0.0065
PG	0.6344	0.5417	18.68	0.0066
DQN	0.6482	0.5467	19.46	0.0068
DDPG	0.7110	0.5476	24.86	0.0079
PPO	0.6810	0.5460	21.43	0.0085
AC	0.6932	0.5844	23.10	0.0080

5.7.3 Impact of λ

We investigate the impact of λ on CPR and APR of our PDA-RDS as shown in Figure 3(b). We vary $\boldsymbol{\lambda}$ with [0, 20] since the immediate reward (IR) is calculated by one source stock and the delay reward is calculated by ten target stocks. We observe from Figure 3(b) that both CPR and APR achieve the best values when $\lambda = 7$. The further increment of λ does not lead to a better performance when $\lambda > 7$ because IR contributes more to the total reward while failing to consider the delay reward. Moreover, a smaller IR (i.e., $\lambda < 7$) also leads to a worse performance since smaller IR contributes as a noise to the delay reward.

5.7.4 Impact of TC

We vary different values of TC to explain the importance between CPR and APR with PDA and PDA w/FL because they have approximate CPR but different APR. As shown in Figure 3(c) and (d), the horizontal axis represents the transaction cost (TC), which gradually increases from 0.001 to 0.01 while the vertical-axis represents the varied CPR and APR. It can be observed that PDA w/ FL loses more than PDA and turns into negative CPR quickly with the increment of transaction costs. More specifically, PDA takes a long time to turn into negative CPR as the red line of PDA goes obviously behind the red line of PDA w/ FL (i.e., $\mathsf{TC} = 0.0075$ vs $\mathsf{TC} = 0.0057$). Therefore, in the case of similar to CPR, the higher APR leads to the stronger the model's ability to tolerate uncertainty in the future trading. The uncertain may be caused by the uncertain transaction slippage.

5.7.5 Impacts of RL algorithms

We use other RL algorithms, such as Policy Gradient (PG) [29], Proximal Policy Optimization (PPO) [33], Deep Q-Network (DQN) [25, 53], and Deep Deterministic Policy Gradient (DDPG) [31] to substitute the actor-critic (AC) scheme in the RDS module to evaluate the impacts of different RL algorithms in our framework. The results are shown in Table 6.

It can be observed from Table 6 that all RL algorithms used in the RDS framework are better than the pure PDA. More specifically, CPR of DDPG achieves slightly higher CPR than AC while PPO achieves slightly higher APR than AC though AC achieves relatively more balanced results (i.e., relatively high values in both CPR and APR). In addition, AC outperforms DQN and PG methods.

6 Conclusion

In this paper, we propose to apply deep learning methods to enhance traditional trading strategies, that is trend-following strategy. However, the enhancing task creates two challenges. To address these two challenges, we propose a profit-based deep architecture with integration of reinforced data selector (PDA-RDS) to enhance trend-following strategies in stock investment. PDA consists of a composite deep network, a dynamic profit weight and a focal loss function, thereby well quantifying profits during the training process. RDS integrates a training-aware immediate reward function into the actor-critic reinforcement learning model to select relevant stock data so as to improve the learning effectiveness of the entire structure. Extensive experiments on realistic stock data demonstrates the superior performance of our proposed model than the state-of-the-art methods.

Funding The research is supported by the Key-Area Research and Development Program of Guangdong Province (2020B010165003), the National Natural Science Foundation of China under project (62032025), and the Technology Program of Guangzhou, China (202103050004), Faculty Research Grants (DB22A5 and DB22B7) of Lingnan University, Hong Kong.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Brock, W., Lakonishok, J., LeBaron, B.: Simple technical trading rules and the stochastic properties of stock returns. J. Financ. **47**(5), 1731–1764 (1992)
2. James, J., et al.: Simple trend-following strategies in currency trading. Quantitative Finance **3**(4), 75–77 (2003)
3. Jegadeesh, N., Titman, S.: Returns to buying winners and selling losers: Implications for stock market efficiency. J. Financ. **48**(1), 65–91 (1993)
4. Fong, S., Si, Y.-W., Tai, J.: Trend following algorithms in automated derivatives market trading. Expert Syst. Appl. **39**(13), 11378–11390 (2012)
5. Zheng, W., Zheng, Z., Wan, H., Chen, C.: Dynamically route hierarchical structure representation to attentive capsule for text classification. In: IJCAI'19, pp. 5464–5470. AAAI Press, (2019)
6. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS'12, vol. 60, pp. 1097–1105. MIT Press, (2012)
7. Feng, F., He, X., Wang, X., Luo, C., Liu, Y., Chua, T.-S.: Temporal relational ranking for stock prediction. ACM Transactions on Information Systems (TOIS) **37**(2), 1–30 (2019)
8. Wang, H., Wu, Y., Min, G., Miao, W.: A graph neural network-based digital twin for network slicing management. IEEE Trans. Industr. Inf. **18**(2), 1367–1376 (2020)
9. Wu, Y., Wang, Z., Ma, Y., Leung, V.C.: Deep reinforcement learning for blockchain in industrial iot: A survey. Comput. Netw. **191**,(2021)
10. Liang, W., Xie, S., Cai, J., Xu, J., Hu, Y., Xu, Y., Qiu, M.: Deep neural network security collaborative filtering scheme for service recommendation in intelligent cyber-physical systems. IEEE Internet of Things Journal, 1–1 (2021)
11. Feng, F., Chen, H., He, X., Ding, J., Chua, T.-S.: Enhancing stock movement prediction with adversarial training. In: IJCAI'19, pp. 5843–5849. AAAI Press, (2019)
12. Tran, D.T., Iosifidis, A., Kanniainen, J., Gabbouj, M.: Temporal attention-augmented bilinear network for financial time-series data analysis. IEEE Transactions on Neural Networks and Learning Systems **30**(5), 1407–1418 (2018)

13. Hu, Z., Liu, W., Bian, J., Liu, X., Liu, T.-Y.: Listening to chaotic whispers: A deep learning framework for news-oriented stock trend prediction. In: ICDM'18, pp. 261–269. Association for Computing Machinery, (2018)
14. Bollen, J., Mao, H., Zeng, X.: Twitter mood predicts the stock market. Journal of Computational Science **2**(1), 1–8 (2011)
15. He, Q.-Q., Pang, P.C.-I., Si, Y.-W.: Transfer learning for financial time series forecasting. In: PRIJCAI'19, vol. 11671, pp. 24–36 (2019). Springer
16. Nguyen, T.-T., Yoon, S.: A novel approach to short-term stock price movement prediction using transfer learning. Appl. Sci. **9**(22), 4745 (2019)
17. Fischer, T., Krauss, C.: Deep learning with long short-term memory networks for financial market predictions. Eur. J. Oper. Res. **270**(2), 654–669 (2018)
18. Cao, Z., Long, M., Wang, J., Jordan, M.I.: Partial transfer learning with selective adversarial networks. In: CVPR'18, pp. 2724–2732 (2018)
19. Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., He, Q.: A comprehensive survey on transfer learning. Proc. IEEE **109**(1), 43–76 (2020)
20. Wang, C., Mahadevan, S.: Heterogeneous domain adaptation using manifold alignment. In: IJCAI'11, pp. 1541–1546 (2011)
21. Xie, S., Zheng, Z., Chen, L., Chen, C.: Learning semantic representations for unsupervised domain adaptation. In: ICML'18, vol. 80, pp. 5423–5432. Cambridge MA: JMLR, (2018)
22. Ruder, S., Plank, B.: Learning to select data for transfer learning with Bayesian optimization. In: EMNLP'17, pp. 372–382. Association for Computational Linguistics (2017)
23. Ye, R., Dai, Q.: A novel transfer learning framework for time series forecasting. Knowl.-Based Syst. **156**, 74–99 (2018)
24. Wang, B., Qiu, M., Wang, X., Li, Y., Gong, Y., Zeng, X., Huang, J., Zheng, B., Cai, D., Zhou, J.: A minimax game for instance based selective transfer learning. In: KDD'19, pp. 34–43. Association for Computing Machinery (2019)
25. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
26. VanHasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI'16, pp. 2094–2100 (2016)
27. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., De Freitas, N.: Dueling network architectures for deep reinforcement learning. In: ICML'16, pp. 1995–2003. JMLR.org (2016)
28. Yan, Z., Ge, J., Wu, Y., Li, L., Li, T.: Automatic virtual network embedding: A deep reinforcement learning approach with graph convolutional networks. IEEE J. Sel. Areas Commun. **38**(6), 1040–1057 (2020)
29. Sutton, R.S., McAllester, D.A., Singh, S.P., Mansour, Y.: Policy gradient methods for reinforcement learning with function approximation. In: NIPS'00, pp. 1057–1063. MIT Press (2000)
30. Konda, V.R., Tsitsiklis, J.N.: Actor-critic algorithms. In: NIPS'00, pp. 1008–1014. MIT Press (2000)
31. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv:1509.02971 (2015)
32. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: ICML'16, vol. 48, pp. 1928–1937 (2016)
33. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. https://arxiv.org/pdf/1707.06347.pdf (2017)
34. Fang, M., Li, Y., Cohn, T.: Learning how to active learn: A deep reinforcement learning approach. In: EMNLP'17, pp. 595–605. Association for Computational Linguistics (2017)
35. Wu, J., Li, L., Wang, W.Y.: Reinforced co-training. In: ACL'18, pp. 1252–1262 (2018)
36. Qu, C., Ji, F., Qiu, M., Yang, L., Min, Z., Chen, H., Huang, J., Croft, W.B.: Learning to selectively transfer: Reinforced transfer learning for deep text matching. In: ICDM'19, pp. 699–707. Association for Computing Machinery (2019)
37. Hurst, B., Ooi, Y.H., Pedersen, L.H.: A century of evidence on trend-following investing. The Journal of Portfolio Management **44**(1), 15–29 (2017)
38. Baltas, N.: Trend-following, risk-parity and the influence of correlations. In: Risk-Based and Factor Investing, pp. 65–95. Elsevier (2015)
39. Wang, J., Zhang, Y., Tang, K., Wu, J., Xiong, Z.: Alphastock: A buying-winners-and-selling-losers investment strategy using interpretable deep reinforcement attention networks. In: KDD'19, pp. 1900–1908. Association for Computing Machinery (2019)
40. Sharpe, W.F.: The sharpe ratio. J. Portf. Manag. **21**(1), 49–58 (1994)

41. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
42. Gao, L., Guo, Z., Zhang, H., Xu, X., Shen, H.T.: Video captioning with attention-based lstm and semantic consistency. IEEE Trans. Multimedia **19**(9), 2045–2055 (2017)
43. Li, Y., Zheng, W., Zheng, Z.: Deep robust reinforcement learning for practical algorithmic trading. IEEE Access **7**, 1–1 (2019)
44. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, U., Polosukhin, I.: Attention is all you need. In: NIPS'17, Red Hook, NY, USA, pp. 6000–6010 (2017)
45. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv:1409.0473 (2014)
46. Nguyen, G.H., Bouzerdoum, A., Phung, S.L.: Learning pattern classification tasks with imbalanced data sets. Pattern Recognition, 193–208 (2009)
47. Wang, X., Matwin, S., Japkowicz, N., Liu, X.: Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In: AAI'13, pp. 174–186. Springer (2013)
48. Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV'17, pp. 2980–2988 (2017)
49. Lin, T., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal Loss for Dense Object Detection. In: ICCV'17, pp. 2999–3007 (2017)
50. Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: ACL'19, pp. 4171–4186 (2019)
51. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR'15. OpenReview.net (2015)
52. Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., Cottrell, G.W.: A dual-stage attention-based recurrent neural network for time series prediction. In: IJCAI'17, pp. 2627–2633. AAAI Press (2017)
53. Chen, W., Qiu, X., Cai, T., Dai, H.-N., Zheng, Z., Zhang, Y.: Deep reinforcement learning for internet of things: A comprehensive survey. IEEE Communications Surveys Tutorials **23**(3), 1659–1692 (2021)

Authors and Affiliations

Yang Li[1] · Zibin Zheng[1,2] · Hong-Ning Dai[3] · Raymond Chi-Wing Wong[4] · Haoran Xie[5]

Zibin Zheng
zhzibin@mail.sysu.edu.cn

Hong-Ning Dai
hndai@ieee.org

Raymond Chi-Wing Wong
raywong@cse.ust.hk

Haoran Xie
hrxie2@gmail.com

[1] School of Data and Computer Science, Sun Yat-sen University, 510275 Guangdong, Guangzhou, China

[2] National Engineering Research Center of Digital Life, Sun Yat-sen University, 510275 Guangdong, Guangzhou, China

[3] Department of Computer Science, Baptist University, Hong Kong, China

[4] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China

[5] Department of Computing and Decision Sciences, Lingnan University, Hong Kong, China