IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 August 2022, accepted 27 August 2022, date of publication 5 September 2022, date of current version 19 September 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3204056

APPLIED RESEARCH

Improving Pairs Trading Strategies Using Two-Stage Deep Learning Methods and Analyses of Time (In)variant Inputs for Trading Performance

WEI-LUN KUO[1], **WEI-CHE CHANG**[2], **TIAN-SHYR DAI**[3,4], **YING-PING CHEN**[5], **(Member, IEEE), AND HAO-HAN CHANG**[3]

[1]Institute of Data Science and Engineering, National Yang Ming Chiao Tung University (NYCU), Hsinchu 300093, Taiwan
[2]Institute of Computer Science and Engineering, National Yang Ming Chiao Tung University (NYCU), Hsinchu 300093, Taiwan
[3]Department of Information Management and Finance, National Yang Ming Chiao Tung University (NYCU), Hsinchu 300093, Taiwan
[4]Insurance Research Center, National Chengchi University, Taipei 11605, Taiwan
[5]Department of Computer Science, National Yang Ming Chiao Tung University (NYCU), Hsinchu 300093, Taiwan

Corresponding author: Tian-Shyr Dai (cameldai@mail.nctu.edu.tw)

This work was supported by the Ministry of Science and Technology (MOST) under Grant 109-2622-H-009-001-CC3, Grant MOST 110-2622-H-A49-001-, and Grant MOST 110-2634-F-A49-004-.

ABSTRACT A pairs trading strategy (PTS) constructs and monitors a stationary portfolio by shorting (longing) when the portfolio is adequately over- (under-)priced measured by a predetermined open threshold. We close this position to earn the price differences when the portfolio's value reverts back to the mean level. When the portfolio is significantly over- (under-)priced measured by another predetermined stop-loss threshold, we close the position to stop loss. This paper develops a two-stage deep learning method to improve the investment performance of a PTS. Note that the literature executes a PTS by selecting the best trigger threshold (a combination of open and stop-loss thresholds) from a restricted, heuristically-determined set of trigger thresholds. Such a design significantly degrades investment performance. However, selecting the best threshold from all possible thresholds yields a non-converged training problem. To resolve this dilemma, we propose in the first stage of our method a representative label mechanism by which to construct a set of candidate trigger thresholds based on all possible thresholds and then train a deep learning (DL) model to select the best from the set. Experiments demonstrate that the proposed first-stage method avoids the non-converged training problem and outperforms most state-of-the-art methods. To further reduce the trading risk, the second stage trains another DL with the profitability of each trade labeled by executing the PTS with trigger thresholds recommended in the first-stage mechanism to remove unprofitable trades. Compared to models that indirectly judge profitability by price movement similarity without considering the quality of the recommended trigger thresholds, our model produces higher win rates and average profits. Furthermore, we find that training with the PTS portfolio value process exhibiting time invariance clearly outperforms training with only time-varying stock/return processes, even though the latter training set contains more information. This is because unpredictable changes in market trends cause the model to learn time-varying patterns from the training set that may not apply to the testing set.

INDEX TERMS Pairs trading strategy, representative labeling, time (in)variant data, two-stage deep learning.

I. INTRODUCTION

A pairs trading strategy (PTS) is a popular, statistical arbitrage investment strategy that forms and trades market-neutral portfolios [1]. Rather than guessing hard-to-predict trends in financial markets, a PTS eliminates the risk of market tendency by longing (or shorting) several assets at the same time, according to specified investment weight ratios determined by various statistical methods [2]. The value of this portfolio, or "spread," oscillates around a mean price level and has a

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

low correlation with the tendency of financial markets. That is why the portfolio processes the market-neutral property. To construct a portfolio with this property, we need to find a group of assets (e.g., stocks in this paper) whose price processes cointegrate, as identified by the Johansen cointegration test [3]. Our PTS strategy longs (shorts) the portfolio when the spread significantly deviates from the mean price level to fall below a lower (exceed a higher) open threshold. Then the portfolio gets closed when the spread converges to the mean price level. The resulting earnings equal the price difference between the open threshold and the mean price level.

Appropriate customized, open thresholds for each stock pair thus determine PTS profitability. If the open threshold is too far from the mean price level, the portfolio is unlikely to open; if it is too close to the mean though, the resulting earning might be insufficient to cover the costs of transaction and price slippage. Furthermore, the statistical arbitrage property entails that the spread may occasionally fail to revert to the mean price level; such failures result in significant losses, as stated in Vidyamurthy [1]. Adding a stop-loss threshold creates a price level at which the portfolio closes once the spread diverges too far from the mean level. Here again, determining a proper threshold for each stock pair is critical because their spreads show different price patterns that vary with changes in financial markets.

In prior efforts to set the thresholds, referred to as actions in a reinforcement learning (RL) framework, Fallahpour *et al.* [4] and Kim and Kim [5] use a limited action set with 6 or 39 actions, respectively, which represent significant limits on investment performance. Therefore, we consider a much larger set of about 2800 open and stop-loss threshold recommendations determined by the maximum price deviations during the training set to cover all possible trading scenarios. Then we label each stock pair in the training set with one of 2800 thresholds that maximize PTS profits. In this effort though, methods based on regression- or classification-based deep learning (DL) fail to converge. To address this non-convergence problem, a representative labeling mechanism to select the recommendation threshold from 25[1] representative thresholds determined by the most frequently chosen thresholds or a k-means method is proposed in our previous conference work [6] and then further improved in this paper. We can relabel each stock pair with a representative threshold. Instead of learning from 2800-label stock pairs, we rely on 25-relabeled stock pairs. If we train a multi-scale residual network (abbreviated ResNet), as proposed by Li *et al.* [7], with the relabeled stock pairs, we achieve smooth, quick convergence. Our later experiments confirm that the investment performance of our representative labeling mechanism outperforms the options offered in prior research.

To further enhance PTS investment quality, Sarmento and Horta [8] and Lu *et al.* [9] indirectly predict and remove unprofitable stock pairs from trading without taking into account the quality of the recommended trigger thresholds. Sarmento and Horta [8] group stocks according to the OPTICS algorithm, then remove pairs whose stocks come from different groups. Furthermore, Lu *et al.* [9] use long short-term memory (LSTM) and wavelet convolutional neural network (CNN) to predict time-series anomaly properties. But these mechanisms do not necessarily determine unprofitability, so they can result in the removal of many profitable trades, which significantly reduces overall profits. Instead, this paper proposes a two-stage model to remove unprofitable trades without significantly sacrificing overall profits; the training data set comprises two parts. The first stage trains a ResNet model on the first part of the training data, labeled by the representative thresholds, to recommend open and stop-loss thresholds. Then to remove unprofitable pairs, the second part of the training data is first inputted into the first-stage model to obtain the recommended thresholds. We then trade each stock pair with the recommended threshold to obtain the profit/loss signal, as a label for the stock pair to train the second-stage model. For each stock pair in the testing set, we also use the first model to recommend an open and stop-loss threshold and the second model to remove unprofitable pairs from trading. This two-stage model yields better win rates and higher Sharpe ratios across all our experiments.

Because frequent dramatic changes in financial markets alter patterns of stock price and return processes, it becomes difficult for machine learning algorithms to capture changing patterns, even when using many features and various data lengths [10]. Thus, researchers tend to train their models using a limited amount of the most recent historical market data; ancient data and corresponding embedded information get discarded. But the cointegration test proposed by Johansen [3] guarantees that the statistical properties of the spread process do not vary with time. This feature effectively improves the performance of the PTS model if we prolong the training period, such that we do not need to tune the hyperparameter that controls the length of the training period. Even if the spread process contains less information than the price processes of stock pairs,[2] models trained on spread process data still outperform those trained on stock pair data.

In Section II-A, we review prior PTS research that relies on quantitative and machine learning models. Section II-B outlines how we construct stock pairs that possess cointegration properties and provide the definitions of PTS reward functions. With Section III, we detail the construction of the optimal combination of the open and stop-loss thresholds (referred to as the ‘‘trigger threshold’’) and the representative labeling mechanism adopted to address the non-convergence training problem. Then in Sections IV-A and IV-B, we describe how we incorporate the multi-scale ResNet into our PTS model, as well as the design of the two-stage model. The experimental results in Section V confirm the superiority of our two-stage models; as we explain, the

[1] This value is determined by the elbow method.

[2] The spread process can be derived from the price processes of stock pairs, as in Equation (2).

time-invariant property provided by cointegration resolves the problem of changing data patterns due to varying financial markets. Section VI concludes.

II. PRIOR LITERATURE AND REQUIRED BACKGROUND KNOWLEDGE

In this section, we review previous applications of PTS and provide a brief survey of the cointegration method we use to construct stock pairs eligible for PTS, together with the corresponding investment ratios. We also describe the motivation behind the proposed two-stage model.

A. PRIOR LITERATURE

Krauss [11] classifies techniques for finding stock pairs eligible for PTS and improvements for PTS strategies into several approaches, including the distance approach, the cointegration approach, time-series models, stochastic control theory, and other techniques. Our stock pair generation method is based on the cointegration approach, which Rad *et al.* [12] and Huck and Afawubo [13] identify as preferable. In addition, Engle and Granger [14] and Johansen [15] develop different statistical tests to determine whether the price processes of a logarithmic stock pair possess cointegration properties. That is, a linear combination of two logarithmic price processes of constitute stocks makes the resulting value process of this two-stock portfolio into a stationary process. The stationary property ensures that statistical properties, such as the value's mean and variance, do not change with time. Thus it is possible to buy (sell) the portfolio when its value is below (above) the mean, then close the position to cash out when the value converges back to the mean. Vidyamurthy [1] and Rad *et al.* [12] use these tests to detect stock pairs eligible for PTS. An effective PTS also be applied to reduce the variance (or risk) in investment portfolios [16], establish optimal asset allocations [17], and support trades of new financial products like cryptocurrency [18], [19].

Machine learning techniques, as first proposed by [20], promise to improve PTS performance. In particular, reinforcement learning (RL) can determine open and stop-loss thresholds for PTS. Fallahpour *et al.* [4] enumerate 39 actions (i.e., 39 trigger thresholds), which enables them to reduce the threshold selection problem to a multi-armed bandit problem, solved using a single-state RL model. However, this naive mechanism cannot capture various properties of different stock pairs, so it is outperformed by other approaches, in terms of our experimental results. Kim and Kim [5] instead use a deep Q-network (DQN) and heuristically set six overly simplistic actions, which significantly limits the profitability of their approach. In addition, they train each PTS-eligible stock pair with a DQN, which necessitates a large number of DQNs. But in line with their observations, we find that cointegration properties for most stock pairs are not durable over a long period; very few stock pairs contain enough data to train the DQN. Therefore, we train our machine learning model instead on trading data from all stock pairs, which produces recommended thresholds for all stock pairs. Some variations of PTS include a double DQN proposed by Brim [21], with three actions (hold, buy, sell), that seeks to predict the trend of the spread, though a low win rate limits their model's applicability. Instead of using open and stop-loss thresholds, Xu and Tan [22] predict open and stop-loss timing for PTS, which they use to form a return-maximized portfolio with a deterministic policy gradient method. In addition, Hsu *et al.* [23] take advantage of opinions from social media to predict spread price movements.

To reduce PTS risk, Sarmento and Horta [8] use the OPTICS algorithm and divide the stocks into groups, according to their average return processes. They then remove PTS pairs with stocks from different groups. In our experiments, their approach slightly improves the win rate and reduces the maximum drawdown; however, it discards many profitable trades, such that it significantly reduces overall investment performance. When Lu *et al.* [9] use the time-series anomaly detection mechanism proposed by Huang *et al.* [24] to label anomalies of the price processes, they can combine LSTM and continuous wavelet CNN to predict structural breaks, which they interpret as losing cointegration properties. But errors in labeling anomalies are difficult to avoid, which biases training efforts to detect structural breaks. Therefore, we propose a two-stage model that determines the optimal open and stop-loss thresholds in the first stage, then detects and removes unprofitable pairs in the second stage. With experiments, we show that this two-stage approach achieves a better win rate, higher trading opportunities, and greater overall profits than filtering pairs with the OPTICS algorithm. Our approach also incurs fewer risks of negative returns than the structural break detection approach. Rather than using RL, we adopt DL with representative labeling mechanisms to find recommended open and stop-loss thresholds. To capture complex features or patterns in financial markets, we adopt the residual network (ResNet) model proposed by He *et al.* [25]; their extensive empirical data affirm that ResNets are simpler to optimize and also achieve higher learning precision because ResNets include more hidden layers. ResNet is extended by Li *et al.* [7] from a single scale to multiple scales by adding convolution kernels of various sizes to adaptively detect data features from different aspects. By combining representative labeling with multiscale ResNet, our proposed method yields superior investment performance.

Financial markets constantly change with time, mainly due to black swan events such as the COVID-19 pandemic and quantitative easing, which caused stock markets to plummet and then soar during the first half of 2020. Such time-based heterogeneity causes trading patterns to vary over time and creates difficulties for DL algorithms, even with many features and long window sizes [10]. Prior literature [26], [27], [28], [29], [30], [31], [32] often limits the length of the most recent historical trading data, to train machine learning models to predict contemporary future market patterns; for example, they might use January 2021 trading data to train the model to forecast February 2021 markets, use

February data to train the model to predict the March market, and so on. But this approach junks information of ancient historical data and still occasionally yields unstable investment performance because it fails to consider whether market tendencies change during the training or the testing period. Zhang *et al.* [33] address this problem by decomposing stock price series into high- versus low-frequency waves with discrete Fourier transforms. As an alternative approach, we train the proposed model with stationary spread processes (i.e., the trends of the spreads do not change with time), as confirmed by cointegration tests [34]. Our experiments accordingly show that extending the length of the training period allows our model to capture more trading patterns and improve PTS performance, without creating vulnerability to drastic market changes. Furthermore, due to the stationary property, even when a spread process contains less information than the return or price processes of a stock pair, training the model on spread process data still outperforms models trained on returns or price processes.

B. CONSTRUCTING PTS WITH COINTEGRATION APPRPAOCHES

We divide a trading duration—a business day to fit the intraday trading setting in this paper— into a formation period and a trading period, as illustrated in Figure 1. During the formation period, the stock tick data is used to generate stock pairs eligible for PTS. Then we use our machine learning model (introduced later) to predict feasible open and stop-loss thresholds for each PTS-eligible stock pair for trading in the trading period. With a cointegration approach [1], [8], [12], [35], [36], we identify PTS-eligible stock pairs from a stock pool, such as 0050. TW constituent stocks from the Taiwan stock market. If we let the i-th pair be composed of stocks S_1^i and S_2^i, and the capital invested in these two stocks to be $\beta_1^i : \beta_2^i$ (if the stock pair is eligible), we can extract logarithmic stock price processes $\ln S_1^i(t)$ and $\ln S_2^i(t)$ from the formation period to form a two-dimensional vector $y(t) \equiv (\ln S_1^i(t), \ln S_2^i(t))'$. The test of the cointegration property of $y(t)$ relies on the Johansen cointegration test [3], with the following vector error correction model (VECM)

$$\triangle y(t) = \Pi y(t-1) + \sum_{i=1}^{p-1} D_i \triangle y(t-i) + \epsilon_t, \tag{1}$$

where $\triangle y(t) \equiv y(t) - y(t-1)$, the rank of the 2×2 matrix Π denotes the number of cointegration relations, $p-1$ denotes the VECM order, D_i is a 2×2 matrix, and ϵ_t denotes a 2×1 white noise vector. We follow Lütkepohl *et al.* [37] and use a power test, which decomposes Π into $\alpha\beta'$, where the 2×1 cointegration vector $\beta \equiv (\beta_1^i, \beta_2^i)'$ determines the ratios of the capital invested in the two stocks. If the i-th stock pair S_1^i and S_2^i passes the cointegration test, we construct a portfolio by investing the two stocks, according to the ratio $\beta_1^i : \beta_2^i$. The spread process of this portfolio,

$$P^i(t) \equiv \beta_1^i \ln S_1^i(t) + \beta_2^i \ln S_2^i(t), \tag{2}$$

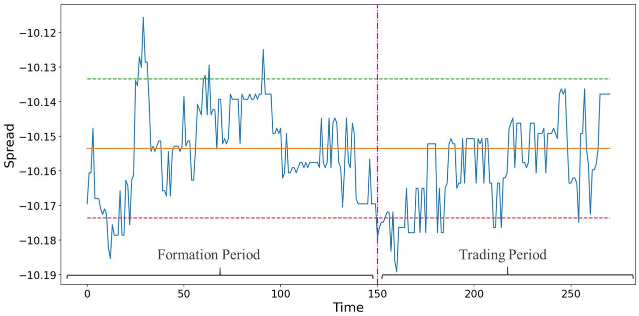


FIGURE 1. Real example for cointegration model calibrations. This figure illustrates real cointegration calibration for Cheng Shin Tyre (with ticker number 2105) and Shin Kong Financial Holdings (2888) on May 20th, 2016. The x-axis denotes the elapsed time from the opening of the stock market. The time span for a trading day is divided into the formation and trading periods. The blue curve reflects the change in the spread process defined in Equation (2). The orange line denotes the mean level of −10.15. The investment weight ratio β is (1, −7.9)′. The green (red) dash line denotes the value of the mean level plus (minus) a standard deviation of the spread process.

is mean-reverting; that is, it oscillates around the mean level of the spread, $E(P^i(t))$. We could also measure the $P^i(t)$ variation by calculating its standard deviation σ^i. A sample cointegration calibration of two 0050.TW constituent stocks, Cheng Shin Tyre (2105) and Shin Kong Financial Holdings (2888), are illustrated in Figure 1. We use the stock trading data during the formation period to calibrate the VECM for determining the trigger threshold for each stock pair. Then we use the threshold to trade the stock pair during the trading period. The mean-reverting property of the spread defined in Equation (2) is illustrated by the blue curve moving around the mean level of -10.15. The magnitude of σ^i, denoted by the distance between the mean level and the green (or red) dashed line, will be used to tune the open and stop-loss thresholds described as follows.

The profit (or loss) to purchase the aforementioned stock pair portfolio at time τ and sell it at τ' can be expressed as the product of the investment amount c and the difference of the spread:

$$\begin{aligned} c &\times \left(P^i(\tau') - P^i(\tau)\right) \\ &= c \times \left(\beta_1^i \ln \frac{S_1^i(\tau')}{S_1^i(\tau)} + \beta_2^i \ln \frac{S_2^i(\tau')}{S_2^i(\tau)}\right) \\ &\cong \frac{c\beta_1^i}{S_1^i(\tau)} \left[S_1^i(\tau') - S_1^i(\tau)\right] + \frac{c\beta_2^i}{S_2^i(\tau)} \left[S_2^i(\tau') - S_2^i(\tau)\right] \end{aligned} \tag{3}$$

where $\ln \frac{S_j^i(\tau')}{S_j^i(\tau)}$ denotes the return rate for investing S_j^i over the time period $[\tau, \tau']$, and $\frac{c\beta_j^i}{S_j^i(\tau)}$ denotes the number of shares for trading S_j^i at time τ.[3]

Due to the mean-reverting nature of Equation (2), we can short (long) the portfolio when the spread $P^i(t)$ soars (falls)

[3] We long (short) S_j^i if the number of shares is positive (negative).

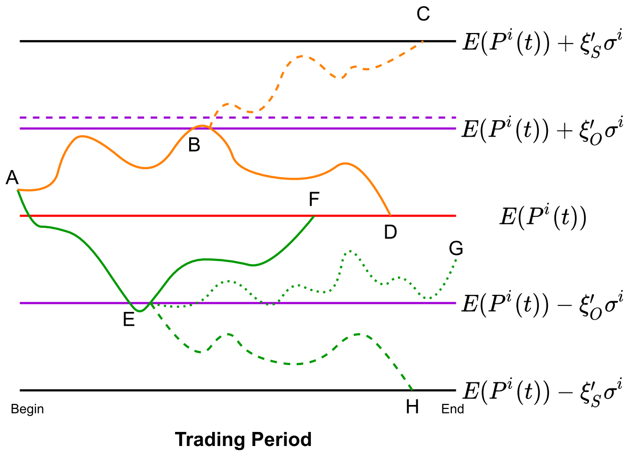

C
$E(P^i(t)) + \xi'_S\sigma^i$
B
$E(P^i(t)) + \xi'_O\sigma^i$
A
F
D
$E(P^i(t))$
G
E
$E(P^i(t)) - \xi'_O\sigma^i$
Begin
H
End
$E(P^i(t)) - \xi'_S\sigma^i$
Trading Period

FIGURE 2. Trading period scenarios. The red, purple, and black lines denote the mean of $P^i(t)$, the thresholds for opening the portfolio, and the thresholds for stopping losses, respectively. The values are listed to the right of the lines. The orange and green curves denote the change of spread processes over the trading period. We would long (short) the portfolio if the spread process begun from A goes down (up) to reach $E(B)$, as denoted by the green (orange) curves. With solid and dashed curves, we indicate actions that close the portfolio to gain profit or to stop loss, respectively. The dotted curve indicates that the portfolio is forced to close at the end of the trading period.

to reach the upper (lower) open threshold (denoted by purple lines), then close the position when it converges to reach the mean price level (denoted by the red line) to earn the profit, as illustrated in Figure 2. To increase the profit in Equation (3) and cover the transaction cost without significantly reducing trading opportunities, we find a suitable open threshold, defined as the product of a scalar ξ'_O and the spread process's volatility σ_i. Then we find another stop-loss threshold, defined as the product of a scalar ξ'_S and σ_i, to prevent occasional failures of the mean-reverting property from seriously eroding profits. The intersection of the spread $P^i(t)$ with either element of the trigger threshold (ξ'_O, ξ'_S) determines the timing to long/short the portfolio or to stop loss, respectively. Specifically, if the spread $P^i(t)$ reaches the upper open threshold (denoted by node B), we short the portfolio with the value investment ratio $\beta^i_1 : \beta^i_2$ for stocks S^i_1 and S^i_2. After shorting the portfolio, $P^i(t)$ may still reach node C, in which case we close the portfolio to stop loss. Otherwise, it may fall to node D, in which case we close the portfolio to gain a profit. If $P^i(t)$ falls to the lower open threshold (denoted by node E), we instead long the portfolio, after which $P^i(t)$ may still fall to node H, in which case we close the portfolio to stop loss. Otherwise, it may reach node F, prompting us to close the portfolio to earn a profit. Finally, the portfolio may remain open at the end of the trading period, say, node G. In this case, the portfolio is forced to close to avoid incurring risks related to keeping cross-day positions.

Note that the situation of simultaneously longing and shorting the portfolio cannot occur under our pairs trading strategy. Recall that we long the portfolio when the spread reaches the lower opening threshold. For the action to short the portfolio, the spread process must increase to reach the upper opening threshold. However, the spread process should come across the mean price level to close the long position before reaching the upper opening threshold. This means that we cannot short the portfolio before closing the previously open position. Similarly, we cannot long the position before closing the previously shorted position. In addition, we cannot simultaneously long and short the portfolio since a spread cannot be simultaneously smaller than the lower open threshold and larger than the upper threshold because the upper threshold is larger than the lower one.

C. MOTIVATIONS

In the above survey, we find that many studies select trigger thresholds from a heuristic and limited set of thresholds. While searching from such a limited set clearly limits PTS investment performance, searching from all possible thresholds results in unconverged training. The motivation of our proposed RLM is that it reduces the number of candidate thresholds to eliminate unconverged training without significantly harming the investment performance. In addition, Fallahpour *et al.* [4], Kim and Kim [5], Brim [21], and Kim *et al.* [38] train each reinforcement learning model with the sequential trading data of a specific stock pair. Such a design causes these papers to focus on trading on selected stock pairs since it is impractical to train the many models needed to cover all possible PTS-eligible stock pairs. Our paper uses deep learning to learn simultaneously occurring trading data of different stock pairs, and thus predicts trigger thresholds for all PTS-eligible stock pairs. To improve PTS performance by reducing unprofitable stock pairs, the literature indirectly judges profitability by the similarity of stock price processes of a stock pair [8] or the occurrence of structural breaks [9]. Our second-stage model directly predicts profitability by learning the investment performance obtained from executing PTS with the trigger thresholds recommended in the first stage. Numerical experiments in Tables 5 and 6 confirm the superiority of our two-stage model.

III. REPRESENTATION LABELING

It is challenging to train naive DL methods to obtain feasible open and stop-loss thresholds for PTS. Generating candidate trigger thresholds heuristically, as exemplified by Fallahpour *et al.* [4] and Kim and Kim [5], harms PTS investment performance, as we detail in Section V. But considering all possible trigger thresholds would cause a non-convergence training problem. To improve trading profits without increasing training difficulty too much, a representation labeling mechanism (RLM), as depicted in Figure 3, is first proposed in our previous conference work [6]. For clarity, our current paper describes the details implementations of RLM with our revisions for generating representative trigger thresholds based on the statistics for the PTS profits obtained from training set data. In Section III-A, we explain how we divide daily data into formation and trading periods and perform data preprocessing (step 1). To obtain information on eligible stock pairs and investment ratios, we apply the Johansen

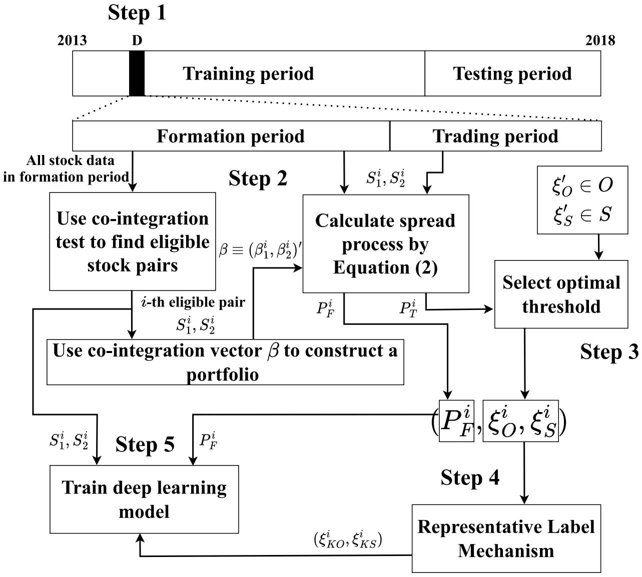

Step 1
2013
D
2018
Training period
Testing period
Formation period
Trading period
All stock data in formation period
Step 2
S_1^i, S_2^i
$\xi'_O \in O$
$\xi'_S \in S$
Use co-integration test to find eligible stock pairs
$\beta \equiv (\beta_1^i, \beta_2^i)'$
Calculate spread process by Equation (2)
Select optimal threshold
i-th eligible pair
S_1^i, S_2^i
P_F^i
P_T^i
Use co-integration vector β to construct a portfolio
Step 3
$(P_F^i, \xi_O^i, \xi_S^i)$
S_1^i, S_2^i
Step 5
P_F^i
Step 4
Train deep learning model
(ξ_{KO}^i, ξ_{KS}^i)
Representative Label Mechanism

FIGURE 3. Recommending PTS thresholds on the basis of RLM.

cointegration test to the formation period of a trading day D, as described in Section II-B (step 2). Then we label the optimal trigger threshold for each stock pair, as described in Section III-B (step 3); relabel each pair with a representative threshold, as described in Section III-D (step 4); and train the DL model with stock pairs and representative labels retrieved from each trading day in the training period, as described in Section IV-A (step 5).

A. PREPROCESSING OF DATA SET

We use tick data for the constituent stocks of the Taiwan Top 50 ETF (0050.TW) between 2013 and 2018. We choose the constituent stocks of Taiwan Top 50 ETF (0050.TW) since these stocks have better liquidity, thus preventing higher price slippage and bid-ask spread from eroding the PTS profits. We update the list of constituent stocks for each trading day so that the pairs eligible for pairs trading on each trading day are selected from the up-to-date list of 0050.TW constituent stocks via the Johansen cointegration test. However, unlike our paper, Endres and Stübinger [39] and Liu *et al.* [40] that adopt high-frequency day trading, many other PTS studies use non-day trading. Note that the dramatic changes in financial markets that make stock pairs feasible for pairs trading in one day may not continue to the next trading day. This instability and frequent stock price jumps at the beginning of trading days typically lead to significant trading losses for non-day trading strategies. We found that using day trading instead significantly increases the profits and Sharpe ratios.[4] Our strategy does not hold positions overnight to avoid the breaking events that usually occur during the closing of the market from eroding PTS profits.[5]

Step 1 in Figure 3 describes the data preprocessing, in which we divide the 2013-2018 period into non-overlapping training and testing periods. The stock tick data for each business day D in the training period generates the labels and spread features required to train the RLM model, whose performance can then be verified on each business day of the testing period. Daily trading takes place from 9:00 a.m. to 1:30 p.m. each business day, divided into the formation period (the first 166 minutes, ignoring the beginning of the first 16 minutes) and the trading period (rest of the business day), as illustrated in Figure 1. We cut the first 16 minutes of trading data since the high volatility of the stock price influences the effectiveness of the cointegration test. As this test also requires sufficiently long time series data to ensure its robustness, we follow [9] by setting the length of the formation period to 150 minutes for the cointegration test, and use the remaining time to execute the PTS. We use tick data from the formation period to calculate each half-minute's weighted average stock price. As described in Section II-B, we examine whether the resultant time series possesses the mean-reverting property by the Johansen cointegration test. Then we derive corresponding investment ratios β defined in Equation (2) for each PTS-eligible stock pair by calibrating Equation (1). The spread process of the i-th stock pair P_F^i (P_T^i) is constructed by substituting the price processes of its constituent stocks S_1^i and S_2^i during the formation period (trading period) into Equation (2). Note that the spread process is stationary; that is, statistical properties such as the mean and variance of the spread process do not change with market trends. Increasing the length of the training period in this way makes it possible for machine learning algorithms to capture more time-invariant features, which improves PTS performance.[6]

B. LABELING: SEARCHING THE OPTIMAL TRIGGER THRESHOLD OVER ALL POSSIBLE THRESHOLDS

We label the i-th stock pair with the spread process in the formation period P_F^i by the optimal trigger threshold (ξ_O^i, ξ_S^i) that maximizes PTS profits by trading the stock pair's spread P_T^i during the trading period. Specifically, we long (short) the portfolio when P_T^i falls to $E(P^i(t)) - \xi_O^i \sigma_i$ (rises to $E(P^i(t)) + \xi_O^i \sigma_i$) and close the portfolio to earn profits when the spread reverts to $E(P^i(t))$. We impose a stop-loss when the process continuously plummets to $E(P^i(t)) - \xi_S^i \sigma_i$ (or soars to $E(P^i(t)) + \xi_S^i \sigma_i$), as illustrated in Figure 2. The profit for executing PTS with P_T^i during the trading period can be evaluated based on Equation (3). Note that the search space composed of all possible trigger thresholds

[4] Also, as much of the PTS literature uses easily obtained daily close price data to determine trading decisions, their models cannot consider day trading due to data limitations. We are not limited by this constraint since we instead use intra-day tick data.

[5] The transaction cost also falls from 0.3% to 0.15% for day trading in the Taiwan Stock Exchange.

[6] The statistical characteristics of training features used in many extant machine learning algorithms [e.g., 33] vary with market trends. Thus training periods of heuristically selected lengths may significantly influence investment performance.

is huge since the threshold ξ_O^i (or ξ_S^i) can be an arbitrary positive real number. This makes searching for the optimal trigger threshold (or the labeling process) intractable. Prior literature [4], [5] either uses fixed trigger thresholds or finds optimal trigger thresholds from a limited set, determined heuristically, which significantly deteriorates investment performance, as we verify subsequently. To search for the optimal trigger threshold over the whole solution space without incurring excessive computational resources, we first collect all the spread processes of all business days in the training period. Then, we define the maximum standardized deviation for all spread processes during the formation period by mimicking the z-score formula as

$$M_a \equiv \max_{i\in\text{all spreads}} \left(\max_{t\in\text{formation period}} \left(\left| P_F^i(t) - \mathrm{E}\left(P_F^i(t)\right) \right| \right) / \sigma_i \right). \tag{4}$$

Similarly, we can also calculate M_c, the maximum standardized deviation for the processes that converge to the mean level before the market close by replacing "all spreads" in Equation (4) with "all converged spreads." To construct a feasible stop-loss threshold set S, we discretely enumerate equal space samples from the range determined by M_a In turn, S is defined as $\{1.5, 1.5 + 1 \times 0.5, 1.5 + 2 \times 0.5, \ldots, \lceil M_a \rceil\}$. To ensure that the estimated profit (proportional to the distance between the mean price level and the open threshold $\xi_O^i \sigma_i$) covers the transaction cost, the open threshold should generally be greater than $0.5\sigma_i$. Thus, we construct the open threshold set O by enumerating samples from the range determined by 0.5 and M_c. The set O is defined as $\{0.5, 0.5 + 0.5, 0.5 + 2 \times 0.5, \ldots, \lceil M_c \rceil\}$, and all trigger thresholds (ξ'_o, ξ'_s) are generated by separately selecting the open threshold ξ'_o from set O and the stop-loss threshold ξ'_S from set S. In addition, we enforce condition $1.5 \times \xi'_o < \xi'_s$ to prevent the open threshold from coming too close to the stop-loss threshold, which would increase the chance to close the portfolio to stop the loss right after opening the portfolio, thereby deteriorating the investment performance. To filter out stock pairs unsuitable for PTS, we add one more trigger threshold $(10, 25)$ with extremely high open and stop-loss thresholds to reflect no trading actions. Then we trade the stock pairs S_1^i and S_2^i by using the spread in the trading period P_T^i and the trigger threshold (ξ'_O, ξ'_S) to determine the timing for opening and closing the portfolio, as illustrated in Figure 2. The trading profit can then be calculated by Equation (3). The optimal trigger threshold that maximizes PTS profit is

$$(\xi_O^i, \xi_S^i) \equiv \text{argmax}_{\xi'_O \in O, \xi'_S \in S} \left[\text{Profit}\left(P_T^i, \xi'_O, \xi'_S\right) \right]. \tag{5}$$

We label the i-th stock pair with the spread P_F^i by the optimal trigger threshold (ξ_O^i, ξ_S^i) defined above to train the proposed machine learning models. About $N(\approx 300)$ out of 2800 trigger thresholds[7] have been selected by at least one stock pair during different training periods in our later experiments. Note that many trigger thresholds enumerated by the procedure mentioned above are never chosen as the optimal threshold by any stock pair, probably because the stock price is quoted as integral multiples of basic units (i.e., ticks) rather than continuously. Many trigger thresholds do not fit discrete changes of the spread process defined in Equation (2), due to discrete stock price quotes, and therefore will never be selected as optimal trigger thresholds. This rationale explains why heuristically selecting trigger thresholds (e.g., [5]) might significantly deteriorate investment performance. Deriving feasible trigger thresholds from discrete changes in the spread process can be very hard, so we discretely enumerate many thresholds and use Equation (3) to calculate profits to filter unprofitable thresholds.

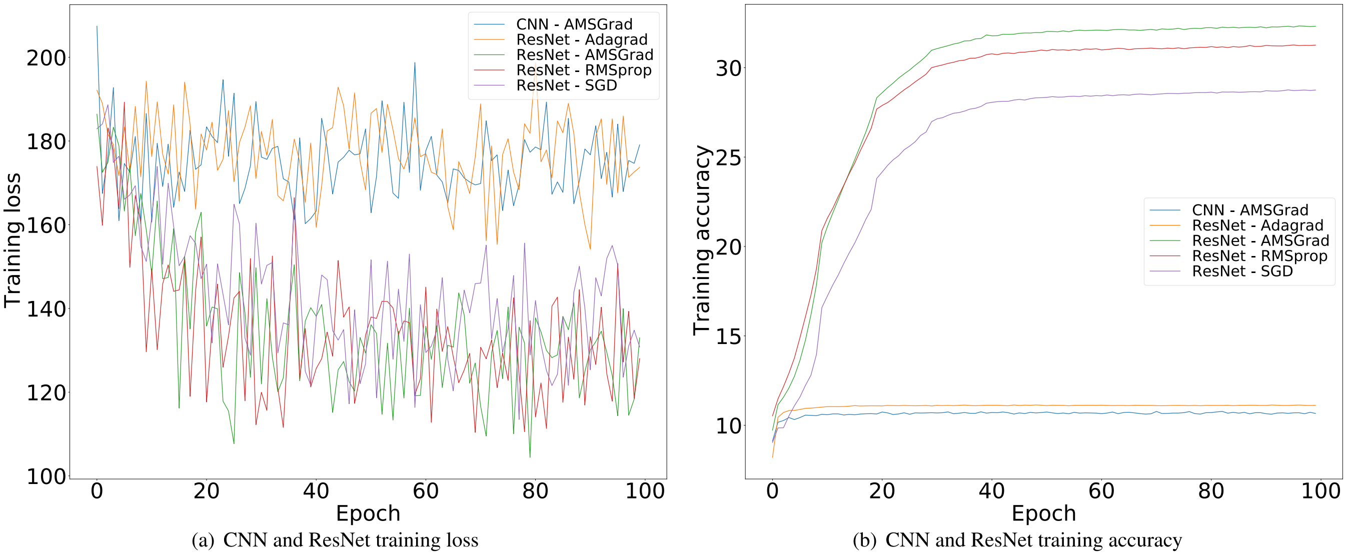
C. NON-CONVERGENT TRAINING OF NAIVE REGRESSION-/CLASSIFICATION-BASED DL

Since feasible open and stop-loss thresholds form a range $[0.5, M_c]$ and $[1.5, M_a]$, it is natural to use a regression-based deep neural network (RDNN) to predict an optimal trigger threshold (ξ_O^i, ξ_S^i). We optimize the training of RDNN by defining the loss as the mean square error (MSE) between the predicted trigger threshold and the optimal one defined in Equation (5). To achieve convergent training results, we have tried different combinations of inputs like the stock price, return, and the spread processes extracted from the formation period. We have also attempted many popular solutions to solve the non-convergent training problems, such as tuning learning rates, changing activation functions, and optimizers. Still, all fail. It seems that RDNN fails to capture discontinuous relationships between open and stop-loss thresholds and profits. Specifically, a minor shift in either threshold can significantly change the profit, as illustrated in Figure 2. For example, shifting the upper open threshold from the upper purple solid line to the dashed one removes the chance to short the portfolio at point B for the solid orange spread and sacrifices the corresponding profit.

Instead of adopting the regression-based method, we could select an optimal trigger threshold for each PTS-eligible stock pair from all possible trigger thresholds using classification-based approaches. Here we use cross-entropy as the loss function and train with different combinations of inputs and techniques to resolve non-convergence problems as we examine the regression-based method. Part of our experimental results[8] are illustrated in Figure 4. It can be observed from Figure 4(a) that the training losses oscillate significantly regardless of the changes in DL models and optimizers. Figure 4(b) also illustrates the highest training accuracy could only achieve around 30%. In the next subsection, we address this non-convergent training problem with RLM, which significantly reduces the number of labels without sacrificing the quality of trigger thresholds. Our experiments show that

[7] N changes with the training set data.

[8] Other non-convergent training results like changing activation functions and tuning learning rates are not illustrated for simplicity.



(a) CNN and ResNet training loss

(b) CNN and ResNet training accuracy

FIGURE 4. CNN and ResNet training losses.

RLM improves training accuracy and resulting PTS investment performance.

D. REPRESENTATION LABELING MECHANISM

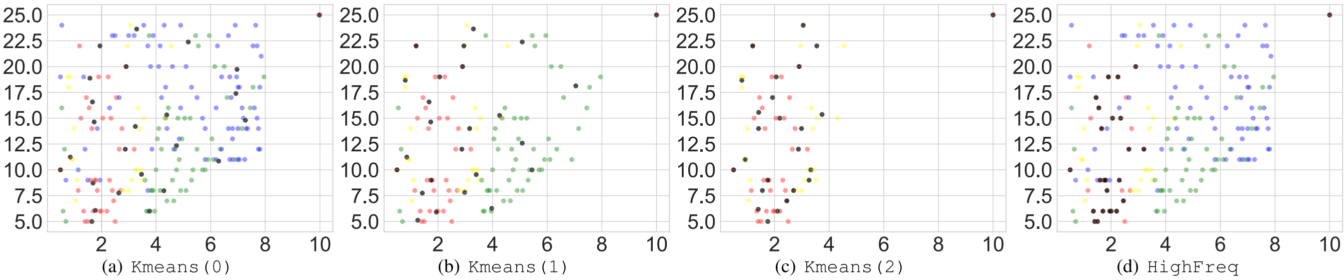
Unlike past RL-based works such as Kim and Kim [5] and Fallahpour *et al.* [4] that merely learn 6 to 39 actions (i.e., trigger thresholds), we combine deep learning models with an RLM to train 25 representative labels (determined by the elbow method) that represent all 2800 trigger thresholds, as specified in Section III-B, to resolve the non-convergent training problem. Section V shows that training with RLM outperforms an RL approach with heuristically selected trigger thresholds.

The RLM maintains trading performance by ensuring properly selected representations, and it resolves the training convergence problem by reducing the number of labels. Specifically, the i-th spread process defined in Equation (2) can be divided into P^i_F (belonging to the formation period) and P^i_T (trading period). We substitute P^i_T into Equation (5) to extract the optimal trigger threshold (ξ^i_O, ξ^i_S) that maximizes the benefit of trading the i-th stock pair. Then (ξ^i_O, ξ^i_S) can be viewed as the label for P^i_F; the trigger threshold distributions are illustrated in Figure 5(a). We use pink, yellow, green, and blue nodes to reflect the magnitudes of the probability of choosing a corresponding trigger threshold as the optimal one. By excluding trigger thresholds with probabilities lower than 0.1% and 0.5%, we obtain Figures 5(b) and 5(c), respectively. The trigger threshold distribution clearly is widespread and far from uniform. Moreover, the probability of selecting some trigger thresholds (like pink or yellow nodes) as optimal ones is much higher than that of other thresholds. This significant lack of smoothness could explain why the training of regression-based DL fails to converge discussed in Section III-C.

We address the lack of training convergence problem by setting representation trigger thresholds, according to either k-means or thresholds with the top-k highest probabilities. With the former method, we partition all trigger thresholds into a reasonable number of clusters by the k-means algorithm; the cluster number 25 is determined by the elbow approach. The set of representation trigger thresholds $\mathbb{R}$ is defined as the centers of the previously mentioned cluster, which we call Kmeans(0). The optimal trigger threshold for the i-th spread process is relabeled by picking one of the representation thresholds that maximizes profit, as follows:

$$(\xi^i_{KO}, \xi^i_{KS}) \equiv \text{argmax}_{(\xi'_O, \xi'_S)\in\mathbb{R}} \left[\text{Profit}\left(P^i_T, \xi'_O, \xi'_S\right)\right]. \quad (6)$$

Note that each representation threshold selected by Kmeans(0) (black nodes) basically does not coincide with any trigger threshold because each cluster center is calculated as the averaging of nearby trigger thresholds belonging to the same cluster. However, a slight shift in the threshold, like moving from the upper purple solid line to the dashed one in Figure 2, could significantly change the investment profit as mentioned above. To prevent disturbances in low-probability trigger thresholds from degrading the quality of representation labels, k-means can be applied to trigger thresholds with probabilities larger than 0.1% and 0.5%, as illustrated in Figures 5(b) and 5(c), respectively. The resulting representation label settings are named Kmeans(1) and Kmeans(2), respectively. Besides, to ensure that each representative label coincides with a trigger threshold, we could choose, as representation trigger thresholds, those trigger thresholds with the top 25 highest probabilities, as shown in Figure 5(d), which we refer to as the HighFreq label setting. Section V compares these RLMs to find the best one.



(a) Kmeans(0) (b) Kmeans(1) (c) Kmeans(2) (d) HighFreq

FIGURE 5. Trigger threshold distributions and representation thresholds. The *x* and the *y* axes denote the open and the stop-loss thresholds, respectively. Pink, yellow, and green nodes denote the trigger thresholds selected with probabilities larger than 1%, 0.5%, and 0.1%, respectively. Blue and black nodes denote other low-probability trigger thresholds and representative thresholds, respectively. Here we illustrate the trigger thresholds distribution for the year 2016. The distributions of other years are similar, so we ignore them for simplicity.

IV. CONSTRUCTIONS OF TWO-STAGE LEARNING MODELS

Given the stock/spread prices processes and representative thresholds recommended by different RLMs in Section III-D as inputs, we can compare the training effectiveness of RLMs and deep learning models to select the best to use in step 5 of Figure 3, as described in Section IV-A. To improve the win rate and reduce PTS risk, we train the second-stage model on the basis of the aforementioned threshold selection model, which prevents unprofitable pairs from trading (see Section IV-B). By combining DL with RLM (first stage) and then unprofitability detection and removals (second stage), we construct a two-stage model that can more effectively reduce PTS risk than [8] and [9] do.

A. MODELS FOR SELECTING PROPER REPRESENTATIVE TRIGGER THRESHOLDS

Here we construct the trigger threshold selection mechanism (step 5 of Figure 3). The inputted features for the i-th stock pair, x_i, can be formed by the pair's stock price processes, return processes, and (or) the spread process. For example, we can define $x_i \equiv \left[S_1^i, S_2^i, P_F^i\right]$, such that the input is formed by the stock price processes of the i-th pair S_1^i and S_2^i during the formation period, plus the corresponding spread process P_F^i determined in Equation (2). The input x_i with length 300 (i.e., the number of half-minute data in the 150-minute formation period) gets extended to 512 by padding the remaining positions with zeros. We number each of the 25 representative thresholds with a unique integer within the range $[1, 25]$. The label of the stock pair i, y_i, is the number of the representative threshold that maximizes the trading profit, as illustrated in Equation (6). We train the plain CNN, a single-scale ResNet [25], and a multi-scale ResNet [41] with input x_i and ground truth y_i for each stock pair i from the training period. The input x_i can have three channels (e.g., spread and the two stock price (or return) processes), two channels (e.g., two stock price (or return) processes), or one channel (e.g., the spread process). The CNN includes a one-dimensional convolutional layer with 25 1×5 kernel maps. The output gets sent to the batch normalization layer [42] to stabilize and speed up the training process; we select Leaky-ReLU after trying different activation functions. The results pass through a one-dimensional convolutional layer with 50 kernel maps, a layer with 100 kernel maps, and a layer with 200 kernel maps, sequentially; the final outputs are then sent to a fully connected layer. The single-scale ResNet uses a single size-3 convolution kernel, which applies to one chain of residual blocks. The three-scale ResNet adds size-5 and size-7 convolution kernels, as well as two corresponding chains of blocks.[9] The features extracted by the three convolution kernels (i.e., outputs from the three chains of residual blocks) are concatenated to form a feature vector, which gets sent to a fully connected network.

To find the optimal settings to achieve the best training results, we have attempted different settings of optimizers and activation functions; training accuracy and loss for part of our experiments that use different kinds of optimizers are illustrated in Figure 6. Training accuracy refers to the percentage of correct predictions of all pairs in the training set; training loss is measured according to cross-entropy. The training accuracy for the CNN model, denoted by the orange curve, increases slowly; the training loss oscillates significantly. Thus we use a residual network, which employs more hidden layers to capture various features embedded in complex financial markets. Although the three-scale ResNet with AMSGrad, RMSprop, and SGD optimizers and the single-scale one achieve almost 100% accuracy and 0% loss after large enough training epochs, we select the three-scale ResNet with AMSGrad since it converges the most smoothly and quickly. By repeating the above comparison, our later experiments choose the three-scale ResNet with AMSGrad optimizer, Leaky-ReLU activation function, and the three-channel input. The inputs are formed by the spread and the two stock return processes unless stated otherwise in our later experiments.

We divide the data into the training and the validation set to determine the number of training epochs. We first train the model on the training set data and then run the resulting model on the validation data set to calculate the accuracy and loss. To retrieve useful information from the training data

[9] We adopt the structure of ResNet and the convolution kernel sizes 3, 5, and 7 that were proposed by [41].

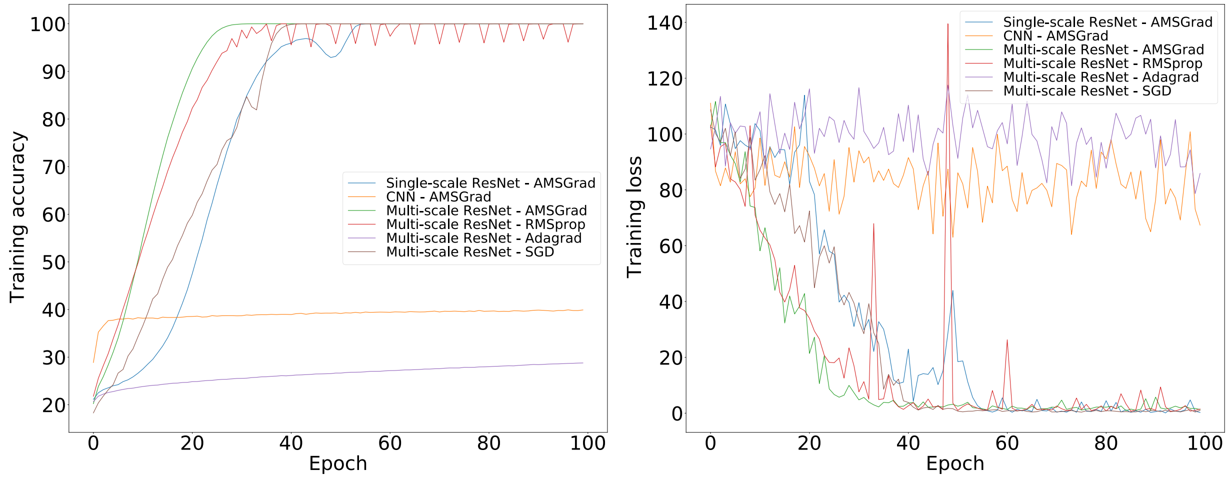


FIGURE 6. **Training accuracies and losses of CNN and ResNet under different settings. We illustrate part of our experiments (denoted by the legends) for finding the optimized settings to achieve the best training results. The activation function used here is Leaky-ReLU. The input has three channels: the spread and the two stock return processes.**

set without incurring overfitting, we halt the training process when the win rate of the validation set reaches a maximum.

We illustrate part of our experiments (denoted by the legends) for finding the optimized settings to achieve the best training results. The activation function used here is Leaky-ReLU. The input has three channels: the spread and the two stock return processes.

B. TWO-STAGE MODEL

To improve the win rate and reduce trading risk further, we produce a two-stage model. The first-stage model recommends a proper representative threshold for each stock pair, as described in the previous sections; the second stage then predicts and prevents unprofitable pairs from trading. The training of the second-stage model is illustrated in Figure 7. Here, we divide the training set data into set 1 to train the first-stage mechanism and set 2 for the second-stage mechanism. In the first stage, the trigger threshold for each pair is determined by the optimal threshold selection from Equation (5), then processed by the representative labeling mechanism discussed in Section III-D. The high training accuracy in Figure 6 indicates that the profit of almost every pair from training set 1, given the trigger threshold recommended by the first-stage model, is positive, an outcome that is of no use if we seek to distinguish unprofitable pairs from profitable ones. We address this issue by using the first-stage mechanism to predict a trigger threshold y for each pair x in training set 2, as in Figure 7. Next, we apply the PTS with the open and stop-loss thresholds y on x to generate the win or loss label, after which we use the features of x, according to the spread and stock price processes, as the input; the win/loss label functions as the ground truth to train the second-stage mechanism in the three-scale ResNet. The trigger threshold recommendation model with RLM, developed in the first stage, and the unprofitability detection mechanism trained in the second stage together execute the PTS on the testing data, as illustrated in Figure 8. Thus, for each stock pair in the test data set, a proper representative threshold gets identified and recommended by the first-stage model, and the pair together with recommended thresholds are examined to determine whether they are profitable or not by the second-stage model. In contrast with the single-stage model, the two-stage model trades only those pairs that are predicted to be profitable. Our experiments show that this design improves the win rate and reduces risk.

V. EMPIRICAL TESTS

We conduct experiments on the constituent stocks of the Taiwan Top 50 ETF (0050.TW) from 2013 to 2018 to back-test improvements in PTS investment performance due to the proposed RLM and the two-stage model. To evaluate the trading performance, we first extract intra-day trading information from each trading day D_1 from the testing period, as illustrated in Figure 3. Then we retrieve stock pairs feasible for PTS by applying the Johansen cointegration test to the formation period data of day D_1. Next, we predict each pair's optimal representative trigger threshold using the trained three-scale ResNet described in Section IV-A. With the retrieved stock pair and the corresponding trigger threshold, we execute tick-by-tick pairs trading in the $D_1's$ trading period. We execute all trades one tick later than the spread process hits the trigger threshold to simulate price slippage effects.

We compare the investment performance of different PTS by analyzing the following financial indicators: the (overall) profit, the win rate, the normal close rate, the number of trades, the Sharpe ratio (SR) calculated on a daily or pair basis, the maximum drawdown (MDD), the maximum required capital, and the average profit (per trade), as listed in the first column of Table 1. To facilitate the performance comparison in the following tables, we set in boldface the best performance for each indicator (except for the number of trades and the maximum required capital) to easily identify the best methods or settings. The (overall) profit sums

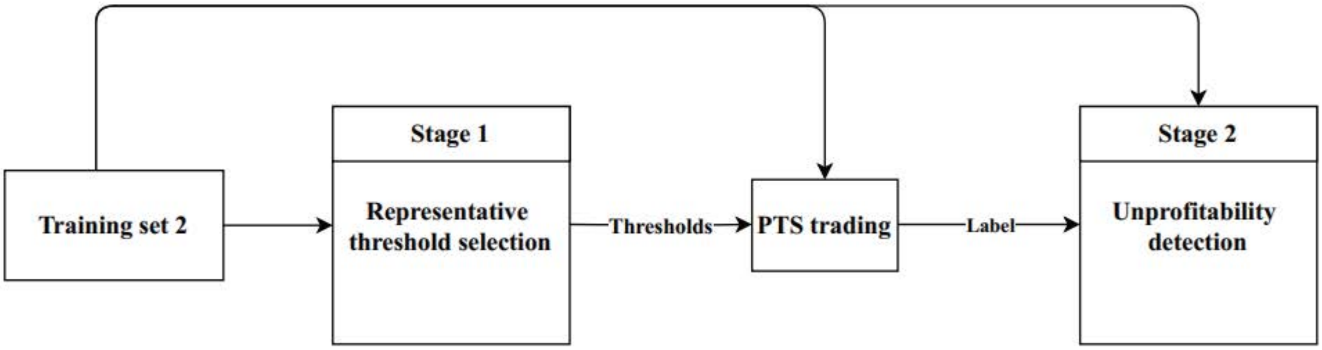


FIGURE 7. Training the second-stage mechanism.

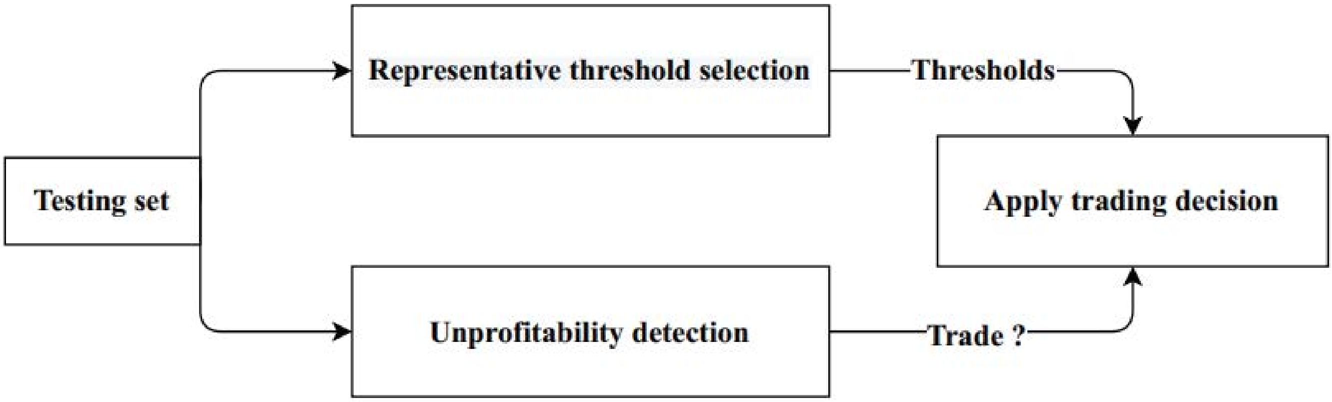


FIGURE 8. Applying the two-stage model for trading.

up the profit or loss for all trading days during the testing period, where the profit of day D is the lump sum profits earned by trading all PTS-eligible stock pairs on that day. The profit for trading each PTS-eligible stock pair is calculated in Equation (3). We measure the required capital for day D as the sum of the capital required to execute each PTS on that day. The maximum required capital is defined as the maximum required capital for each trading day in the testing period. The daily (pair) return is calculated as the daily (pair) profit divided by the required capital for that day (trade). The Sharpe ratio, which estimates the average excess trading return divided by the corresponding risk, can be estimated on either a daily basis, as

$$\frac{\text{Average daily return} - \text{Risk-free rate}}{\text{Standard deviation of daily return}}, \tag{7}$$

or else a pair basis, as

$$\frac{\text{Average pair return} - \text{Risk-free rate}}{\text{Standard deviation of pair return}}.$$

The maximum drawdown (MDD) is defined as the maximum cumulative daily loss during the testing period. The win rate is defined as the number of profit-making trades divided by the total number of trades during the testing period. The normal close rate reflects the number of trades whose spread process converges to the mean level[10] divided by the total number of trades.

Section V-A compares various representative labeling methods discussed in Sections IV-A and III-D. Because combining multi-scale ResNet with the settings described in Figure 6 and HighFreq (or KMeans(0)) yields the best performance, these settings are adopted in our subsequent experiments. Section V-B demonstrates that the proposed RLM outperforms existing trigger threshold selection mechanisms. Section V-C shows that training a machine learning model with the spread process defined in Equation (2), whose patterns are time-invariant, prevents changes in financial markets from harming the model's predictability. Finally, Section V-D illustrates how the two-stage model developed in Section IV-B can effectively reduce PTS risk than existing methods do.

A. SELECTION OF RLMs

To improve PTS investment performance, we select the best DL and corresponding settings as in Section IV-A and proper RLM in this section to ensure the efficiency of the training described in step 5 of Figure 3. Table 1 compares the different RLM proposed in Section III-D. In row 4, KMeans(0),

[10] That is, the spread process should converges to the red line like nodes F and D, as illustrated in Figure 2.

KMeans(1), and KMeans(2) denote representative label settings that apply k-means to the total trigger thresholds (see Figure 5(a)), trigger thresholds with probabilities greater than 0.1% (see Figure 5(b)), and trigger thresholds with probabilities greater than 0.5% (see Figure 5(c)), respectively. HighFreq picks the trigger thresholds with the top-25 highest probabilities, as illustrated in Figure 5(d).

Both the win rate and the normal close rate are high for these label mechanisms because the spread processes selected by the cointegration test described in Section II-B likely have mean-reverting properties. Therefore, a mechanism with larger total opening numbers likely yields higher profits and Sharpe ratios. KMeans(0) has the highest number of trades among the four mechanisms probably because it does not exclude information from other trigger thresholds with lower probabilities. However, unlike representative thresholds produced by k-means, which typically do not coincide with trigger thresholds due to the average calculation, every threshold recommended by HighFreq is directly a trigger threshold with the highest 25 probabilities. Without disturbances in the open and stop-loss thresholds, HighFreq produces better pair-based investment results (i.e., SR (pair) and average profit (per trade)) than the other k-mean-based mechanisms. Because KMeans(0) and HighFreq possess distinct advantages, as illustrated by the best result set in boldface, we include either KMeans(0) or HighFreq for comparisons in our later experiments.

B. COMPARISONS WITH THRESHOLD SELECTION METHODS

Table 2 compares existing trigger threshold selection mechanisms with our RLM (i.e., method 2). Fallahpour *et al.* [4] (method 1) reduce the threshold selection problem to a multi-armed bandit problem and solve it using a reinforcement learning model with 39 actions (i.e., trigger thresholds), extracted from a much narrower set $O \in \{0.5, 1, \cdots 3\}$ and $S \in \{0.5, 1, \cdots 5\}$. Kim and Kim [5] (method 4) use deep reinforcement learning (DRL) to select one of six heuristically generated actions for trading. Because the number of actions (threshold choices) is relatively small, these models do not suffer from the training non-convergence problem described in Section III-C. However, limiting the number of actions (or the choices of trigger thresholds) likely deteriorates the PTS investment performance.

For a fair comparison with the six actions of the DRL method proposed by [5], we add the HighFreq mechanism with six representative trigger thresholds (method 3). We find that HighFreq outperforms DRL in almost every aspect, even though DRL recommends more trading opportunities (i.e., more number of trades). However, the low win rate for DRL results in lower overall profits and SR metrics than HighFreq with six representative trigger thresholds. Increasing the number of representative trigger thresholds from 6 to 25 increases the number of trades and win rate, improves profit and SR, and reduces risk (reflected by MDD). The best performance terms set in boldface also suggest that the proposed methods 2 and 3 generally outperform other methods. The quarterly pair-based Sharpe ratios illustrated in the upper panel in Figure 9 also show that a well-designed PTS yields absolute positive returns regardless of changes in the market trend shown in the lower panel. Methods 2 and 3 also outperform other methods irrespective of stock market changes. Besides, the naive reinforcement learning model proposed by Fallahpour *et al.* [4] performs poorly, with a win rate below 50% and negative profits. However, if transaction costs are ignored, as mentioned in their papers, the profits of their model become positive. This implies that their model fails to find a proper solution to filter out unprofitable trades due to transaction costs.

C. TRAINING WITH TIME-(IN) VARIANT DATA

Market trends vary with time, and the non-stable nature of a stock price/return process makes it difficult for machine learning models to capture and predict stable patterns of trading data [10]. Thus model performances vary significantly, depending on whether there are turning points during the training or testing periods. Therefore, a proper hyperparameter setting that controls the length of training and testing periods could be challenging to identify [33]. However, spread processes (Equation (2)) generated by the cointegration test are stationary; their statistical properties do not change when shifted in time. This valuable property allows us to extend the training period to capture more patterns to improve PTS profitability, without exposing the model to changes in financial markets. In Table 3, lengthened training periods generally coincide with increases in win rate, SR, trading opportunities (i.e., number of trades), and overall profit. In contrast, using only non-stationary series, such as stock prices and stock returns, as training data yields unstable performance with each increment of the training period. The spread process also can be expressed as the non-invertible function of the pair of stocks' prices, as in Equation (2). These two stock price processes thus contain broader information than is available in the spread process. Even if the spread process contains less information though, its stationary property makes it easier for machine learning algorithms to capture time-invariant patterns, rather than the time-variant patterns of the stock return and price processes.

Next, we proceed to analyze the impacts of combining different types of processes as inputs. Combining the stationary process with the non-stationary one as inputs (i.e., "S + R" and "S + P" cases) provides stable investment performance that grows with the increment of the training period. In addition, their performances are better than those generated by training with just the spread process (i.e., "S" case). But investment performance generated by training with non-stationary return and stock price processes (i.e., "R + P") is unstable. This result again confirms the value of the time-invariant property.

To strengthen our arguments, we extend the experiment in Table 3 to different testing periods across 2016–2018, as shown in Table 4. We observe the same phenomena.

TABLE 1. Comparing different representative labeling mechanisms. We list the training period, validation period, and testing period in the first, second, and third rows. The fourth row lists RLMs to generate representative trigger thresholds. The investment performance indicators used to measure performance are in the first column. For each indicator, we set in boldface the best of the four RLMs.

Training period	Jan. 2015 – Oct. 2016							
Validation period	Nov. 2016 – Dec. 2016							
Testing period	2017				2018			
Method	`KMeans(0)`	`KMeans(1)`	`KMeans(2)`	`HighFreq`	`KMeans(0)`	`KMeans(1)`	`KMeans(2)`	`HighFreq`
Profit (thousand)	2109.40	2057.28	1922.17	**2233.66**	2021.37	1803.69	1710.59	**2255.96**
Win rate (%)	76	76	76	**77**	74	74	75	**76**
Normal close rate (%)	75	75	76	**78**	76	75	76	**77**
Number of trades	12196	11377	11434	11901	13772	12801	13029	13357
SR (daily based)	**7.4776**	7.1598	7.2011	7.4389	1.8921	1.5952	1.6411	**2.1582**
SR (pair based)	0.1829	0.1916	0.1952	**0.2189**	0.1332	0.1224	0.1283	**0.1513**
MDD	**71**	91	84	75	418	386	364	**300**
Required capital (thousand)	49311	43551	48496	50784	64391	74760	74251	62389
Average profit (per trade) (thousand)	0.1729	0.1810	0.1677	**0.1877**	0.1467	0.1408	0.1312	**0.1699**

TABLE 2. Comparisons among different trigger threshold selection methods. The performance indicators (first column) for different trigger threshold settings proposed by Fallahpour *et al.* [4] with 39 actions (method 1), our HighFreq with 25 representative thresholds (method 2), our HighFreq with 6 representative thresholds (method 3), and the deep reinforcement method proposed by Kim and Kim [5] with 6 heuristic actions (method 4) are listed for comparison. For each indicator, we set in boldface the best of the four methods.

Training period	Jan. 2014 – Oct. 2015				Jan. 2015 – Oct. 2016				Jan. 2016 – Oct. 2017			
Validation period	Nov. 2015 – Dec. 2015.				Nov. 2016 – Dec. 2016				Nov. 2017 – Dec. 2017			
Testing period	2016				2017				2018			
Method	1	2	3	4	1	2	3	4	1	2	3	4
Profit (thousand)	-900.19	**2270.83**	1945.01	727.99	-1469.52	**2233.66**	1804.99	1285.98	-1590.59	**2289.93**	1758.83	102.45
Win rate (%)	64	**80**	79	68	56	**77**	76	67	47	**76**	75	65
Normal close rate (%)	71	**81**	78	67	62	**77**	**77**	67	44	77	**78**	68
Number of trades	42488	16777	13734	18628	34566	11902	11385	15440	13577	13151	12093	16388
SR (daily based)	-2.38	**8.1791**	7.5344	2.5716	-4.74	**7.4383**	6.4398	4.2653	-1.73	**2.2289**	1.8119	0.1115
SR (pair based)	-0.01	**0.2416**	0.2341	0.1099	-0.06	**0.2190**	0.1867	0.1113	-0.04	**0.1426**	0.1337	0.0689
MDD	208	67	**49**	102	366	75	53	**51**	422	**343**	504	381
Required capital (thousand)	151443	50543	44147	57155	96012	50784	47309	62088	61672	80292	68094	86349
Average profit (per trade) (thousand)	-0.0212	0.1354	**0.1416**	0.0390	-0.0466	**0.1877**	0.1594	0.0832	-0.1172	**0.1743**	0.1454	0.0062

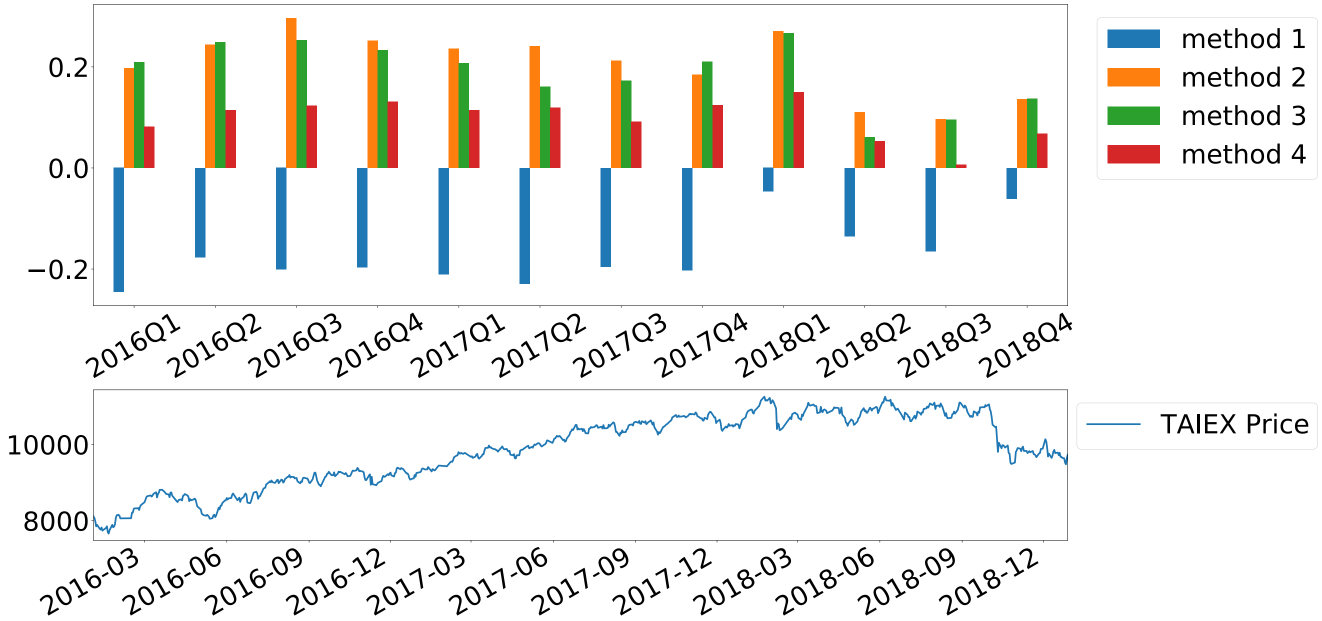


FIGURE 9. Comparisons with quarterly pair-based sharpe ratios among different methods. The upper panel illustrates the quarterly pair-based Sharpe ratios for executing PTSs with the trigger threshold selection methods mentioned in Table 2. The lower panel shows the price trend of the Taiwan Capitalization Weighted Stock Index (TAIEX) for the corresponding period.

The investment performance generated by training with both time-invariant and variant data (i.e., "S + R" or "S + P" cases) is better than that generated by training with time-invariant data (i.e., "S" case), which in turn is better than those generated by training with time-variant data (i.e., "R" or "P" cases). Moreover, the distribution of the stock return process is more stable than that of the stock price, as the former process is evaluated by applying the difference operators on the logarithm of the latter process. Specifically, the return for stock S_j^i over the period $[\tau, \tau']$ can be evaluated by $\ln \frac{S_j^i(\tau')}{S_j^i(\tau)}$, as in Equation (3). Thus, the machine learning

TABLE 3. Impact of different lengths of training periods and inputs on PTS performance. The input data can be the spread process (S), prices (P), and/or return (R) processes of stocks. For example, "R + P" indicates that the input data are composed of the stocks' returns and price processes. The validation period is Oct. 2016-Dec. 2016, and the testing period is 2017. The length of the training period ranges from 0.25 to 1.75 years. For example, the training period Jan. 2015-Sep. 2016 pertains to the 1.75-year case.

Testing period	2017											
Input data	S				R				P			
Training period	1.75	1.25	0.75	0.25	1.75	1.25	0.75	0.25	1.75	1.25	0.75	0.25
Profit (thousand)	1937.08	1802.65	1646.85	1391.34	1842.20	806.35	1973.53	1197.34	513.02	1647.67	1881.95	1148.43
Win rate (%)	76	75	76	74	74	68	75	73	70	74	75	75
Normal close rate (%)	76	74	76	74	73	64	74	70	65	73	75	75
Number of trades	10692	10509	9089	8172	10866	8466	11187	8321	4876	10669	11663	7052
SR (daily based)	7.4237	6.5282	5.9178	6.4087	5.9951	3.0337	7.0880	4.7318	2.9517	6.1777	6.4918	5.9240
SR (pair based)	0.2000	0.1815	0.1901	0.1836	0.1724	0.1026	0.1704	0.1635	0.1100	0.1638	0.1839	0.1766
MDD	62	62	83	51	68	105	47	63	53	87	59	59
Required capital (thousand)	52089	39906	38086	37292	42703	42661	50222	33841	20467	49372	45869	44004
Average profit (per trade) (thousand)	0.1812	0.1715	0.1811	0.1702	0.1695	0.0952	0.1764	0.1439	0.1052	0.1544	0.1613	0.1629
Input data	R + P				S + R				S + P			
Training period	1.75	1.25	0.75	0.25	1.75	1.25	0.75	0.25	1.75	1.25	0.75	0.25
Profit (thousand)	1998.79	647.22	1463.47	1235.13	2744.18	2137.75	1783.44	1128.70	2232.66	2118.94	1646.85	1437.55
Win rate (%)	75	76	75	77	78	77	76	76	77	76	76	77
Normal close rate (%)	75	75	76	77	80	76	76	76	76	76	76	76
Number of trades	11266	2775	7833	5969	13670	11181	10415	6450	11900	11392	10344	7237
SR (daily based)	6.7917	5.5990	6.3876	6.7940	9.1746	7.7868	6.3554	5.5612	7.4381	7.6869	6.0328	6.9709
SR (pair based)	0.1992	0.2126	0.1776	0.2246	0.2306	0.2130	0.1843	0.1723	0.2188	0.2137	0.1897	0.2003
MDD	77	41	55	47	55	73	78	66	76	60	58	51
Required capital (thousand)	45305	15150	33827	34843	56223	49807	38972	28256	50783	44278	38528	37720
Average profit (per trade) (thousand)	0.1774	0.2332	0.1868	0.2069	0.2007	0.1911	0.1712	0.1749	0.1876	0.1860	0.1592	0.1986

TABLE 4. Comprehensive examination of different inputs for PTS performance. The training, validation, and testing periods are listed in the first three rows. The PTS performances of HighFreq with different input data are compared for the period from 2016 to 2018. For each indicator, we set in boldface the best of the different input data sources.

Training period	Jan. 2014 – Oct. 2015			Jan. 2015 – Oct. 2016			Jan. 2016 – Oct. 2017		
Validation period	Nov. 2015 – Dec. 2015			Nov. 2016 – Dec. 2016			Nov. 2017 – Dec. 2017		
Testing period	2016			2017			2018		
Input data	S	R	P	S	R	P	S	R	P
Profit (thousand)	**1944.31**	1905.96	1557.94	**2207.08**	2062.02	2007.25	**1870.16**	1842.02	1688.83
Win rate (%)	**77**	76	76	**76**	75	75	**74**	**74**	73
Normal close rate (%)	**76**	75	75	**76**	75	75	**77**	76	76
Number of trades	14433	14766	11689	11719	12831	12165	11728	12463	12736
SR (daily based)	**6.0314**	6.0143	5.9901	**7.7379**	6.7916	6.3686	**1.9245**	1.9128	1.6942
SR (pair based)	**0.2394**	0.2315	0.2115	**0.2025**	0.1705	0.1879	**0.1385**	0.1365	0.1336
MDD	100	73	**57**	83	**72**	90	241	**194**	224
Required capital (thousand)	39835.172	42172.367	46428.715	43073.422	51461.329	53587.048	57366.327	60658.698	68463.454
Average profit (per trade) (thousand)	**0.1347**	0.1291	0.1332	**0.1883**	0.1607	0.1650	**0.1595**	0.1478	0.1326
Input data	R + P	S + R	S + P	R + P	S + R	S + P	R + P	S + R	S + P
Profit (thousand)	1761.82	**2269.83**	2229.59	1998.79	2232.66	**2744.18**	1702.85	**2289.50**	2204.04
Win rate (%)	76	**79**	77	75	77	**78**	73	**75**	**75**
Normal close rate (%)	76	**81**	77	75	76	**80**	76	**77**	**77**
Number of trades	14715	16776	15656	11266	11900	13670	12211	13149	13268
SR (daily based)	5.9387	**8.1789**	7.6765	6.7919	7.4381	**9.1746**	1.7937	**2.2287**	2.0995
SR (pair based)	0.2176	0.2415	**0.2426**	0.1992	0.2188	**0.2306**	0.1324	0.1422	**0.1423**
MDD	80	**66**	69	77	76	**55**	373	344	**210**
Required capital (thousand)	46232.812	50542.895	55244.04	45305.439	50783.951	56223.750	70309.697	80291.094	62750.708
Average profit (per trade) (thousand)	0.1197	0.1353	**0.1424**	0.1774	0.1876	**0.2007**	0.1394	**0.1741**	0.1661

algorithm better captures patterns in the stock return process, yielding PTS performance that is generally better than that of stock prices. Observing the best performances set in boldface shows that training with a time-invariant spread process (S) is better than training with time-varying stock returns (R) or prices (P) in the upper panel of Table 4. Training with data containing time-invariant processes (i.e., S + R or S + P) is better than training with only time-varying data (R + P) in the lower panel.

D. TWO-STAGE LEARNING MODEL

It is hard to prevent unprofitable stock pairs from trading, no matter the level of sophistication achieved by the pair and trigger threshold selection methods (e.g., cointegration test and HighFreq adopted herein). To reduce PTS risk (or loss) by avoiding trading unprofitable pairs, Sarmento and Horta [8] use OPTICS to group stocks by their 5-minute moving average returns; they prevent trading a stock pair if the stocks of the pair belong to different groups. We combine their OPTICS-based risk-reduction into our first-stage RLM mechanism, illustrated in Figure 3, and thereby compare their risk-reduction model and our second-stage model that detects and removes unprofitable trades, as in Figure 7. Comparisons of the one-stage model (i.e., adopting only the RLM mechanism for trading) and the combinations of RLM with different risk-reduction methods are listed in Table 5.

Detecting and removing unprofitable pairs may erroneously remove profitable transactions, which would reduce overall profits and the daily Sharpe ratio. But our removal mechanism also improves the win/normal close rate and significantly enhances pair-based SR and the average profit (per trade) by up to 40%. In addition, MDD falls significantly, attesting to the effectiveness of our second-stage approach to protect investors from unexpected significant loss. The OPTICS-based approach [8] achieves similar effects, but our proposed two-stage model outperforms their model on almost

TABLE 5. Comparison with the optics-based risk-reduction algorithm. The training data set 1 and validation data used to train the first-stage model in Figure 3 are listed in the first two rows. The training data set 2 used to train our or Sarmento and Horta [8] risk-reduction methods is listed in the third row. "O-S" and "T-S" denote our proposed one-stage and two-stage models, respectively. "5 min-M" indicates the OPTICS grouping based on the 5-minute moving average returns. The performance when we use Kmeans(0) or HighFreq to select representative thresholds appears in subsequent rows. For each indicator, we set in boldface the best of the three methods.

Training dataset 1	Jan. 2013 – Oct. 2014			Jan. 2014 – Oct. 2015			Jan. 2015 – Oct. 2016		
Validation period	Oct. 2014 – Dec. 2014			Oct. 2015 – Dec. 2015			Oct. 2016 – Dec. 2016		
Training dataset 2	Jan. 2015 – Dec. 2015			Jan. 2016 – Dec. 2016			Jan. 2017 – Dec. 2017		
Testing period	2016			2017			2018		
Model	O-S	T-S	5 min-M	O-S	T-S	5 min-M	O-S	T-S	5 min-M
Threshold: Kmeans(0)									
Profit (thousand)	**1937.45**	441.06	461.77	**1918.16**	454.04	313.77	**2020.37**	406.24	62.55
Win rate (%)	77	**78**	77	76	**77**	76	74	**76**	74
Normal close rate (%)	76	**77**	75	75	**76**	**76**	76	**78**	75
Number of trades	14577	2561	3199	11579	2270	2027	13771	1947	1250
SR (daily based)	**6.3641**	5.6972	3.7878	**6.3954**	4.7917	4.0301	**1.8919**	1.8208	0.6060
SR (pair based)	0.2232	**0.2598**	0.2385	0.1823	0.2126	**0.2400**	0.1322	**0.1712**	0.1621
MDD	90	**26**	63	71	32	**31**	419	105	**80**
Required capital (thousand)	45910	15520	24597	47465	11832	30350	64390	15353	24770
Average profit (per trade) (thousand)	0.1329	**0.1803**	0.1379	0.1656	**0.2000**	0.1578	0.1467	**0.2093**	0.054
Threshold: HighFreq									
Profit (thousand)	**2533.94**	724.73	564.80	**2543.85**	720.65	337.87	**2254.96**	442.95	89.87
Win rate (%)	**80**	**80**	**80**	79	**80**	**80**	75	**77**	**77**
Normal close rate (%)	81	**83**	81	80	**82**	**82**	77	**81**	80
Number of trades	16820	3986	3754	12418	3055	2063	13355	2179	1369
SR (daily based)	**7.3559**	5.2082	4.8660	**8.4031**	7.4118	4.2987	**2.1572**	2.0918	0.7574
SR (pair based)	0.2668	**0.3117**	0.2883	0.2249	0.2637	**0.2972**	0.1512	**0.1784**	0.1779
MDD	97	43	**34**	66	**18**	29	400	107	**102**
Required capital (thousand)	58351	23010	31681	47130	15816	31519	62389	15853	24562
Average profit (per trade) (thousand)	0.1507	**0.1818**	0.1505	0.2048	**0.2358**	0.1638	0.1688	**0.2033**	0.066

TABLE 6. Comparison with Lu *et al.* [9]. All experimental results (except the last row) are retrieved from Table 4 in Lu *et al.* [9]. The experimental settings also match their paper. For each financial indicator, we set in boldface the best of the methods listed in the first column.

	Sharpe ratio	Sortino ratio	MDD
SAPT	**4.30**	13.18	**0.020**
SAPT w/o Break	3.45	9.53	0.044
SAPT w/o Time	3.42	9.78	0.043
SAPT w/o Hold	3.07	6.71	0.090
PTDQN	1.01	1.41	0.169
SAPT-3-std	1.07	1.77	0.143
SAPT-ADF	-0.23	-0.32	0.127
SAPT-BCD	-3.15	-2.95	0.250
SAPT-LSTM	-1.32	-1.49	0.297
HighFreq	2.40	**28.89**	0.041

all financial indicators, according to the direct comparisons. Observing the best performances set in boldface shows that our first-stage model affords more trading opportunities and aggregated profits. However, our two-stage model has better average profitability and lower risk per trade.

Lu *et al.* [9] design a "structural break aware pair trading" strategy (SAPT) that stops losses by detecting "structural breaks", or the loss of cointegrating properties, as illustrated in the first column of Table 6. In their two-phase framework, DL first serves to detect the probability of a structural break. Then second phase trains a deep reinforcement learning model to select heuristically generated thresholds. We extend their experiment by inserting our two stage model with 25 representative thresholds selected by HighFreq, in the last row of the table. Although the Sharpe ratio of HighFreq performs worse than the first four versions of SAPT, it significantly outperforms all different versions of SAPT in terms of the Sortino ratio. This is because the denominator of the Sharpe ratio (Equation (7)) is the standard deviation of all returns, regardless of positive or negative signs. In contrast, the Sortino ratio is the standard deviation of negative returns. Therefore, we can deduce that the risk of negative returns in HighFreq is much smaller than in SAPT.

VI. CONCLUSION

This paper proposes a novel two-stage model to improve PTS investment performance and reduce trading risk by optimally selecting trigger thresholds and removing unprofitable stock pairs. In the first stage, we train a multi-scale ResNet with the proposed RLM to select optimal thresholds without incurring the non-convergence training problem. Our approach outperforms other approaches that heuristically generate a set of thresholds for selections. To remove unprofitable stock pairs in the second stage, we train another multi-scale ResNet with the profitability of each stock pair obtained by executing the PTS with trigger thresholds recommended in the first stage. Therefore, our second-stage model outperforms models that indirectly predict stock pair profitability by the similarity of stock price processes [8] and the occurrence of structural breaks [9]. We also find that the time invariance of the spread process (i.e., portfolio value process) makes it easier for machine-learning algorithms to capture features and hence improve investment performance. Indeed, as in much of the financial-market-prediction literature, training with time-varying patterns such as stock prices or returns yields unstable investment performance as the patterns learned from the training set may change in the testing set. Thus, changing the training period length influences investment performance in an unstable way due to unpredictable changes in financial markets. However, training with the time-invariant spread process monotonically improves the performance with prolonged training periods and outperforms training with time-variant stock prices and returns, even though the spread process contains less information than the stock price or return data.

Our work yields meaningful insights for further developments in financial market prediction and investment. To avoid unpredictable changes in financial markets due to outbreaks of pandemics and wars from harming the predictability of machine learning models for PTS, we can train with time-invariant spread processes to eliminate unstable performance. The high profitability and the low-risk properties also render our two-stage model a good PTS candidate for applications such as reducing the variance (or risk) in investment portfolios [16] and establishing optimal asset allocations [17].

REFERENCES

[1] G. Vidyamurthy, *Pairs Trading: Quantitative Methods and Analysis*, vol. 217. Hoboken, NJ, USA: Wiley, 2004.

[2] C. Krauss and J. Stübinger, "Non-linear dependence modelling with bivariate copulas: Statistical arbitrage pairs trading on the S&P 100," *Appl. Econ.*, vol. 49, no. 52, pp. 5352–5369, Nov. 2017.

[3] S. Johansen, *Likelihood-Based Inference in Cointegrated Vector Autoregressive Models*. Oxford, U.K.: Oxford Univ. Press, 1995.

[4] S. Fallahpour, H. Hakimian, K. Taheri, and E. Ramezanifar, "Pairs trading strategy optimization using the reinforcement learning method: A cointegration approach," *Soft Comput.*, vol. 20, no. 12, pp. 5051–5066, Dec. 2016.

[5] T. Kim and H. Y. Kim, "Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries," *Complexity*, vol. 2019, pp. 1–20, Nov. 2019.

[6] W.-L. Kuo, T.-S. Dai, and W.-C. Chang, "Solving unconverged learning of pairs trading strategies with representation labeling mechanism," in *Proc. CIKM Workshops*, vol. 3052, 2021, pp. 1–11.

[7] J. Li, F. Fang, K. Mei, and G. Zhang, "Multi-scale residual network for image super-resolution," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 517–532.

[8] S. M. Sarmento and N. Horta, "Enhancing a pairs trading strategy with the application of machine learning," *Expert Syst. Appl.*, vol. 158, Nov. 2020, Art. no. 113490.

[9] J.-Y. Lu, H.-C. Lai, W.-Y. Shih, Y.-F. Chen, S.-H. Huang, H.-H. Chang, J.-Z. Wang, J.-L. Huang, and T.-S. Dai, "Structural break-aware pairs trading strategy using deep reinforcement learning," *J. Supercomput.*, vol. 78, no. 3, pp. 3843–3882, Feb. 2022.

[10] R. Singh and S. Srivastava, "Stock prediction using deep learning," *Multimedia Tools Appl.*, vol. 76, no. 18, pp. 18569–18584, 2017.

[11] C. Krauss, "Statistical arbitrage pairs trading strategies: Review and outlook," *J. Econ. Surv.*, vol. 31, no. 2, pp. 513–545, Apr. 2017.

[12] H. Rad, R. K. Y. Low, and R. Faff, "The profitability of pairs trading strategies: Distance, cointegration and copula methods," *Quant. Finance*, vol. 16, no. 10, pp. 1541–1558, Oct. 2016.

[13] N. Huck and K. Afawubo, "Pairs trading and selection methods: Is cointegration superior?" *Appl. Econ.*, vol. 47, no. 6, pp. 599–613, Feb. 2015.

[14] R. F. Engle and C. W. J. Granger, "Co-integration and error correction: Representation, estimation, and testing," *Econometrica*, vol. 55, no. 2, pp. 251–276, Mar. 1987.

[15] S. Johansen, "Statistical analysis of cointegration vectors," *J. Econ. Dyn. Control*, vol. 12, nos. 2–3, pp. 231–254, Jun. 1988.

[16] J. Giner, "Orthant-based variance decomposition in investment portfolios," *Eur. J. Oper. Res.*, vol. 291, no. 2, pp. 497–511, Jun. 2021.

[17] S. C. P. Yam, H. Yang, and F. L. Yuen, "Optimal asset allocation: Risk and information uncertainty," *Eur. J. Oper. Res.*, vol. 251, no. 2, pp. 554–561, Jun. 2016.

[18] M. Fil and L. Kristoufek, "Pairs trading in cryptocurrency markets," *IEEE Access*, vol. 8, pp. 172644–172651, 2020.

[19] P. S. Lintilhac and A. Tourin, "Model-based pairs trading in the bitcoin markets," *Quant. Finance*, vol. 17, no. 5, pp. 703–716, May 2017.

[20] N. Huck, "Pairs trading and outranking: The multi-step-ahead forecasting case," *Eur. J. Oper. Res.*, vol. 207, no. 3, pp. 1702–1716, Dec. 2010.

[21] A. Brim, "Deep reinforcement learning pairs trading with a double deep Q-network," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 0222–0227.

[22] F. Xu and S. Tan, "Dynamic portfolio management based on pair trading and deep reinforcement learning," in *Proc. 3rd Int. Conf. Comput. Intell. Intell. Syst.*, Nov. 2020, pp. 50–55.

[23] T.-W. Hsu, C.-C. Chen, H.-H. Huang, M. C. Chen, and H.-H. Chen, "Hedging via opinion-based pair trading strategy," in *Proc. Companion Web Conf.*, 2020, pp. 69–70.

[24] C. Huang, G. Min, Y. Wu, Y. Ying, K. Pei, and Z. Xiang, "Time series anomaly detection for trustworthy services in cloud computing systems," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 60–72, Feb. 2022.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jan. 2016, pp. 770–778.

[26] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, "Stock price prediction using LSTM, RNN and CNN-sliding window model," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, 2017, pp. 1643–1647.

[27] M. S. Hegde, G. Krishna, and R. Srinath, "An ensemble stock predictor and recommender system," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2018, pp. 1981–1985.

[28] K. Khare, O. Darekar, P. Gupta, and V. Z. Attar, "Short term stock price prediction using deep learning," in *Proc. 2nd IEEE Int. Conf. Recent Trends Electron., Inf. Commun. Technol. (RTEICT)*, May 2017, pp. 482–486.

[29] S. Jain, R. Gupta, and A. A. Moghe, "Stock price prediction on daily stock data using deep neural networks," in *Proc. Int. Conf. Adv. Comput. Telecommun. (ICACAT)*, Dec. 2018, pp. 1–13.

[30] C. Peng, Z. Yin, X. Wei, and A. Zhu, "Stock price prediction based on recurrent neural network with long short-term memory units," in *Proc. Int. Conf. Eng., Sci., Ind. Appl. (ICESI)*, 2019, pp. 1–5.

[31] M. Wen, P. Li, L. Zhang, and Y. Chen, "Stock market trend prediction using high-order information of time series," *IEEE Access*, vol. 7, pp. 28299–28308, 2019.

[32] Y. Gu, D. Yan, S. Yan, and Z. Jiang, "Price forecast with high-frequency finance data: An autoregressive recurrent neural network model with technical indicators," in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 2485–2492.

[33] L. Zhang, C. Aggarwal, and G.-J. Qi, "Stock price prediction via discovering multi-frequency trading patterns," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 2141–2149.

[34] S. Johansen, "Identifying restrictions of linear equations with applications to simultaneous equations and cointegration," *J. Econometrics*, vol. 69, no. 1, pp. 111–132, Sep. 1995.

[35] J. Rudy, C. Dunis, G. Giorgioni, and J. Laws, "Statistical arbitrage and high-frequency data with an application to Eurostoxx 50 equities," Social Sci. Res. Netw., 2010.

[36] S. Broumandi and T. Reuber, "Statistical arbitrage and FX exposure with south American ADRs listed on the NYSE," *Financial Assets Investing*, vol. 3, no. 2, pp. 5–18, May 2012.

[37] H. Lütkepohl, P. Saikkonen, and C. Trenkler, "Maximum eigenvalue versus trace tests for the cointegrating rank of a VAR process," *Econometrics J.*, vol. 4, no. 2, pp. 287–310, 2001.

[38] S.-H. Kim, D.-Y. Park, and K.-H. Lee, "Hybrid deep reinforcement learning for pairs trading," *Appl. Sci.*, vol. 12, no. 3, p. 944, Jan. 2022.

[39] S. Endres and J. Stübinger, "A flexible regime switching model with pairs trading application to the S&P 500 high-frequency stock returns," *Quant. Finance*, vol. 19, no. 10, pp. 1727–1740, Oct. 2019.

[40] B. Liu, L.-B. Chang, and H. Geman, "Intraday pairs trading strategies on high frequency data: The case of oil companies," *Quant. Finance*, vol. 17, no. 1, pp. 87–100, Jan. 2017.

[41] R. Liu, F. Wang, B. Yang, and S. J. Qin, "Multiscale kernel based residual convolutional neural network for motor fault diagnosis under nonstationary conditions," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 3797–3806, Jun. 2019.

[42] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

WEI-LUN KUO received the B.S. degree in computer science from the National Taiwan Normal University, Taipei, Taiwan, in 2019, and the M.S. degree in computer science from the National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan, in 2021. His research interest includes deep learning in finance.

WEI-CHE CHANG received the B.S. degree in computer science from the National Yang Ming Chiao Tung University (NYCU), Hsinchu, Taiwan, in 2020, where he is currently pursuing the M.S. degree in computer science. His research interest includes reinforcement learning in finance.

TIAN-SHYR DAI received the Ph.D. degree from the Department of Computer Science, National Taiwan University. He was the Chairperson of the Department of Information Management and Finance, from 2016 to 2019, and the Director of the Taiwan Association of Business School, from 2018 to 2020. He is currently a Full Professor at the Department of Information Management and Finance, National Yang Ming Chiao Tung University (NYCU). He is also a Research Member of the Risk and Insurance Research Center, NCCU. He has been a Senior Fellow of AdvanceHE and a Faculty Member of Beta Gamma Sigma, since 2021. His research interests include financial engineering and financial technology.

YING-PING CHEN (Member, IEEE) received the B.S. and M.S. degrees in computer science and information engineering from the National Taiwan University, Taiwan, in 1995 and 1997, respectively, and the Ph.D. degree from the Department of Computer Science, University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2004. He is currently a Full Professor with the Department of Computer Science, National Yang Ming Chiao Tung University (NYCU), Taiwan. His research interests include understanding intelligence from computational perspectives and via computational mechanisms, novel, emerging computational technologies, and theories, working principles, and dimensional/facet-wise models in genetic and evolutionary computation.

HAO-HAN CHANG is currently pursuing the Ph.D. degree with the Institute of Finance, National Yang Ming Chiao Tung University (NYCU). His research interests include statistical arbitrage and valuation employee stock options with forest model.