Solving Unconverged Learning of Pairs Trading Strategies with Representation Labeling Mechanism

Wei-Lun Kuo[a], Tian-Shyr Dai[b] and Wei-Che Chang[c]

[a]Institute of Data Science and Engineering, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC
[b]Department of Information Management and Finance, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC
[c]Institute of Computer Science and Engineering, National Chiao Tung University, 1001 University Road, Hsinchu, Taiwan 300, ROC

Abstract

A pairs trading strategy (PTS) constructs a market-neutral portfolio whose value typically moves back and forth around a mean price level; investors short (long) the portfolio when its value reaches the upside (downside) opening threshold and close the position when the value reverts to the mean to earn the price difference. Recent machine learning models select the open and stop-loss thresholds either heuristically or chosen from a limited set, which significantly limits the investment performance. We address this by creating a wider set of open/stop-loss threshold recommendations that generally cover all possible scenarios; but regression- or classification-based deep learning methods for recommending thresholds fail to converge. Thus, we design a representative labeling mechanism that selects representative open and stop-loss thresholds from all possible optimal thresholds according to the selection frequencies of the thresholds and the k-means algorithm. Experiments suggest that training the multi-scale residual network with stock pairs relabeled by representative thresholds yields better investment performance than other methods in the literature.

Keywords

Pairs trading, Representation labelling, ResNet, Opening and stop-loss triggers tuning

1. Introduction

A pairs trading strategy (abbreviated as PTS hereafter) is a popular market-neutral investment strategy introduced by Wall Street econometricians no later than the 1990s. Instead of guessing at unpredictable financial market trends, a PTS eliminates market tendency risk by simultaneously longing one stock and shorting another at a specific ratio. The net value of this long-short portfolio, referred to as the "spread", moves back and forth around a certain mean price level without being influenced by financial market trends, as suggested by the "market-neutral" modifier. A portfolio with this mean reverting property can be constructed by finding a pair (or a group) of stocks whose price processes have co-integration properties per the Johansen co-integration test [see 1]. We long (short) the portfolio when the spread is below (above) the mean price level to reach a lower (higher) opening threshold, and then close the portfolio when the spread converges to the mean level to earn the price difference.

Although much statistical literature focuses on generating high quality stock pairs for PTS, recommendations for a customized threshold for each stock pair have not been well-studied. In addition, a PTS is a "statistical" arbitrage strategy;[1] that is, the mean reverting property occasionally fails, leading to significant loss. Thus much literature adds a "stop-loss" threshold at which point the portfolio is closed to stop-loss when the spread diverges excessively from the mean level. Determining a proper opening and stop-loss threshold for each stock pair is critical, as the spread of each stock pair has a different price pattern that varies with changes in the financial markets. If the opening threshold is too far from the mean level, the likelihood to open the portfolio is too low to earn any benefit. However, if the threshold is too close to the mean, the resultant price differences are insufficient to cover transaction and price slippage costs.

Recently, reinforcement learning (abbreviated as RL hereafter) is used by Fallahpour et al. [2] and Kim and Kim [3] to recommend open/stop-loss thresholds to improve the investment performance of PTS. However, the actions—or open/stop-loss thresholds—of their RL models are either selected heuristically (with 6 actions) or chosen from a limited set (with 39 actions), which significantly limits the investment performance of their models as described in our experiments. We address this by creating a wider set of 300 open/stop-loss threshold recommendations that generally cover all possible scenarios, and then label each stock pair from the training set with one of 300 thresholds that maximizes the PTS profit. We find that methods based on regression- or classification-based deep learning (abbreviated as DL hereafter) all fail to converge. To resolve this problem, we develop a represen-

MUFin21: International Workshop on Modelling Uncertainty in the Financial World, November 01–05, 2021, Gold Coast, Australia
allenlike20@gmail.com (W. Kuo); cameldai@mail.nctu.edu.tw (T. Dai); destiny10191019.cs05@nctu.edu.tw (W. Chang)

[1]See https://en.wikipedia.org/wiki/Pairs_trade.

tative labeling mechanism that selects 25 representative thresholds (determined by the Elbow method) to represent 300 thresholds by picking most frequently selected thresholds or use the k-means method. Each stock pair is then relabeled with a representative threshold. Our alternative to learn from the 300-label stock pairs is changed to learn from 25-relabeled stock pairs. Experiments show that training a multi-scale residual network (abbreviated as ResNet) proposed by Li et al. [4] with relabeled stock pairs facilitates smooth and quick convergence. They also show that this representative labeling mechanism outperforms past work.

Our paper is organized as follows: Section 2 reviews PTS research and studies on relevant machine learning models. In Section 3, we discuss the construction of optimal open and stop-loss thresholds and the representative labeling mechanism adopted to address the failure to converge. Sections 4 describes how we select and incorporate the multi-scale ResNet into our PTS trading model. The experimental results in Section 5 confirm the superiority of our models. Section 6 concludes the paper.

2. Preliminaries

2.1. Literature Review

[5] shows that the techniques for finding stock pairs eligible for PTS can be classified into five approaches. Our stock pair generation method is based on the co-integration approach, as Rad et al. [6] and Huck and Afawubo [7] argue that this approach is better than other approaches. Engle and Granger [8] and Johansen [9] develop different statistical tests to determine whether the price processes of a stock pair possess the co-integration property; that is, there exists a linear combination of two stock prices that make the value process of this two-stock portfolio a stationary process. The stationary property ensures that statistical properties such as the mean of the value process do not change with time. Thus we can buy (sell) the portfolio when its value is below (above) the mean and cash out when the value converges back. These tests are used by Vidyamurthy [10] and Rad et al. [6] to detect stock pairs that are eligible for PTS.

[2], [3], [11], and [12] use the reinforcement learning (RL) method to determine opening/stoploss thresholds for PTS. Fallahpour et al. [2] enumerate 39 actions (i.e., 39 combinations of open and stop-loss thresholds) and reduce the threshold selection problem to a multi-armed bandit problem solved using a single-state RL model. Our experiments show that this naive mechanism fails to capture various properties of different stock pairs and is outperformed by other approaches in terms of investment results. Kim and Kim [3] instead use a deep Q-network (DQN), which outperforms Fallahpour et al. [2]'s model. They heuristically set six overly simplistic actions, which significantly limits the profitability as shown later. In addition, they train each PTS-eligible stock pair with a DQN, which necessitates a large number of DQNs. Confirming their observations, we find that co-integration properties for most stock pairs are not durable over a long period of time; thus only a small amount of stock pairs contain enough data to train the DQN. We instead train our machine learning model on trading data from all stock pairs; the resultant model recommends thresholds for all stock pairs. Brim [11] proposes the Double-DQN with three actions, but the low win rate limits the practical value of the model. Xu and Tan [12] uses deterministic policy gradient (DPG) to predict open and stoploss timing for PTS and the value weights of pairs to form a return-maximized portfolio. Hsu et al. [13] uses several deep learning models with the opinions on social media to predict the price movement in PTS. Threshold selection problem can be found in other domain. In order to manage dynamic network traffic, the number of Virtual Network Functions(VNFs) instances need to be chosen. Rahman et al. [14] models the problem as a classification problem, and uses several machine learning models to predict the number of VNFs.

The PTS literature largely adopts RL but here we use DL with representative labeling. Since our experiments show that a DL model with only a few layers fails to learn efficiently and accurately, we adopt the residual network (ResNet) model proposed by He et al. [15], which uses deeper layers to capture complex features/patterns in financial markets. He et al. [15] provide extensive empirical data demonstrating that ResNets are simpler to optimize and achieve higher learning precision due to their greater numbers of hidden layers. Furthermore, Li et al. [4] extend ResNet from a single scale to multiple scales by adding convolution kernels of various sizes to adaptively detect data features from different aspects. Our paper combines representative labeling with multi-scale ResNet to yield superior investment performance.

2.2. Co-integration Method and PTS

A trading duration—in this paper a business day—is divided into a formation period and a trading period: data in the formation period is used to select PTS-eligible stock pairs in the trading period. We use the co-integration approach [16, 10, 17, 18, 6] to find eligible stock pairs from a stock pool, for instance, the 0050 constituent stocks from the Taiwan stock market. Let the i-th pair be composed of stocks S_1^i and S_2^i, and let the capital invested in these two stocks be $\beta_1^i : \beta_2^i$ (if the stock pair is eligible). We extract the 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 co-integration property of $y(t)$ can be tested using the

Johansen co-integration test [see 1] 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 co-integration relations, $p-1$ denotes the VECM order, D_i is also a 2×2 matrix, and ϵ_t denotes a 2×1 white noise vector. We follow Lütkepohl et al. [19] in using the power test which decomposes Π as $\alpha\beta'$, where the 2×1 co-integration 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 co-integration test, then we construct a portfolio by investing the two stocks at the ratio $\beta_1^i : \beta_2^i$. The spread process of this portfolio

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

is market neutral and moves back and forth around the mean of the spread $E(P^i(t))$. We could also measure the variation of $P^i(t)$ by calculating its standard derivation σ^i. If we purchase this portfolio at time τ and sell it at time τ', the profit (or loss) can be expressed as product of the investment amount c and the difference of the spread:

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

where $\ln \frac{S_j^i(\tau')}{S_j^i(\tau)}$ denotes the return rate for investing S_j^i over the time period $[\tau, \tau']$. $\frac{c\beta_j^i}{S_j^i(\tau)}$ denotes the numbers of shares for trading S_j^i at time τ.[2]

The market-neutral nature of Equation (2) allows us to long (short) the portfolio when the spread is below (above) its mean and close the position when it converges to the mean to make a profit as illustrated in Figure 1. To increase the profit in Equation (3), which simultaneously covers the transaction cost, we find a suitable open threshold, defined as the product of a scalar ξ_O' and the volatility σ_i. We also find another stop-loss threshold, defined as the product of a scalar ξ_S' and σ_i, to prevent occasional failures of the market-neutral property from seriously influencing profits. The intersection of the spread P_t^i with either element of the trigger pair (ξ_O', ξ_S') determines the timing to long/short the portfolio or to stop loss, respectively. Specifically, if the spread P_t^i reaches the upper opening trigger (denoted by node B), then we short the portfolio with the value investment ratio $\beta_1^i : \beta_2^i$ for stocks S_1^i and S_2^i. After shorting the portfolio, P_t^i 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. On the other hand, if P_t^i falls to the lower opening trigger (denoted by node E), then we long the portfolio. After longing the portfolio, P_t^i may still fall to node H, in which case we close the portfolio to stop loss. Otherwise, it may reach node F, in which case we close the portfolio to gain a profit. Finally, the portfolio may still be open at the end of the trading period, say, the closing time of Taiwan Stock Exchange. In this case the portfolio is forced to close to avoid incurring risks to keep cross-day positions.

[2] We long (short) S_j^i if this value is positive (negative).

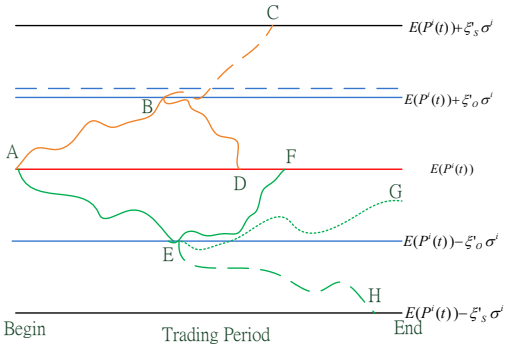


Figure 1: Possible Scenarios in the Trading Period. The red line, blue lines, and black lines denote the mean of $P^i(t)$, the triggers for opening the portfolio, and the triggers for stoping losses. The values of these triggers are listed on the right of these lines. The orange and green curves illustrate all possible scenarios to open the portfolio and to close the portfolio. After opening the portfolio, we use dash curves and solid curves to denote that the portfolio is closed to get profit and to stop loss, respectively. The dot curve denotes that the portfolio is closed at the end of the trading period. The period begins from time 0 and ends at time T. Node E and H occur at time τ and τ', respectively.

3. Representation Labeling Mechanism

We first describe our dataset and the preprocessing of the stock tick data, after which we discuss why it is difficult to train naive deep learning methods for regression or classification to pick feasible open and stop-loss thresholds for PTS. We propose the core idea and several variations of the representation labeling mechanism (abbreviated as RLM hereafter) to address this problem by generating "representation" thresholds and labels suitable for training deep learning methods.

3.1. Dataset and Preprocessing

The dataset used to develop and examine pair trading strategies is composed of the constituent stocks of Taiwan Top 50 ETF (0050) from January 1, 2013 to December 31, 2018. We adopt a day-trading strategy without holding positions overnight, as day trades provide 50% discounts on transaction costs,[3] which significantly increases win rates and profits. Figure 2 illustrates the overall procedure of the proposed PTS; Step 1 describes the data preprocessing. We first set non-overlapping training and testing periods from 2013 to 2018. The stock tick data for each business day in the training period generates the spread features and labels needed to train the RLM model, whose performance is then verified at each business day of the testing period. Daily trading is conducted from 9:00 a.m. to 13:30 p.m. for each business day, divided into the formation period (the first 166 minutes, ignoring the beginning of the first 16 minutes) and the trading period (e.g., the rest of the business day). We use the tick data from the formation period to calculate the weighted average stock price for each minute, and use the resultant time series to construct eligible stock pairs and corresponding investment ratios based on the Johansen co-integration method, as described in Section 2.2. The feature of the i-th stock pair is the spread process P_F^i (P_T^i) constructed by substituting the stock price processes of S_1^i and S_2^i during the formation period (training period) into Equation (2).

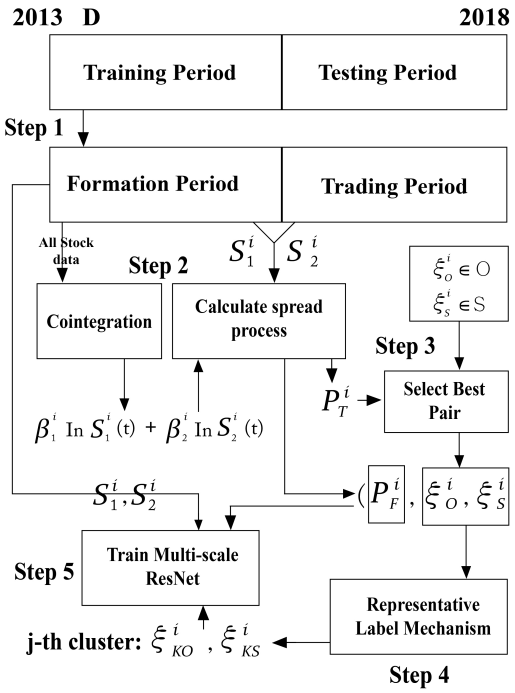


Figure 2: The Flowchart for Generating PTS Trigger Pairs with RLM.

3.2. Labeling: Finding the Optimal Trigger Pair

Now we label the i-th spread process in the formation period P_F^i with the optimal trigger threshold (ξ_O^i, ξ_S^i) that maximizes the profit when executing the PTS. Specifically, we long (short) the portfolio when the corresponding spread process in the trading period P_T^i reaches $-\xi_O^i\sigma_i$ ($\xi_O^i\sigma_i$) and stop loss when the process falls to $-\xi_S^i\sigma_i$ (or rises to $\xi_S^i\sigma_i$), as defined in Figure 1; the corresponding PTS profit is calculated by Equation (3). Note that both open and stop-loss triggers can be any positive real number, which makes the search for the optimal trigger thresholds (or the labeling process) intractable. In the literature [3, 2] either fixed triggers are used or optimal triggers are found from a limited set which is determined heuristically, which significantly weakens the performance as verified later. 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, after which we define the maximum derivation for each spread process during the formation period as $\max_t \left(P_F^i(t) - \mathrm{E}\left(P_F^i(t)\right)\right)$. A feasible stop-loss threshold set S is constructed by discretely enumerating equal-space samples from the range determined by the minimum and the maximum derivation of the spreads that do not converge to the mean. Hence S is defined as $\{1.5, 1.5+1, 1.5+2\times 1, \ldots, 24\}$.[4] Also, to ensure that the profit (which is proportional to the distance between the open threshold and the mean level) covers the transaction cost and the price slippage, the opening trigger generally should be larger than $0.5\sigma_i$. Thus, the open threshold set O is constructed by enumerating samples from the range determined by 0.5 and the maximum derivation of the spreads that converge to the mean. Hence the set O is defined as $\{0.5, 0.5+0.05, 0.5+2\times 0.05, \ldots, 8\}$, and all trigger thresholds (ξ'_o, ξ'_s) are generated by separately selecting the opening trigger threshold ξ'_o from set O and the stop-loss trigger threshold ξ'_S from set S. In addition, the condition $1.5\times\xi'_o < \xi'_s$ is enforced to prevent the two thresholds from being too close together, as such proximal thresholds increase the likelihood of closing the portfolio to stop loss immediately after opening the portfolio, which results in degraded trading performance. In addition, we add one more combination $(10, 25)$ with extremely high open and stop-loss thresholds to filter out stock pairs that are not suitable for trading. Then

[3] The transaction cost is 0.3% but reduces to 0.15% for day trading.

[4] Here we replace minimum and maximum derivations with nearby numbers 1.5 and 24, respectively.

we trade stock pair S_1^i and S_2^i by using the spread at the trading period P_T^i and the trigger threshold (ξ'_O, ξ'_S) to determine the timing for opening and stopping loss as in Figure 1 and calculate the profit with Equation (3). An optimal trigger threshold that maximizes the profit is

$$(\xi_O^i, \xi_S^i) \equiv \text{argmax}_{\xi'_O \in O, \xi'_S \in S} \left[\text{Profit}\left(P_T^i, \xi'_O, \xi'_S\right)\right]. \quad (4)$$

This trigger threshold becomes the label for the spread P_F^i for training the proposed machine learning models. There are about 300 combinations[5] for opening and stop-loss trigger thresholds that have been selected by at least one stock pair. Note that many aforementioned enumerated trigger thresholds are never selected by any stock pair. This is because the stock price is quoted as integral multiples of basic units (i.e., ticks) rather than continuously. Thus many trigger thresholds would not fit discrete changes of the spread process defined in Equation (2) due to discrete stock price quotes and thus will never be selected as optimal trigger thresholds. This explains why heuristically setting trigger thresholds [3] would significantly limit PTS performance. Since deriving proper trigger thresholds from discrete changes of the spread process could be difficult, thus we enumerate many thresholds and use Equation (3) to filter improper ones.

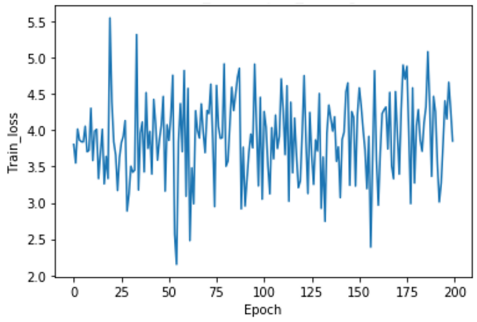
3.3. Failure of Naive Regression/Classification Deep Learning Approaches

Since the feasible ranges of both open and stop-loss thresholds are positive real numbers, it is natural to predict the optimal trigger threshold (ξ_O^i, ξ_S^i) by adopting a two-output-neuron regression-based deep neural network (RDNN) using the spread and stock processes (i.e., P_F^i, $S_1^i(t)$, and $S_2^i(t)$) during the formation period as input.

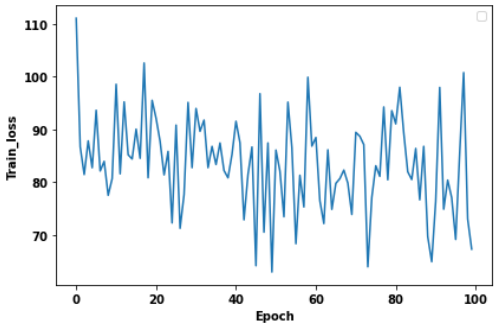
The RDNN training loss is defined as the mean square error (MSE) between the predicted thresholds and the optimal thresholds produced in Section 3.2. The training loss illustrated in Figure 3(a) does not converge as training proceeds, which shows that RDNN does not capture discontinuous relationships between open and stop-loss thresholds and profits. This is because minor shifts in either threshold can yield large changes in PTS profits, as illustrated in Figure 1. For example, a slight increment in the upper open threshold from the upper blue solid line to the blue dashed line removes the chance to short the portfolio at point B for the solid orange spread and reduces the profit to zero.

Instead of predicting open and stop-loss thresholds with regression-based approaches, we could select the optimal trigger threshold from all possible thresholds generated by the method described in Section 3.4 using classification-based approaches. As illustrated in Figures 3(b) and 3(c), it is difficult to train a reliable CNN or ResNet to recommend each stock pair with one of 300 possible thresholds. We use cross-entropy as the loss function, and use the spread and the stock price processes during the formation period, P_F^i, $S_1^i(t)$, $S_2^i(t)$, as inputs; we number all 300 possible thresholds as outputs. In this case the CNN training loss fails to converge, and that of ResNet converges only very slowly. We address this problem in the next section with representative labeling, which significantly reduces the number of labels without

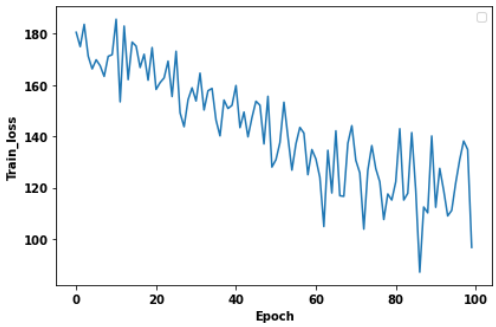
[5]This value changes with the training set data.



(a) **Losses of Training DNN**



(b) **Lossese of Training CNN**



(c) **Losses of Training ResNet**

Figure 3: Training Losses with different Models

significantly sacrificing threshold quality. Our experiments show that this mechanism improves the ResNet training accuracy and the resultant PTS investment performance.

3.4. Representation labelling

In contrast to machine-learning based work such as Kim and Kim [3] and Fallahpour et al. [2], which use RL to learn 6 to 60 trigger thresholds (i.e., actions), we propose a novel deep learning model with a representation labeling mechanism to train above 25 representative trigger thresholds (determined by the Elbow method) that represent the 300 optimal thresholds determined in Section 3.2. Experimental results in Section 5 show that using representations of the optimal trigger thresholds yields performance significantly better than that of the RL approach with heuristically selected thresholds.

The representation mechanism resolves the training problem of non-convergence by reducing the number of classifications and maintains trading performance via properly selected representations. Recall that we divide the i-th spread process defined in Equation (2) into a formation period part P_F^i and a trading period P_T^i, and then substitute P_T^i into Equation (4) to extract the optimal trigger threshold (ξ_O^i, ξ_S^i) that maximizes the benefit for trading the stock pair S_1^i and S_2^i. Then (ξ_O^i, ξ_S^i) can be viewed as the label for P_F^i; the trigger threshold distributions are illustrated in Figure 4(a). Pink, yellow, green, and blue reflect the magnitude of the probability for choosing a threshold as the optimal threshold. By excluding thresholds with probabilities lower than 0.1% and 0.5%, we obtain Figures 4(b) and 4(c). We observe that the trigger threshold distribution is widespread and far from uniform. In addition, for some thresholds, the probability of selecting them (denoted by pink or yellow nodes) as optimal thresholds is much higher than that of other thresholds. This significant lack of smoothness could explain why regression-based DL fails to converge in Figure 3(a).

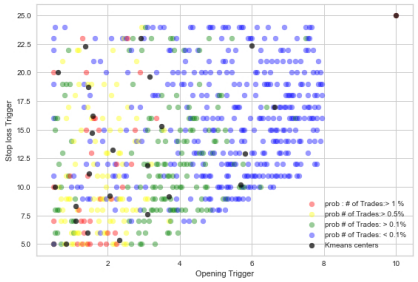
We address this lack of training convergence problem by setting representative trigger thresholds via either k-means or by using the thresholds with the top-k highest probabilities. In the first method, we partition all trigger thresholds into a reasonable number of clusters by the k-means algorithm; the cluster number 25 is determined by the Elbow approach. The set of representation thresholds $\mathbf{R}$ are defined as the centers of aforementioned clusters, and we call this label setting as `Kmean(0)`. Then the threshold for the i-th spread process is relabeled by picking one of these representation thresholds that maximize profit as follows:

$$(\xi_O^i, \xi_S^i) \equiv \text{argmax}_{(\xi_O', \xi_S') \in \mathbf{R}} \left[\text{Profit} \left(P_T^i, \xi_O', \xi_S' \right) \right]. \quad (5)$$

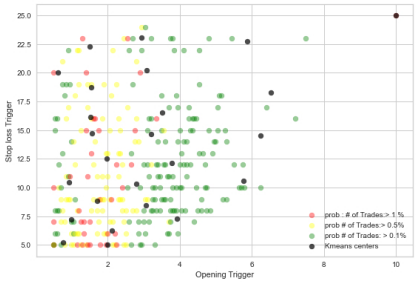
Note that each representation threshold selected by `Kmean(0)` (the black nodes) does not coincide with any optimal threshold, as k-means clustering calculates each cluster center by averaging; however, as mentioned above, slight shifts in the threshold (e.g., from the upper blue solid line to the dashed one in Figure 1) could yield significant changes to the profit. To prevent disturbances in low-probability thresholds from degrading the quality of the representation labels, we apply k-means on thresholds with probabilities larger than 0.1% and 0.5%, as illustrated in Figures 4(b) and 4(c); we term the resulting representation label settings `Kmean(1)` and `Kmean(2)`, respectively. In addition, to ensure that the representative thresholds coincide with an optimal threshold, we could alternatively choose as representation thresholds those trigger thresholds with the top 25 highest probabilities (denoted by the orange nodes), as shown in Figure 4(d); we term this the HighFreq label setting. In Section 5, we show how these representation labeling mechanisms outperform previous approaches.

4. Learning Model Constructions

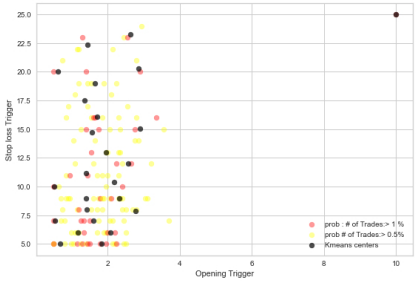
Given the stock and spread prices processes determined in Equation (2) as inputs and the representative thresholds generated in Section 3.4 as labels, we can compare the investment performance of several deep learning models and select the best one as the model used in step 5 of Figure 2. The input $x^i = \left[S_1^i, S_2^i, P_F^i\right]$ is formed by the price processes of the i-th stock pair S_1^i and S_2^i in the formation period and 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) is extended to 512 by padding the remaining positions with zeros. We number each of the 25 representative thresholds with a unique integer from the range $[1, 25]$. The number for the representative threshold recommended by the representative label mechanism (see Section 3.4), y^i, is used as the label for the stock pair i. We train the "plain" CNN (i.e., without adopting ResNet), the single-scale ResNet [15], and the multi-scale ResNet [4] with input x_i and ground truth y_i for each stock pair i from the training period. The CNN includes a one-dimensional convolutional layer with three channels (the spread and the two stock price processes) and 25 1×5 kernel maps. The output is sent to the batch normalization layer [20] to stabilize and speed up the training process; we use Leaky-ReLU activations. The results are passed through a one-dimensional convolutional layer with 50 kernel maps, a layer with 100 kernel maps, and a layer with 200 kernel maps, sequently; and the final outputs are then sent to a fully connected layer. The single-scale ResNet uses one size-3 convolution kernel, which applies to one chain of residual blocks. The three-scale ResNet adds size-5 and size-7 convolution



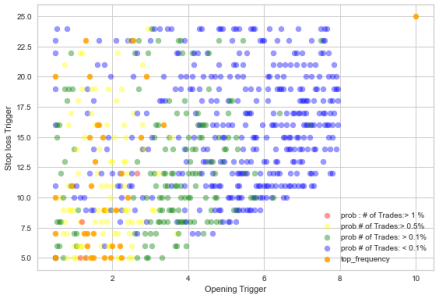
(a) The Distribution of Trigger Thresholds



(b) Trigger Thresholds Excluding the Thresholds with Probabilities Lower than 0.1%



(c) Trigger Thresholds Excluding the Thresholds with Probabilities Lower than 0.5%



(d) Trigger Thresholds and the Thresholds with Top-25 Highest Probabilities

Figure 4: The Distribution of Trigger Thresholds

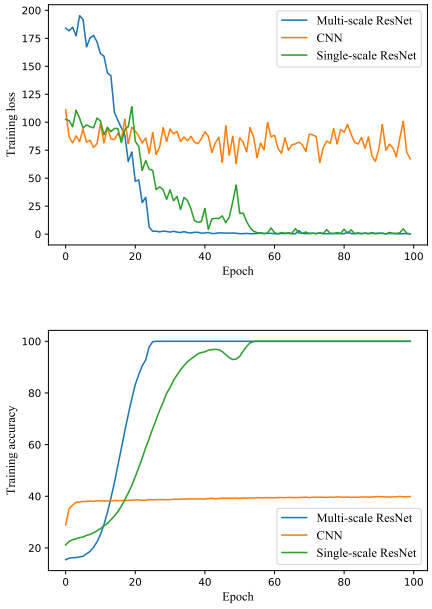


Figure 5: Training Accuracies and Losses among `CNN`, single- and multi-scale `ResNets`

kernels and two corresponding chains of blocks.[6] The features extracted by the three convolution kernels (i.e., the outputs from the three chains of residual blocks) are concatenated to form a feature vector which is then sent to a fully connected network.

The training results for these three models are shown in Figure 5. The training accuracy measures the percentage of correct predictions of all pairs in the training set; training loss is measured by cross entropy. The training accuracy for the CNN model, denoted by the orange curves, increases slowly, while the training loss oscillates significantly, which renders this model impractical. Thus we use a residual network, which employs hidden layers to capture complex features in financial markets. Although both the single- and three-scale ResNet achieve almost 100% accuracy and 0% loss after enough numerous training epochs, the latter mechanism converges more smoothly and quickly. Thus we adopt the three-scale ResNet in the following experiments.

To determine the number of training epochs, the data are divided into the training set and the validation set. We train the model on the training set data and run the resulting model on the validation dataset to calculate the accuracy and loss. To fairly retrieve useful information from the training dataset without overfitting, training is

[6]The structure of ResNet can be found in https://github.com/geekfeiw/Multi-Scale-1D-ResNet. The convolution kernel sizes 3, 5, and 7 are suggested by that website.

halted when the win rate of the validation set reaches a maximum.

5. Empirical Tests

We conducted experiments on the Taiwan Top 50 ETF component stocks from 2013 to 2018 to back-test improvements in PTS performance due to the proposed representative labeling mechanisms. As illustrated in Figure 2, information on stock pair eligibility and investment ratios is obtained by applying the Johansen co-integration test on half-minute average stock price data during the formation period. We then label the optimal trigger threshold for each stock pair (Sec. 3.2), relabel each pair with a representative threshold (Sec. 3.4), and train the ResNet model with stock pairs and representative labels retrieved from each training day in the training period. To evaluate the trading performance, we extract each trading day D from the testing period, retrieve stock pairs by applying the Johansen test to the formation period of day D, and predict the representative trigger threshold for each pair using the trained ResNet. We then use the retrieved stock pair and the threshold to execute tick-by-tick pair trading in the day's trading period. The transaction tax is set to 0.15%, as defined in the Taiwan Stock Exchange for day trading. To simulate price slippage effects, all trades are executed one tick after the spread process hits the trigger threshold.

To compare the trading performance of different PTS over the trading period, we list the (overall) profit, the win rate, the normal close rate, the number of trades, the Sharpe ratio (SR) calculated on the daily base or the pair base, the maximum drawdown (MDD), the (maximum) required capital, and the average profit (per trade), as illustrated in the first column of each table. The (overall) profit is the sum of the daily profit (or loss) for all trading days in the testing period, where the profit of day D is the sum of the profits when trading all PTS-eligible stock pairs on the day. The profit of each trade is calculated in Equation (3). The required capital for day D is measured as the sum of the capital required to execute each PTS on the day. The maximum required capital is defined as the maximum of the required capital for each trading day in the testing period. The daily (pair) return is then calculated as the daily (pair) profit divided by the maximum required capital (the capital required to trade the pair). The Sharpe ratio, which estimates the excess investment return divided by the corresponding risk, is calculated either on a daily basis as

$$\frac{\text{Daily return} - \text{Risk-free return}}{\text{Standard derivation of daily return}},$$

or on a pair basis as

$$\frac{\text{Pair return} - \text{Risk-free return}}{\text{Standard derivation of pair return}}.$$

The maximum drawdown is 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 made in the testing period. The normal close rate is defined as the number of trades whose spread process converges back to the mean[7] divided by the total number of trades. The profit per open is the average profit for each trade.

In the experiments in Section 5.1, we first compare the various DL methods and representative labeling methods discussed in Sections 4 and 3.4. We find that combining multi-scale `ResNet` and `KMean(0)` (or `HighFreq`) produce best investment results; thus we will use these settings in the following experiments. Section 5.2 demonstrates that the proposed mechanism for representative thresholds outperforms past threshold selection mechanisms.

5.1. Selection of Learning Models and Representative Labeling Mechanisms

To ensure the efficiency of training described in step 5 of Figure 2, it is necessary to select the proper machine learning models and representative labeling mechanisms. Table 1 compares the performance when training with CNN, the single-scale ResNet, and the multi-scale ResNet. The unstable, slow convergence of CNN clearly yields poor results. Multi-scale ResNet outperforms single-scale ResNet, as applying more convolution kernels with different sizes extracts more information from the input data. Accordingly, in subsequent experiments we use the multi-scale ResNet as the training model.

Table 2 compare different representative labeling mechanisms proposed in **Section 3.4**. In column 4, `KMean(0)`, `KMean(1)`, and `KMean(2)` denote the representative label settings that applying the k-mean methods on total optimal thresholds (see **Figure 4(a)**), the optimal thresholds selected with probability larger than 0.1% (see **Figure 4(b)**), and 0.5% (see **Figure 4(c)**), respectively. `HighFreq` picks the trigger thresholds with top 25 highest probabilities as in **Figure 4(d)**.

We observe that both the win rate and the normal close rate are high for these label mechanisms, as the spread processes after applying the co-integration test described in Section 2.2 are likely to have the mean reverting property. This suggests that a mechanism with large total opening numbers yields high profits and Sharpe ratios. Also, the total open number for KMean(0) is the highest of the four mechanisms as 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 optimal

[7] That is, the portfolio is neither closed to stop loss (like nodes C of H) or forced to close (like node G) as illustrated in Figure 1.

Table 1
Comparing Different Training Models.

Train 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.		
Test Period	2016			2017			2018		
Model	M-ResNet	S-ResNet	CNN	M-ResNet	S-ResNet	CNN	M-ResNet	S-ResNet	CNN
Profit (thousand)	2269.83	1920.12	354.13	2232.66	1799.20	35.84	2289.50	1927.66	344.72
Win rate (%)	79	77	75	77	77	77	75	75	74
Normal close rate (%)	81	77	76	76	76	77	77	77	78
Total open number	16776	14433	3130	11900	10631	400	13149	12228	1653
SR (daily based)	8.1789	6.1964	2.2175	7.4381	6.5731	0.4281	2.2287	1.9841	0.4318
SR (pair based)	0.2415	0.2349	0.0956	0.2188	0.1989	0.0719	0.1422	0.1498	0.1296
MDD	66	113	71	76	83	47	344	363	274
Trade capital (thousand)	50542.895	41740.827	26587.459	50783.951	39458.214	5715.334	80291.094	66228.371	24152.580
Profit per open (thousand)	0.1353	0.1330	0.1131	0.1876	0.1512	0.0896	0.1741	0.1595	0.2085

The training period, the validation period, and the testing period are listed in the first, second, and the third row, respectively. The performance of the CNN model (CNN), the single-scale ResNet (S-ResNet), and the multi-scale ResNet (M-ResNet) are compared with the performance indicators listed in the first column. The training data are the price processes of spreads and the two stocks (of the pair). Representative thresholds are generated by `HighFreq`. SR and MDD are abbreviations of Sharpe ratios and maximum drawdown, respectively.

Table 2
Compare Different Representative Labeling Mechanisms.

Train Period	Jan. 2015 - Oct. 2016.							
Validation Period	Nov. 2016 - Dec. 2016.							
Test Period	2017				2018			
Method	`KMean(0)`	`KMean(1)`	`KMean(2)`	`HighFreq`	`KMean(0)`	`KMean(1)`	`KMean(2)`	`HighFreq`
Profit (thousand)	2107.40	2056.18	1920.77	2232.66	2020.37	1802.69	1709.59	2254.96
Win rate (%)	75	76	76	77	74	74	74	75
Normal close rate (%)	75	75	76	76	76	75	76	77
Total open number	12195	11375	11434	11900	13771	12801	13028	13355
SR (daily based)	7.4766	7.1593	7.2001	7.4381	1.8919	1.5950	1.6401	2.1572
SR (pair based)	0.1825	0.1915	0.1942	0.2188	0.1322	0.1223	0.1281	0.1512
MDD	72	91	85	76	419	387	365	400
Trade capital (thousand)	49310.304	43552.278	48495.936	50783.951	64390.225	74759.619	74250.312	62389.603
Profit per open (thousand)	0.1728	0.1808	0.1679	0.1876	0.1467	0.1408	0.1312	0.1688

The training period, the validation period, and the testing period are listed in the first, second, and the third row, respectively. The fifth rows lists the mechanisms to generate representative thresholds. The indicators used to measure the performance are listed in the first column.

thresholds due to the average calculation, every threshold recommended by HighFreq is directly an optimal threshold with the highest 25 probabilities. Absent the disturbances on opening/stop-loss thresholds, HighFreq yields better pair-based results (i.e., SR (pair) and profit per open) than other k-mean-based mechanisms. Since KMean(0) and HighFreq possess advantages in different aspects, subsequent experiments use either KMean(0) or HighFreq for comparison.

5.2. Comparisons among Past Works

The statistical tests used to determine eligible stock pairs and investment weights (see Equation (2)) significantly influence PTS performance, as illustrated in Table 3. In additional to the co-integration test described in Section 2.2, Kim and Kim [3] propose the ordinary least squares (OLS) and total least squares (TLS) methods. The win rate and normal close rate of TLS and OLS are relatively low and the overall profits from 2016 to 2018 and the average profit (for each trade) are nearly all negative. In addition, the co-integration method yields more pairs that are eligible for trading (i.e., have larger total open numbers) than both TLS and OLS; thus the overall profit, the Sharpe ratios, and the average profit of the co-integrated method are all significantly better. To fairly compare all machine learning methods for threshold selections, for subsequent experiments we generated pairs and investment weights using the co-integrated method.

Table 4 compares different threshold selection mechanisms with our representative labeling mechanism. Fallahpour et al. [2] reduce the threshold selection problem to a multi-armed bandit problem and solve it using a reinforcement learning model with 39 actions (i.e, open and stop-loss thresholds) generated by Equation (4) with a much narrower set $O \in \{0.5, 1, \cdots 3\}$ and $S \in \{0.5, 1, \cdots 5\}$ (denoted by method 1). Kim and Kim [3] use deep reinforcement learning to select one of six heuristically-generated actions for trading (denoted by method 4). Note that as the number of actions (the threshold choices) in their papers is relatively small, their models do not suf-

Table 3
Comparing Statistical Testing Methods.

Train 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.		
Test Period	2016			2017			2018		
Model	Co-I	TLS	OLS	Co-I	TLS	OLS	Co-I	TLS	OLS
Profit (thousand)	727.67	-458.33	-254.02	1284.95	-439.49	-173.00	101.24	-140.05	72.60
Win rate (%)	67	44	57	67	43	53	65	44	55
Normal close rate (%)	66	50	62	67	52	62	67	49	60
Total open number	18627	3073	3826	15440	3390	3767	16388	2579	2954
SR (daily based)	2.5716	-7.0078	-4.0312	4.2652	-5.9474	-3.0189	0.1114	-0.4528	0.2577
SR (pair based)	0.1099	-0.2138	-0.0662	0.1113	-0.2151	-0.0831	0.0688	-0.0531	0.0141
MDD	103	46	50	52	64	28	382	68	35
Trade capital (thousand)	57155.256	10235.96	10572.52	62088.118	28371.28	23257.66	86349.235	15632.03	15781.25
Profit per open (thousand)	0.0390	-0.1491	-0.0664	0.0832	-0.1296	-0.0459	0.0061	-0.0543	0.0246

The training period, the validation period, and the testing period are listed in the first, second, and the third row, respectively. We compare the performance among TLS, OLS, and the co-integration method (Co-I) by using the deep reinforcement method proposed in Kim and Kim [3] to select actions (thresholds) for PTS.

Table 4
Comparisons among Different Threshold Selection Methods for PTS.

Train 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.			
Test Period	2016				2017				2018			
Method	1	2	3	4	1	2	3	4	1	2	3	4
Profit (thousand)	-1908.02	2269.83	1943.04	727.67	-2087.61	2232.66	1804.02	1284.95	-2395.03	2289.50	1756.73	101.24
Win rate (%)	40	79	79	67	41	77	75	67	46	75	75	65
Normal close rate (%)	38	81	78	66	40	76	77	67	47	77	78	67
Total open number	11977	16776	13733	18627	10955	11900	11384	15440	16072	13149	12092	16388
SR (daily based)	-9.17	8.1789	7.5343	2.5716	-10.83	7.4381	6.4397	4.2652	-3.24	2.2287	1.8119	0.1114
SR (pair based)	-0.20	0.2415	0.2340	0.1099	-0.21	0.2188	0.1866	0.1113	-0.09	0.1422	0.1336	0.0688
MDD	225	66	49	103	300	76	53	52	290	344	505	382
Trade capital (thousand)	45874	50542	44146	57155	58641	50783	47308	62088	67905	80291	68093	86349
Profit per open (thousand)	-0.1593	0.1353	0.1415	0.0390	-0.1906	0.1876	0.1593	0.0832	-0.1490	0.1741	0.1453	0.0061

The performance indicators (listed in the first columns) for the reinforcement learning method proposed in Fallahpour et al. [2] 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 in Kim and Kim [3] with 6 heuristic actions (method 4) are listed for comparisons.

fer from the training non-convergence problem described in Section 3.3. However, limiting the number of actions (or threshold choices) also limits PTS performance.

For a fair comparison with the six actions of the DRL approach [3], we add the HighFreq performance with six representative thresholds as method 3: HighFreq outperforms DRL in almost every aspect, even though DRL recommends more trading opportunities (i.e., has a higher total open number) which requires higher (daily) trade capital. However DRL's low win rate translates to lower overall profits and SR metrics than HighFreq with six representative thresholds. Increasing the number of representative thresholds from 6 to 25 (denoted as method 2) increases both the win rate and the total open number, and also improves profit and SR, and reduces risk (proxied by MDD). In addition, the naive reinforcement learning model proposed by Fallahpour et al. [2] performs poorly with a win rate lower than 50% and negative profits. If transaction costs are ignored, as in their experiments, the profit of their RL model becomes positive; thus their model fails to find proper thresholds to filter out unprofitable trades due to transaction costs. Generally speaking, the proposed representative threshold mechanism outperforms other relevant work.

6. Conclusions

To improve PTS investment performance, we adopt a supervised learning paradigm to recommend feasible opening and stop-loss triggers that differ greatly from existing RL-based approaches. To address the lack of convergence during training when using regression-based DL models to learn trigger thresholds, we reformulate the problem as a classification problem as follows. We first label each spread process with an optimal trigger pair that maximizes the trading profit. To avoid a huge number of labels from harming training performance, we relabel each spread process with our proposed representative labeling mechanism. Then we train the multi-scale ResNet with stock pairs relabeled by representative thresholds. Experimental results show that our proposed approach outperforms other existing approaches in terms of investment performance measures.

Acknowledgments

We thank the Ministry of Science and Technology (MOST) for supporting our work under 109-2622-H-009 -001 -

CC3.

References

[1] S. Johansen, Likelihood-Based Inference in Cointegrated Vector Autoregressive Models, Oxford University Press, 1995.

[2] S. Fallahpour, H. Hakimian, K. Taheri, E. Ramezanifar, Pairs trading strategy optimization using the reinforcement learning method: a cointegration approach, Soft Computing 20 (2016) 5051–5066.

[3] T. Kim, H. Y. Kim, Optimizing the pairs-trading strategy using deep reinforcement learning with trading and stop-loss boundaries, Complexity 2019 (2019) 1–20.

[4] J. Li, F. Fang, K. Mei, G. Zhang, Multi-scale residual network for image super-resolution, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 517–532.

[5] C. Krauss, Statistical arbitrage pairs trading strategies: Review and outlook, Journal of Economic Surveys 31 (2017) 513–545.

[6] H. Rad, R. K. Y. Low, R. Faff, The profitability of pairs trading strategies: distance, cointegration and copula methods, Quantitative Finance 16 (2016) 1541–1558.

[7] N. Huck, K. Afawubo, Pairs trading and selection methods: is cointegration superior?, Applied Economics 47 (2015) 599–613.

[8] R. F. Engle, C. W. J. Granger, Co-integration and error correction: Representation, estimation, and testing, Econometrica 55 (1987) 251–276.

[9] S. Johansen, Statistical analysis of cointegration vectors, Journal of Economic Dynamics and Control 12 (1988) 231 – 254.

[10] G. Vidyamurthy, Pairs Trading: quantitative methods and analysis, volume 217, John Wiley & Sons, 2004.

[11] A. Brim, Deep reinforcement learning pairs trading with a double deep q-network, in: 2020 10th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2020, pp. 0222–0227.

[12] F. Xu, S. Tan, Dynamic portfolio management based on pair trading and deep reinforcement learning, in: 2020 The 3rd International Conference on Computational Intelligence and Intelligent Systems, 2020, pp. 50–55.

[13] T.-W. Hsu, C.-C. Chen, H.-H. Huang, M. Chang Chen, H.-H. Chen, Hedging via opinion-based pair trading strategy, in: Companion Proceedings of the Web Conference 2020, 2020, pp. 69–70.

[14] S. Rahman, T. Ahmed, M. Huynh, M. Tornatore, B. Mukherjee, Auto-scaling vnfs using machine learning to improve qos and reduce cost, in: 2018 IEEE International Conference on Communications (ICC), IEEE, 2018, pp. 1–6.

[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[16] S. M. Sarmento, N. Horta, Enhancing a pairs trading strategy with the application of machine learning, Expert Systems with Applications (2020).

[17] J. Rudy, C. Dunis, G. Giorgioni, J. Laws, Statistical arbitrage and high-frequency data with an application to eurostoxx 50 equities, Available at SSRN 2272605 (2010).

[18] S. Broumandi, T. Reuber, Statistical arbitrage and fx exposure with south american adrs listed on the nyse, Financial Assets and Investing 3 (2012) 5–18.

[19] H. Lütkepohl, P. Saikkonen, C. Trenkler, Maximum eigenvalue versus trace tests for the cointegrating rank of a var process, The Econometrics Journal 4 (2001) 287–310.

[20] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the 32nd International Conference on Machine Learning, 2015, pp. 448–456.