A pair trading using reinforcement learning and wavelet decomposition

Mr. Panudate Nithinon

An Independent Study Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science in Financial Engineering
Department of Banking and Finance
FACULTY OF COMMERCE AND ACCOUNTANCY
Chulalongkorn University
Academic Year 2022

การซื้อขายแบบคู่ด้วยการเรียนรู้แบบเสริมกำลังและการแปลงข้อมูลเวฟเล็ต

นายภาณุเดช นิธินนท์

สารนิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมการเงิน ภาควิชาการธนาคารและการเงิน
คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2565

Independent Study Title	A pair trading using reinforcement learning and wavelet decomposition
By	Mr. Panudate Nithinon
Field of Study	Financial Engineering
Thesis Advisor	Assoc. Prof. Dr. THAISIRI WATEWAI

Accepted by the FACULTY OF COMMERCE AND ACCOUNTANCY, Chulalongkorn University in Partial Fulfillment of the Requirement for the Master of Science

INDEPENDENT STUDY COMMITTEE

.... Chairman
(Assoc. Prof. Dr. SIRA SUCHINTABANDID)

.... Advisor
(Assoc. Prof. Dr. THAISIRI WATEWAI)

.... Examiner
(Dr. Tanawit Sae-Sue)

ภาณุเดช นิธินนท์ : การซื้อขายแบบคู่ด้วยการเรียนรู้แบบเสริมกำลังและการแปลงข้อมูลเวฟเล็ต. (A pair trading using reinforcement learning and wavelet decomposition) อ.ที่ปรึกษาหลัก : รศ. ดร.ไทยศิริ เวทไว

ในการวิจัยนี้มีวัตถุประสงค์เพื่อ เสนอวิธีการปรับปรุงประสิทธิภาพการซื้อขายหุ้นแบบคู่ตามหลักการ Johansen cointegration การซื้อขายหุ้นแบบคู่คือการทำกำไรจากดุลยภาพของหุ้นสองตัวที่ราคามีความสัมพันธ์เชิงเส้นตรง ข้อมูลที่ใช้ในการทดลองเป็นข้อมูลราคาหุ้นใน SET index โดยใช้ Johansen cointegration test และค่า Pearson correlation เป็นเกณฑ์ในการตรวจหาและชี้วัดความสัมพันธ์เพื่อจับคู่หุ้น ข้อมูลสัญญาณการซื้อขายจะถูกนำมาแปลงข้อมูลแบบเวฟเล็ตด้วยอัลกอริทึม maximum overlap discrete wavelet transformation เพื่อแยกส่วนประกอบข้อมูลที่ขึ้นกับช่วงเวลาและลักษณะข้อมูลระยะยาวออกจากกัน หลังจากนั้นข้อมูลที่ถูกแปลงจะนำไปใช้ในการเรียนรู้แบบเสริมกำลังด้วยโครงข่ายประสาทเทียมซับซ้อนแบบคิวเพื่อเพิ่มประสิทธิภาพในการทำกำไร จากผลจากการทดลองพบว่า ระบบสามารถเพิ่มประสิทธิภาพในการทำกำไรได้ดีกับข้อมูลที่กำหนดให้เรียนรู้แต่ระบบไม่สามารถปรับปรุงประสิทธิภาพในการทำกำไรบนข้อมูลชุด validation ในการทดลองกับคู่หุ้นบางคู่ อีกทั้งยังขาดประสิทธิภาพในการทำกำไรกับข้อมูลทดสอบ สรุปผลการวิจัยในครั้งนี้ผลลัพธ์ที่ได้จากข้อมูลทดสอบมีความคลาดเคลื่อนสูงเทียบกับข้อมูลที่กำหนดให้เรียนรู้เนื่องจากปัญหา overfitting

สาขาวิชา	วิศวกรรมการเงิน	ลายมือชื่อนิสิต ....
ปีการศึกษา	2565	ลายมือชื่อ อ.ที่ปรึกษาหลัก ....

6284052226 : MAJOR FINANCIAL ENGINEERING

KEYWORD: Pair trading, Cointegration approach, Reinforcement learning, Maximum overlap discrete wavelet transformation, Deep-Q-network

Panudate Nithinon : A pair trading using reinforcement learning and wavelet decomposition. Advisor: Assoc. Prof. Dr. THAISIRI WATEWAI

In this study, we propose a trading optimization methodology for the pair trading strategy. The Johansen cointegration test and the correlation measure are used for pair selection. We apply Deep-Q-network (DQN) technique in which the trainable reinforcement learning agent is designed to directly control the trading positions. The maximum overlap discrete wavelet transformation (MODWT) algorithm is used for generating the trading signal from the spread time series. Wavelet signal preprocessing is used to extract the original time series into cyclic time series components and long-term behavior components. Based on the in-sample performance this trading model successfully solves a profit maximizing in the pair trading problem using wavelet components predictors. However, poor out-of-sample results observed in many sampled pairs indicate that the proposed procedure has an overfitting problem.

Field of Study:	Financial Engineering	Student's Signature ....
Academic Year:	2022	Advisor's Signature ....

ACKNOWLEDGEMENTS

I am grateful to everyone I have had the pleasure of working with on this project. I would like to thank my advisor for valuable guidance and discussions. Thanks should also go to his proofreading this work. I also thank the committees for their significant comments and suggestions.

Lastly, I would like to thank the staffs who helped me by providing valuable tools and data for this research.

Panudate Nithinon

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

1. Introduction

Pair trading is a statistical arbitrage strategy that generates profit by taking long positions on an asset and short positions on another assets. The basic idea behind pair trading is to take advantage of relative performance of a pair of assets. The process starts with finding two cointegrated price series. The spread time series is constructed from linear combination of two prices in the way that they form a mean reversion process. Then statistical arbitrage is to exploit the divergence of the spread in which the time series is known to converge in long-run. The profitability of pair trading relies on the condition that two prices remain their tracking movement. This circumstance frequently happened in the stock market since some set of stock share the same common factors. Following an ordinary pair trading rule, the spread is monitored for a divergence. Once the spread hits a determined boundary, we open a long position on the undervalued asset and open a short position on the overvalued asset. Then close the position when the spread converges long-run mean.

To improve the performance of this strategy, it is necessary to pick an appropriate level of boundary. Many researchers applied reinforcement learning to optimize this strategy. The trainable agent is assigned to dynamically select an optimal level of the trading boundary. Fallahpour *et al.* (2016) proposed an optimized high-frequency pair trading strategy on sampled stocks from the S&P. The agent is trained to optimize the n-arm bandit problem by selecting the four trading parameters: the cointegration estimation window, trading window, trading boundary and stop-loss boundary. The proposed method empirically outperformed fixed trading boundary method. Kim and Kim (2019) proposed an optimized pair trading strategy using the Deep-Q-Network (DQN) algorithm. The agent is assigned to select optimal trading and stop-loss boundaries. Based on the selected stocks from the S&P 500, the testing result shows that the proposed method outperformed a constant boundary strategy. Kim *et al.* (2022) developed a hybrid deep reinforcement learning. The learning agent is split into two learning networks. The first network is for trading action selection and the second network is for stop-loss determination. The strategy was tested using data from twenty stock pairs in the S&P 500. The empirical result shows that this structure generates a significantly higher profit compared to the previously proposed methods.

However, price modeling is a difficult task due to the chaotic movement in stock prices. Wavelet decomposition is a method that provides insight into the dynamics of time series. This method is increasingly popular in financial time series research. This technique analyzes the time domain and frequency domain by separating the time series into sub-series with different time scales. Several studies have found that applying a wavelet-based preprocess to time series improves forecasting accuracy when compared to the original time series method. Hsieh *et al.* (2011) presented a combination of wavelet transformation and the recurrent neural network for stock prediction. The Haar wavelet pre-processing method was used in this study to remove noise from technical indicator data. The stepwise regression-correlation selection is then used to create a collection of input features. The model testing is based on the market indices data: Dow Jones index, Financial Times Stock Exchange 100 (FSTE100), Nikkei 225 index, and Taiwan Stock Exchange Capitalization Weighted Stock Index (TAIEX). This model provided a higher forecasting accuracy compared to the straightforward forecasting method with the neural network. Kao *et al.* (2013) proposed a stock price forecasting model which integrates a wavelet transformation, multivariate adaptive regression splines, and Support Vector Regression (SVR). For feature extraction, the Daubechies wavelet, a sort of waveform of discrete wavelet decomposition, was used in this study. Researchers employed feature selection to optimize the model performance after decomposing the time series into different sub-series. This model is evaluated using the following stock market indices: The Composite index of China, the Bovespa index of Brazil, the Dow Jones index of the US, and the Nikkei 225 index of Japan. The empirical results indicate that the proposed method outperforms the Auto Regressive Integrated Moving Average model (ARIMA) and SVR, including the nested procedures of the proposed model. Lahmiri (2014) presented a wavelet transformation application for stock price forecasting. A process starts with using the discrete wavelet transform to decompose the price data. The obtained high-frequency components capture a discontinuity and rupture in the original data. The low-frequency component represents the long-term movement structure. These wavelet components are used as the input of a fully connected neural network. After testing with the sampled technology stocks from the USA market and S&P 500 index, the forecasting accuracy produced by this model outperformed both Auto Regressive Moving Average model (ARMA) model and the random walk model.

This paper presents a methodology for a pair trading optimization with a reinforcement learning method. The pair selection is conducted using the Johansen cointegration approach. To optimize the pair trading, we train a DQN agent with an objective to maximize a cumulative profit. The agent was designed to directly control the trading position, with the restriction that the estimated trading policy must follow the mean reverting exploitation strategy. We use both detail and approximation components from the spread time series decomposition as trading signal.

2. Related concepts

2.1 Johansen cointegration test

Cointegration implies that there is a long-run equilibrium on a set of time series variables. The notation $I(d)$ represents a time series that is integrated of order d. Consider the n times series variables $x_1, x_2, \dots x_n$ having the same order of integration $I(d)$. If there exists parameters of equilibrium relationship $\beta_1, \beta_2, \dots \beta_n$ that generate a linear combination $\beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n$ which is integrated of order $I(e)$ and $e < d$, then $x_1, x_2, \dots x_n$ are cointegrated. By the Johansen method, time series variables are formulated as a multivariate autoregression. The p-lag vector autoregression (VAR) is defined as

$$y_t = A_0 + A_1 y_{t-1} + A_2 y_{t-2} + \cdots + A_p y_{t-p} + \Phi D_t + \varepsilon_t \tag{1}$$

where y_t is an n-dimension vector of time series variable at time t, A_p is a coefficient matrix of lag p variable, D_t is a deterministic term, Φ is a deterministic coefficient. and ε_t follows independent and identically distributed (i.i.d.) Gaussian noise terms. The vector error correction model (VECM) is defined from the first difference series $\Delta y_t = y_t - y_{t-1}$ of VAR which can be written as

$$\Delta y_t = \mu + \Phi D_t + \Pi y_{t-1} + \Gamma_1 \Delta y_{t-1} + \Gamma_2 \Delta y_{t-2} + \cdots + \Gamma_{p-1} \Delta y_{t-p-1} + \varepsilon_t \qquad (2)$$

where $\Pi = A_1 + A_2 + \cdots + A_p - I$ and $\Gamma_i = A_1 + A_2 + \cdots + A_i - I\,, i = 1,2, \dots, p-1$. If there exists a reduced rank of matrix Π where $0 < rank(\Pi) = r < n$, then y_t is cointegrated. r indicates the number of cointegration relationship. The existence of reduced rank r implies $\Pi = \alpha\beta'$ where α and β are rank r matrices. The coefficient β is the vector of cointegrating coefficients which also represents the long-run equilibrium parameters. The series $\beta' y_t$ is stationary. The intercept term μ can also be interpreted as $\alpha\beta_0$ in which the β_0 is a cointegration intercept. The equation of VECM can be rewritten as follows.

$$\Delta y_t = \alpha(\beta_0 + \beta' y_{t-1}) + \Phi D_t + \Gamma_1 \Delta y_{t-1} + \Gamma_2 \Delta y_{t-2} + \cdots + \Gamma_{p-1} \Delta y_{t-p-1} + \varepsilon_t \quad (3)$$

The coefficient α captures the adjustment after deviation, $\beta_0 + \beta' y_{t-1}$ is a long-run equilibrium process, Γ_i captures short-run dynamic. There are two methods for the Johansen test which are Trace test and Maximum Eigen value test. This study focuses on the Trace test. This approach starts from estimating characteristic roots $\lambda_1, \lambda_2, .., \lambda_n$ of matrix Π. The null hypothesis of the test is $H_0 : rank(\Pi) \leq r_0$ against the alternative hypothesis $H_a : rank(\Pi) = n$. The statistical value for likelihood ratio test is calculated from $-T \sum_{i=r_0+1}^{n} \log{(1 - \lambda_i)}$ where T is the sample size. The test is carried out for r_0 as 0, 1, …, $n-1$ respectively. An estimation of $rank(\Pi)$ is obtained by the first value of r_0 that fails to reject the null hypothesis.

2.2 Maximal overlap discrete wavelet transformation

In signal processing, wavelet transformation is a time series transformation that is Fourier based. Unlike sine wave and cosine wave, wavelet allows us to expand or stretch wave signal along a time axis. Time series can be decomposed into many resolution components on the time domain and frequency domain. Each wave component guarantees locally stationary property with the condition that the original time series are not necessary to be stationary (Bozic & Babic 2015). The basic idea of this approach is to extract a embeded wave patterns from the original time series. The original time series can be considered as a mixture of many wave based components and long-term behaviours. High resolution components are related to the volatility that occurs within a short period. Low resolution components are related to the long-term movement.

The maximal overlap discrete wavelet transformation (MODWT) is an algorithm developed from a discrete wavelet transformation (DWT). This context focuses on the developed version of Percival and Mofjeld (1997). Both methods are stated that the original time series X_t can be decomposed into two series which are Smooth $S_{j,t}$ components and Details $D_{j,t}$ components. The equation of components is

$$X_t = \sum_{j=1}^{J} D_{j,t} + S_{j,t} \tag{4}$$

where $D_{j,t}$ is a detail time series at scaling level j = 1,2,3,... J, and X_t is decomposed for J times. $S_{j,t}$ is a smooth component of X_t after extraction of waves for j times. More specifically, the extraction at time j decomposes $S_{j-1,t}$ into $D_{j,t}$ and $S_{j,t}$ where $S_{0,t}$ is X_t (see Figure 1). To break X_t into smooth components and detail components, low-frequency and high-frequency filters are applied to X_t. In this context, the approach of Haar (1910) is implemented. The MODWT can be initiated by constructing filters. Define h_t as the time series vector $[1/\sqrt{2},\ -1/\sqrt{2},\ 0, 0, 0, \dots]$ and g_t as $[1/\sqrt{2},\ 1/\sqrt{2},\ 0, 0, 0, \dots]$. Both vectors have the same length as X_t which is denoted by N . The time series $\tilde{h}_t$ and $\tilde{g}_t$ are Discrete Fourier Transform (DFT) of h_t and g_t respectively and are defined by

$$\tilde{h}_t = \sum_{n=0}^{N-1} h_t \, e^{\frac{-2\pi i t n}{N}} \tag{5}$$

$$\tilde{g}_t = \sum_{n=0}^{N-1} g_t \, e^{\frac{-2\pi i t n}{N}} , \tag{6}$$

where t = 1, 2, .., N. Define time series $\tilde{h}^*{}_t$ and $\tilde{g}^*{}_t$as the downsampled time series of $\tilde{h}_t$ and $\tilde{g}_t$ respectively. To downsampled the time series, we first expand the time series in equation (2) and (3) infinitely with their full length duplicates. Then the downsampled time series are constructed from picking the values from the expanded series for every 2^{j-1}th. The mathematical notations can be shown as:

$$\tilde{h}^*{}_{j,t} = \tilde{h}_{2^{j-1}t \bmod N} \tag{7}$$

$$\tilde{g}^{*}{}_{j,t} = \tilde{g}_{2^{j-1}t \ mod \ N} \tag{8}$$

where t = 1, 2, .., N. Let $\widetilde{V_{0,t}}$ denote a DFT of X_t given by

$$\widetilde{V_{0,t}} = \sum_{n=0}^{N-1} X_t \, e^{\frac{-2\pi itn}{N}} \tag{9}$$

where t = 1, 2, .., N. Then

$$\widetilde{W}_{j,t} = \tilde{h}^{*}{}_{t} \cdot \tilde{V}_{j,t} \tag{10}$$

$$\tilde{V}_{j+1,t} = \tilde{g}^{*}{}_{t} \cdot \tilde{V}_{j,t} \ . \tag{11}$$

After decomposition for j times, wavelet transformation output $D_{j,t}$ and $S_{j,t}$, are the inverse of the DFT of time series $\widetilde{W}_{j,t}$ and $\tilde{V}_{j,t}$ respectively.

$$D_{j,t} = \frac{1}{N} \sum_{n=0}^{N-1} \widetilde{W}_{j,t} \, e^{\frac{2\pi itn}{N}} \tag{12}$$

$$S_{j,t} = \frac{1}{N} \sum_{n=0}^{N-1} \tilde{V}_{j,t} \, e^{\frac{2\pi itn}{N}}, \tag{13}$$

where t = 1, 2, .., N. The time series need to be recursively decomposed for j times to receive the j th level components. See Figure 1 for a wavelet transformation algorithm.

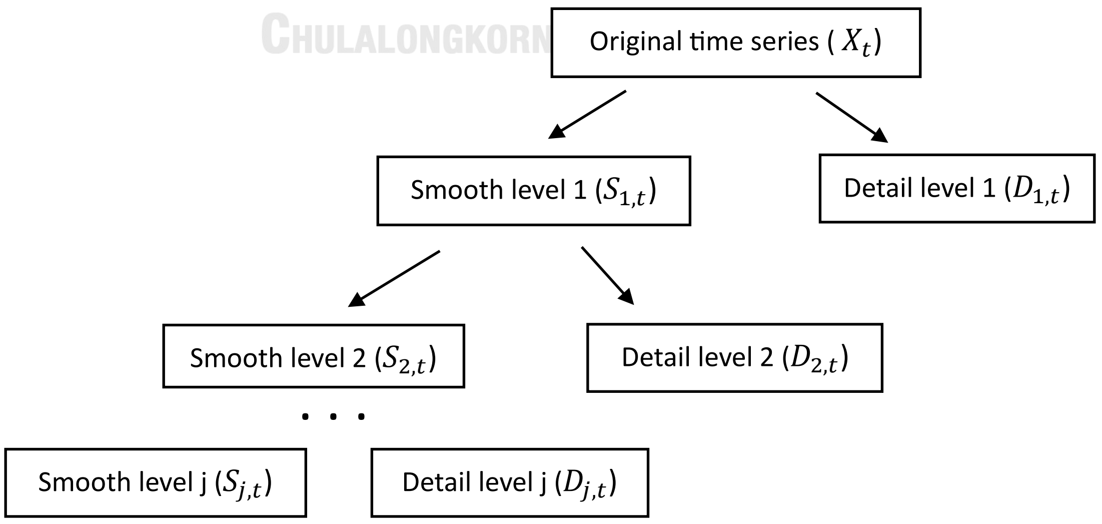
Original time series (X_t)
Smooth level 1 ($S_{1,t}$)
Detail level 1 ($D_{1,t}$)
Smooth level 2 ($S_{2,t}$)
Detail level 2 ($D_{2,t}$)
. . .
Smooth level j ($S_{j,t}$)
Detail level j ($D_{j,t}$)

Figure 1 Wavelet extracted components

2.3 Deep Q-network

Reinforcement learning is a type of machine learning in which the agent is assigned to take a desired action in an environment for determined reward. In each time-step, the agent receives the observed input state s_t to chooses the best action a_t. In the next time-step, the agent receives the observed input state s_{t+1} and observed reward r_{t+1} which relates to the action a_t in the previous time-step. The agent updates the decision-making policy based on the observed experience transition $(s_t, a_t, s_{t+1}, r_{t+1})$ in the adjacent time-steps with an objective to maximize the cumulative reward function.

Deep Q-network (DQN) is a type of reinforcement learning method which utilizes the neural network agent to execute an action selection in a finite Markov decision process. A neural network structure allows the agent to handle a difficult task in a complex environment. The agent is assigned to interact with an environment to maximize a future discounted reward function defined as

$$R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t \tag{14}$$

where T is a number of time-steps until termination, γ is a discount factor, r_t is a reward obtained at time-step t. The action-value function $Q(s, a)$ indicates the estimated future discounted reward of the input state variable s and action a. The optimal action-value $Q^*(s', a')$ follows the intuition of the Bellman equation. All possible actions in the next time-step a' and state variables in the next time-step s' are known values. Accordingly, an optimal policy is to select the a' that maximizes the expected value of $r + \gamma Q^*(s', a')$. In the DQN framework, the set of weights of the Q-network θ representing the policy is iteratively optimized with an objective to update the $Q(s, a)$ toward the target $r + \gamma Q(s', a')$.

Mnih *et al.* (2015) proposed a more stable algorithm of DQN. The weight updating method in the original paper is unstable. While training the network, the value $Q(s, a)$ typically increases. $Q(s', a')$ and the target value also increases because of using the same set of weights for updating. The target value keeps moving which leads to divergence or oscillation. To improve stability of this algorithm, the authors suggested to use a seperated Q-network θ^{target} for producing target values. In every certain number of iteration, weights of the updated network θ^{online} are cloned to θ^{target}. Adding time delay to a change in target value reduces divergence or oscillation. Moreover, reinforcement learning can also be unstable due to the following causes, correlation presented in the sequence of the observations and correlation between action-value and target value $r + \gamma Q^*(s', a')$. The authors implemented these two main procedures to reduce the instabilities. First, each observation (s_t, a_t, s_{t+1}, r_t) is kept in the replay memory dataset D for each time-step. During updates, samples are randomly drawn from D to perform batch gradient descent, therefore the data are averaged with many previous observations and the effect from correlation in the sequence of data is removed. Second, the iterative update of the action-value is allowed only periodically updates, thereby correlation with the target is reduced.

Additionally, observation exploration can be enhanced by performing a random action in some time-steps. The action execution aligns the ε-greedy policy in which the maximized action-value is chosen with probability $1 - \varepsilon$ and a random action is chosen with probability ε where ε decreases through time. Random actions reduce the number of unseen observations generated by unselected actions. During training, experience transitions are sampled with the same chance of being selected from the replay memory $(s, a, r, s') \sim U(D)$ to perform a batch gradient descent step. In each iteration i, the weight of Q-network θ_i^{online} is optimized with the target $y = r + \gamma \max_{a'} Q\left(s', a'; \theta_i^{target}\right)$. The objective is to minimize the following loss function.

$$L_i(\theta_i^{online}) = E_{(s,a,r,s') \sim U(D)}\left[\left(y - Q\left(s, a; \theta_i^{online}\right)\right)\right] \quad (15)$$

Van Hasselt *et al.* (2015) found that the max operation, used in both action selection and evaluation, leads to an over-optimistic estimation. They proposed an improved method by rewriting the formula of the target action-value as follows

$$y = r + \gamma Q(s', \underset{a'}{\operatorname{argmax}}\, Q\left(s', a'; \theta_i^{online}\right); \theta_i^{target}) \quad (16)$$

θ_i^{online} should be used for action selection instead of θ_i^{target}, as it is a current set of Q-network weights that generates a current estimated result, thereby the immediate action is fairly evaluated with another network θ_i^{target}. A performance of DQN using the new target formula empirically outperforms the original algorithm in cases of playing various emulator game.

3. Methodology

3.1 Data and cointegration estimation

Pair trading is a statistical arbitrage strategy which exploits the temporary deviation in a long-run equilibrium in two related prices. In this study, the Johansen cointegration approach is employed to identify the statistical relationship. This approach aims to construct a stationary spread from a linear combination of two non-stationary prices. The time series of prices are considered as $I(1)$ series. If a linear combination of prices becomes a stationary $I(0)$ series, the corresponding price series are cointegrated. This research focuses on employing the Johansen test to seek for cointegrated stocks in which the relationships of prices is assumed to follow bivariate VECM with no deterministic term. The model incorporates one lag of short-run effect. To implement a pair trading, a mean reverting timeseries $spread_t$ is calculated by $spread_t = \beta_0 + \beta_1 P_{1t} + \beta_2 P_{2t}$. By the ordinary trading rule, a trading signal is triggered when the deviation of the spread exceeds the determined trading boundary. When the spread falls below the lower boundary, open a long position on the asset with a positive coefficient and a short position on the asset with a negative coefficient When the spread exceeds the upper boundary, we execute opposite positions. After the spread reverses to its mean, we close the position to realize a profit. The portfolio requires that the ratio of quantity between two assets must remain β_1/β_2.

The dataset includes sampled stocks from the Stock Exchange of Thailand index (SET index). The daily price data is downloaded from Thomson Reuter data streaming. For pair selection, the cointegrated stock pairs with the highest Pearson correlation measures are selected. We use MATLAB to carry out the Johansen cointegration test to a pairwise of stock within the same Sector index determined by SET. The experiments incorporate the top correlated pairs which are INTUCH-

ADVANC, TISCO-TCAP, UNIQ-PREP, STEC-CK, LH-BLAND, QH-LH, , QH-BLAND. Table 1 shows the pair description. Table 2 shows the Johansen cointegration test results. The dataset is preprocessed based on one year formation window and six month trading window beginning on the first day of the year and keeps rolling forward. Only stacked data from the trading windows are used for the experiments. Data of the year 2012 to 2016 are utilized for reinforcement agent learning, data of the year 2017 to 2018 are used for validation, and data of the year 2019 are used for testing.

Table 1 Selected pair description

Stock pair	Industry	Sector	Correlation
INTUCH-ADVANC	Technology	Information and communication technology	0.9664
TISCO-TCAP	Financials	Banking	0.9498
UNIQ-PREB	Property and construction	Construction services	0.9497
STEC-CK	Property and construction	Construction services	0.9381
LH-BLAND	Property and construction	Property development	0.9364
QH-LH	Property and construction	Property development	0.9073
QH-BLAND	Property and construction	Property development	0.9063

Table 2 Johansen cointegration test results of selected pairs

Stock pair	Johansen cointegration test			
	Null hypothesis	Statistic value	Critical value	P Value
INTUCH-ADVANC	$rank(\Pi) \leq 0$	21.8585	20.2619	0.0299
	$rank(\Pi) \leq 1$	5.8160	9.1644	0.2099
TISCO-TCAP	$rank(\Pi) \leq 0$	23.0277	20.2619	0.0203
	$rank(\Pi) \leq 1$	1.4758	9.1644	0.8776
UNIQ-PREB	$rank(\Pi) \leq 0$	22.2858	20.2619	0.0260
	$rank(\Pi) \leq 1$	3.4592	9.1644	0.5639
STEC-CK	$rank(\Pi) \leq 0$	22.5061	20.2619	0.0242
	$rank(\Pi) \leq 1$	3.4942	9.1644	0.5587
LH-BLAND	$rank(\Pi) \leq 0$	30.5301	20.2619	0.0018
	$rank(\Pi) \leq 1$	3.9833	9.1644	0.4852
QH-LH	$rank(\Pi) \leq 0$	24.2885	20.2619	0.0134
	$rank(\Pi) \leq 1$	4.5175	9.1644	0.4050
QH-BLAND	$rank(\Pi) \leq 0$	21.3089	20.2619	0.0358
	$rank(\Pi) \leq 1$	4.2268	9.1644	0.4486

3.2 Problem formulation

By the reinforcement learning framework, the environment is a pair trading simulation on the daily closing price data. In each time step, the agent receives the input state $s \in S$ from the input space where set S incorporates the selected trading signals. Then the agent chooses a trading action $a \in A$ from the action space A. In the learning process, the action choosing policy π is updated with a goal to maximize the discounted future reward $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_t$ where r_t is one period ahead return.

3.3 Input states

The input states function as a trading signal. This study aims to detect the pattern in the movement of the spread time series. Then, we apply MODWT to the spread series. Wavelet transformation starts at the first period of the formation period. We use the extracted components which are the detail components and the approximation components as trading signals. The input set includes the signal sa_t, sb_t and time series values of lag m wavelet extraction at level J. The signal sa_t equals 1 if the current value of the spread is above the zero line otherwise sa_t equals 0. Similarly, sb_t equals 1 if the current value of the spread is below the zero line otherwise sb_t equals 0. To conclude, the set of trading signals is

$$S = \{(sa_t, sb_t, D_{i,t}, D_{i,t-1}, \dots D_{i,t-m}, S_{J,t}, S_{J,t-1}, \dots S_{J,t-m}), i = 1,2,..J\} \quad (17)$$

3.4 Action space

In each trading period, the agent is assigned to choose one action from the set of actions $A = \{a_1, a_2, a_3\}$. The action a_1 means to invest in the first asset by v_1 and invest in the other asset by v_2. The action a_2 takes no trading position. The action a_3 means to invest in the first asset by $-v_1$ and invest in the other asset by $-v_2$. By the cointegration approach, the investment proportion v_1 and v_2 are normalized values of the cointegration coefficient β_1 and β_2 from the VECM. This strategy takes a long position on the asset with a positive coefficient and a short position on the asset with negative coefficient. When the spread passes zero line or the trading window ends, the reinforcement agent is forced to close the position automatically.

3.5 Reward function

By the environment simulation, actions taken at the end of each trading day yield a trading profit that is observable from a pair trading simulation. The reward relates to a one-period ahead return from trading executions :

$$r_{a',t} = \frac{v_1(P_{1,t+1} - P_{1,t})}{P_{1,t}} + \frac{v_2(P_{2,t+1} - P_{2,t})}{P_{2,t}} - C(a') \tag{18}$$

where $r_{a',t}$ is a reward function if the agent takes an action a'. $P_{i,t}$ is price data of asset i at time t. n_i is an investment proportion of asset i. $C(a')$ is a constant transaction cost. When the a'causes agent to open a position, $C(a')$ equals a fixed percentage value to the investment amount or otherwise $C(a')$ is zero. When the agent executes inappropriate actions such as betting for the spread divergence, $r_{a',t}$ is replaced by a constant penalty PNT. The size of the penalty should be much larger than a possible one-period profit to disrupt inappropriate actions.

3.6 Training process

In this study, the DQN training algorithm is based on the developed version of Van Hasselt *et al.* (2015). For a more stable learning curve, we apply the target network soft updating. Kobayashi and Ilboudo (2021) demonstrated that it outperforms conventional target updating. A training procedure is shown in *Algorithm 1*. The action space is initialized with no trading position. There are five training passages with five different initial points. The notation % represents a modulus operator. For reinforcement learning, the data are split into three periods: training period, validation period, and testing period. The data in training period are used for agent learning. The episode means a forward run on the sequence of observation transitions in the training data until terminated. To avoid overfitting, the data in the validation period are used for measuring the performance of the trained agent. We apply the early stopping to the training passage. Figure 2 illustrates the learning curve in the training period and the validation period. The marked point is where the model is chosen. The agent is set to end the training passage when the reward during the validation period stops increasing. While training, the action selection is based on the decay ε-greedy policy. The probability ε decreases through training iterations.

Algorithm 1 training algorithm

```
Initialize weight of neural network $\theta_{online}$, $\theta_{target}$
Initialize replay memory $D$
foreach episode do
    foreach time-step $t$ do
        perform random action $a_t$ with probability $\varepsilon$
        otherwise choose action from $a_t = \max_a Q(s_t, a; \theta_{online})$
        simulation pair trading based on action $a_t$ to get reward $r_t$
        preprocess next time steps state variable $s_{t+1}$
        store new observation $(s_t, a_t, r_t, s_{t+1})$ into $D$
        if $t$ % $Update\_Freq = 0$
            sample experience transitions from replay memory $D$
            compute $y_t = r_t + \gamma Q(s_{t+1}, \operatorname{argmax}_{a'} Q(s_{t+1}, a'; \theta^{online}), \theta^{target})$
            compute loss function $L = (y_t - Q(s_t, a_t; \theta^{online}))^2$
            perform a step of gradient descent with loss $L$
            soft update target $\theta^{target} = \tau\theta^{online} + (1-\tau)\theta^{target}$
        end if
    end for
    evaluate the model
end for
```

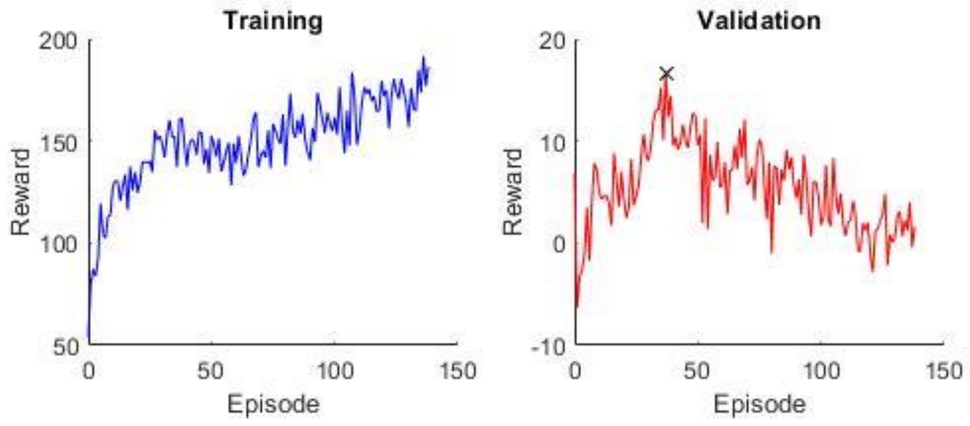

Figure 2 Early stopping on learning curves

The model architecture and data preprocessing are illustrated in Figure 3. The neural network topology is a simple feed forward network. Nodes in each layer is fully connected to the adjacent layer. The inputs are wavelet decomposed components.

The network composes of two hidden layers. The activation function of the hidden layer is the Rectified Linear Unit (ReLU). The output activation function is a linear function. The weights of Q-network are initialized using the method of He *et al.* (2015). The Q-network are optimized using the Adam algorithm proposed by Kingma and Ba (2014). Table 3 shows fixed hyperparameters in all experiments. This set of hyperparameters provides an increasing pattern of cumulative training reward.

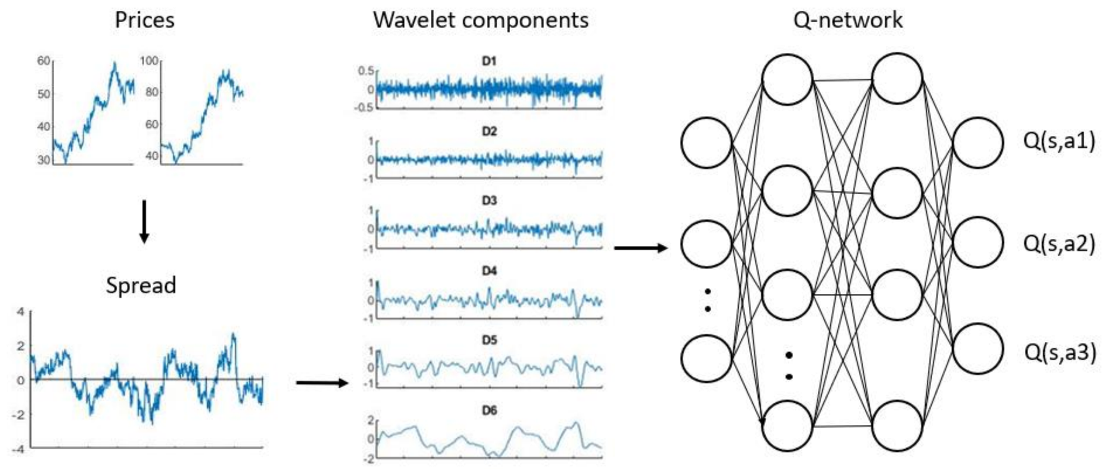

Figure 3 Model architecture

Table 3 Hyperparameters

Hyperparameter	Value
Batch size	32
Online network update frequency $\boldsymbol{Update\_Freq}$	Every 4 time-steps
Target soft update weight $\boldsymbol{\tau}$	0.001
Discounted factor $\boldsymbol{\gamma}$	0.99
Maximum memory replay size	100,000
Initial random action probability $\boldsymbol{\varepsilon}$	1
Minimum random action probability $\boldsymbol{\varepsilon}$	0.5
Learning rate	0.001
Momentum of stochastic gradient	0.9
Momentum of the squared gradient	0.99
Epsilon	10^{-7}
Early stopping	100 episodes

3.7 Evaluation and benchmarks

The trained DQN agent is evaluation without ε-greedy policy. The trading performance indicators cover both return and risk dimensions. In terms of risk, we consider only the return stability and downside risk. Table 4 shows the list of performance indicators. The notation r_t denotes a one-period percentage return from executing a pair trading and CR_t is a cumulative return series calculated from of r_t, The Sharpe ratio is calculated with an assumption that a risk-free rate equals 2% yearly basis. The excess return is calculated by the linear interpolation technique as a daily rate r_f distributed into each trading day. All indicators are interpreted on a yearly basis. The performance of the best agent is compared to the traditional pair trading strategy. We use various trading boundary settings as baseline strategies. Fixed trading boundary (FTB) executes trading when the spread diverges over a fixed level TB. This baseline strategy does not include the stoploss boundary. The trading executions are as follows: perform action a_3 when the spread rises above TB, perform action a_1 when the spread falls below $-TB$, and perform action a_2 once the spread passed the zero level. The comparison cases are FTB(0.5), FTB(1.0), FTB(1.5), and FTB(2.0) which have TB equals 0.5, 1.0, 1.5, and 2.0 respectively.

Table 4 Performance indicators

Indicator	Description	Formula
Return	Rate of return, measuring the profitability.	$R = \Sigma_{t=1}^{T} r_t$
Sharpe ratio	Average excess return divided by standard deviation of excess return, measuring return and stability trade-off.	$\sigma = StdDev\left[r_t - r_f\right]$ $SR = \dfrac{E(r_t - r_f)}{\sigma}$
Maximum drawdown	Maximum of historical the peak-to-trough decline in cumulative gain/loss, measuring the downside risk.	$Peak_t = \max_{t' \in [0,t]} CR_{t'}$ $MDD = \max_{t' \in [0,T]} \left(\dfrac{Peak_{t'} - CR_{t'}}{CR_{t'}}\right)$

4. Result and discussion

4.1 In-sample results

In this experiment, the model comparison is based on the sum of rewards through the dataset. We study the effects of varying the number of lagged input components, starting with no lags and extending to wavelet levels 3 and 5 while incorporating hidden node pruning. Table 5 displays the optimized models for each pair. All training passages show a rising pattern in the monitored cumulative reward, indicating the nonlinear mapping between wavelet component inputs and the Q-function of trading actions is well captured by the model. By the simulation using data in the training period, the peaked model in the learning curves outperformed the baseline strategy FTB in all cases. On average, the proposed method generates a 26.47% higher return than the optimal FTB strategy during the training period. For model generalization, an increased curve in the training reward is expected to result in an increasing curve in the validation reward. This pattern occurs in the INTUCH-ADVANC, TISCO-TCAP, UNIQ-PREP, STEC-CK, and QH-LH. Typical learning curves of the corresponding pairs are presented in Figure 4 and Figure 5. The validation performance has an increasing pattern followed by a decreasing pattern. On the other hand, an increasing validation curve does not appear in the learning curves of LH-BLAND and QH-BLAND in some training passages. Figure 6 illustrates that the validation performance has no clear improvement through the training iterations. This occurrence indicates that the related distribution of Q-value and wavelet predictors in the training data and the validation data are different. Therefore, the generalizability of the proposed model is inconsistent.

Table 5 Optimized hyperparameters

Stock pair	Wavelet level	Lag features	Neural network nodes	Validation return
INTUCH-ADVANC	3	2	19-12-12-3	14.17%
TISCO-TCAP	5	0	13-12-12-3	17.93%
UNIQ-PREB	5	1	19-16-16-3	19.86%
STEC-CK	3	1	15-24-24-3	25.44%
LH-BLAND	5	0	13-8-8-3	5.52%
QH-LH	5	2	25-16-16-3	22.27%
QH-BLAND	3	2	19-16-16-3	6.84%

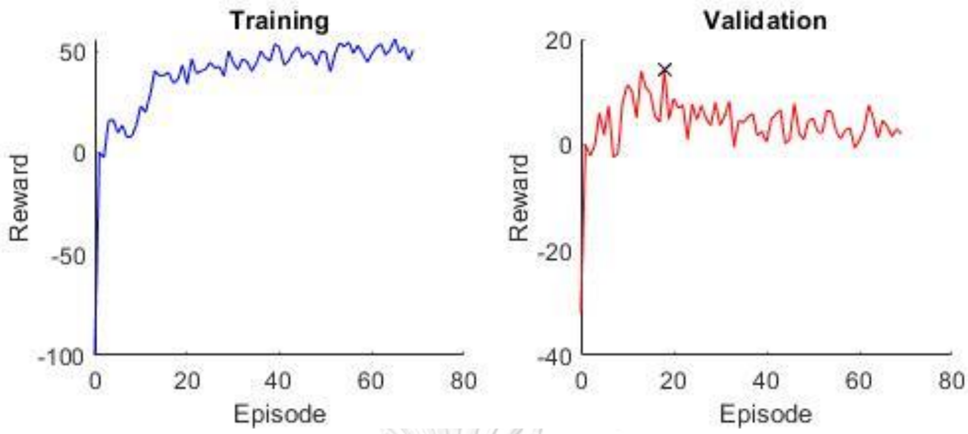

Figure 4 Sample of learning curve in the pair INTUCH-ADVANC

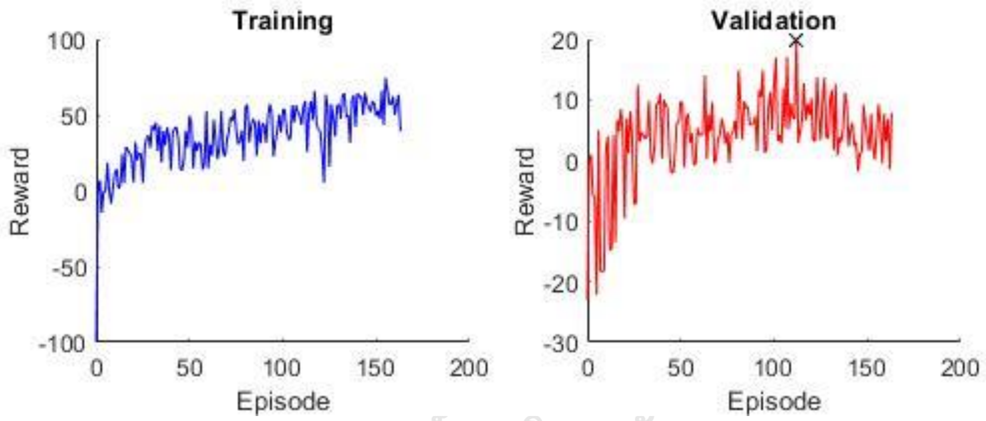

Figure 5 Sample of learning curve in the pair UNIQ-PREP

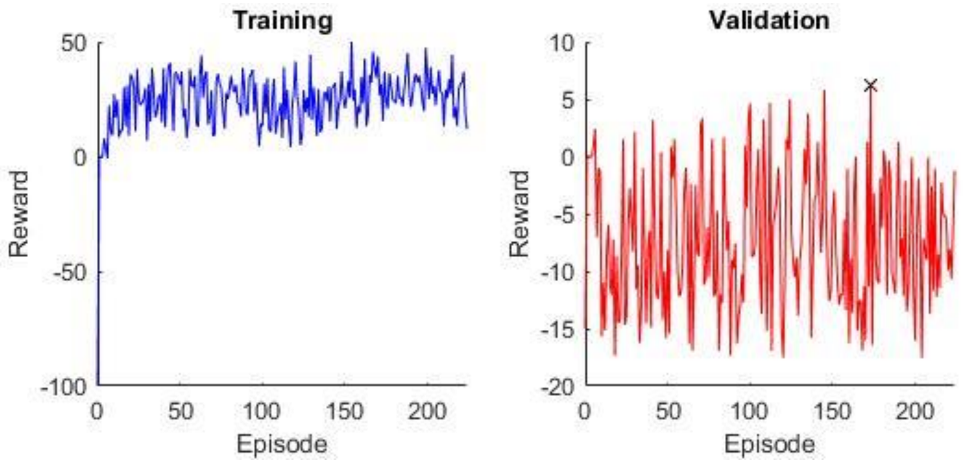

Figure 6 Sample of learning curve in the pair QH-BLAND

4.2 Out-of-sample results

The performance assessment in this section is responsible for the testing data. Table 6, Table 7, and Table 8 show the strategies comparison based on the determined indicators. The proposed method produces successful results on the pair STEC-CK. The trained agents solve the patterns in the input features and substantially surpass the optimal FTB strategy. For return analysis, the proposed method notably underperforms the FTB strategy on the pairs QH-LH, TISCO-TCAP, UNIQ-PREP, LH-BLAND, and QH-BLAND while it is observed slightly outperforming on the INTUCH-ADVANC. The profitability of the trained agents on these pairs is not reliable in the testing period. Despite the profit maximization objective, the optimized trading models still underperform the FTB strategy in overall profitability. For the return and stability trade-off measure, the return generated by the trained agent on most pairs is less stable than the FTB, resulting in a lower Sharpe ratio. For pairs QH-LH, TISCO-TCAP, UNIQ-PREP, the trained agents produce a negative Sharpe ratio since the generated returns do not overcome the risk-free rate. For downside analysis, the trained agents provide a lower maximum drawdown on the INTUCH-ADVANC, STEC-CK and QH-BLAND. In the other pairs, the trained agents suffer a higher drawdown because of losing trades caused by out-of-sample inaccuracy. The number of trades represents trade counting on a stock pair, with each trade encompassing the opening and closing positions. Table 9 presents the number of trades where the number of trades executed by trained agents is much greater than that by the FTB strategy. The trained agent behaves like a heavy trader, characterized by a tendency to close positions early and open new positions frequently.

Table 6 Return comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	2.53%	2.57%	4.75%	5.53%	**5.72%**
TISCO-TCAP	**12.06%**	11.63%	10.53%	8.31%	-2.82%
UNIQ-PREB	8.83%	**10.22%**	2.94%	4.25%	0.06%
STEC-CK	6.40%	2.53%	2.66%	4.27%	**16.23%**
LH-BLAND	7.91%	6.56%	6.89%	**9.95%**	4.66%
QH-LH	4.90%	2.12%	3.01%	**5.48%**	-1.68%
QH-BLAND	**9.98%**	8.69%	7.00%	7.19%	0.20%
Average	**6.80%**	5.84%	4.90%	5.93%	3.20%

Table 7 Sharpe ratio comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	0.34	0.36	0.68	**0.84**	0.58
TISCO-TCAP	1.33	1.39	1.40	**1.75**	-0.90
UNIQ-PREB	1.42	2.01	0.97	**2.02**	-0.36
STEC-CK	0.86	0.37	0.42	0.70	**2.24**
LH-BLAND	1.02	1.01	1.06	**2.01**	0.50
QH-LH	0.53	0.34	0.50	**1.09**	-0.77
QH-BLAND	1.52	1.45	1.40	**1.97**	-1.85
Average	1.00	0.99	0.92	**1.48**	-0.08

Table 8 Maximum drawdown comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	5.05%	5.05%	5.05%	5.05%	**4.31%**
TISCO-TCAP	6.54%	5.49%	5.14%	**2.32%**	8.77%
UNIQ-PREB	4.06%	3.43%	3.43%	**1.12%**	3.82%
STEC-CK	3.98%	3.98%	3.89%	3.89%	**2.35%**
LH-BLAND	6.73%	6.14%	6.14%	**3.45%**	4.49%
QH-LH	4.86%	4.97%	4.01%	**2.04%**	3.60%
QH-BLAND	5.47%	5.47%	5.47%	4.07%	**0.66%**
Average	5.24%	4.93%	4.88%	**3.16%**	4.00%

Table 9 Number of trades comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	4	3	3	3	38
TISCO-TCAP	11	7	5	3	14
UNIQ-PREB	7	5	1	1	35
STEC-CK	11	5	4	3	50
LH-BLAND	8	4	3	3	20
QH-LH	12	9	5	3	45
QH-BLAND	8	4	2	2	10
Average	8.71	5.29	3.29	2.57	30.29

Table 10 displays the winning rate which is a percentage proportion of the number of profitable trades divided by the total number of trades. Table 11 complements by presenting the average winning return and average losing returns. These metrics are calculated based on the returns achieved from profitable and losing trades respectively. Figure 7-13 visualizes a decision-making process and the resulting cumulative returns over time. In the trading signal charts, the green arrow indicates the opening of a position in the same direction as the cointegration coefficients, the red arrow represents an investment in the opposite direction to the cointegration coefficients, and the blue circle denotes the closing of positions. Based on the research findings, the profitable result on the pair STEC-CK has a win rate of 74% while the other pairs are observed to be close to 60%. Considering the magnitude of the returns, we find that the average positive returns cover losses in the pairs STEC-CK and INTUCH-ADVANC. On the other hand, average positive returns slightly exceed the average negative returns on the pairs UNIQ-PREP, LH-BLAND, and QH-BLAND. The magnitude of the returns indicates that the losses incurred from the losing trades are larger than the gains from the winning trades in the pairs QH-LH and TISCO-TCAP.

Table 10 Winning rate comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	**66.67%**	**66.67%**	**66.67%**	**66.67%**	57.89%
TISCO-TCAP	**90.00%**	85.71%	80.00%	66.67%	57.14%
UNIQ-PREB	**100.00%**	**100.00%**	**100.00%**	**100.00%**	57.14%
STEC-CK	**90.00%**	75.00%	66.67%	50.00%	74.00%
LH-BLAND	85.71%	**100.00%**	**100.00%**	**100.00%**	60.00%
QH-LH	72.73%	62.50%	80.00%	**100.00%**	60.00%
QH-BLAND	85.71%	**100.00%**	**100.00%**	**100.00%**	50.00%
Average	**84.40%**	79.51%	80.00%	78.57%	59.45%

Table 11 Average positive and negative return comparison in the testing period

Stock pair	FTB(0.5)	FTB(1.0)	FTB(1.5)	FTB(2.0)	Proposed
INTUCH-ADVANC	2.38% | -2.60%	2.57% | -2.60%	3.76% | -2.60%	**4.07%** | -2.60%	0.54% | **-0.44%**
TISCO-TCAP	1.84% | -4.15%	2.42% | -2.88%	3.09% | -1.85%	**4.24% | -0.18%**	0.90% | -1.66%
UNIQ-PREB	1.77% | **0.00%**	2.61% | **0.00%**	2.94% | **0.00%**	**4.25% | 0.00%**	0.47% | -0.44%
STEC-CK	1.09% | -1.97%	1.46% | -0.62%	1.85% | -0.62%	**2.17%** | -0.62%	0.58% | **-0.42%**
LH-BLAND	1.89% | -1.84%	2.22% | **0.00%**	2.30% | **0.00%**	**3.22% | 0.00%**	0.88% | -0.85%
QH-LH	0.99% | -1.23%	1.21% | -0.83%	1.13% | -1.50%	**1.83% | 0.00%**	0.33% | -0.59%
QH-BLAND	2.01% | -0.09%	2.21% | **0.00%**	3.50% | **0.00%**	**3.60% | 0.00%**	0.26% | -0.23%
Average	1.71% | -1.70%	2.10% | -0.94%	2.65% | -0.94%	**3.34% | -0.49%**	0.57% | -0.66%

Despite having a greater number of transactions and a short-term profit-taking, the proposed method fell short of achieving the same level of performance as the ordinary strategy. The proposed method suffered from a loss in modeling accuracy, whereas the static trading rules demonstrated a more accurate mean-reverting exploitation. The corresponding profitability and risk measurements show a difference in accuracy between the proposed method and the conventional strategy. Analyzing the optimized trading behavior reveals that most winning trades are a result of successive short-term exploitation. In the opposite side, notable losses are observed when the trained agent incorrectly predicts and holds positions for longer periods, particularly when the spread diverges.

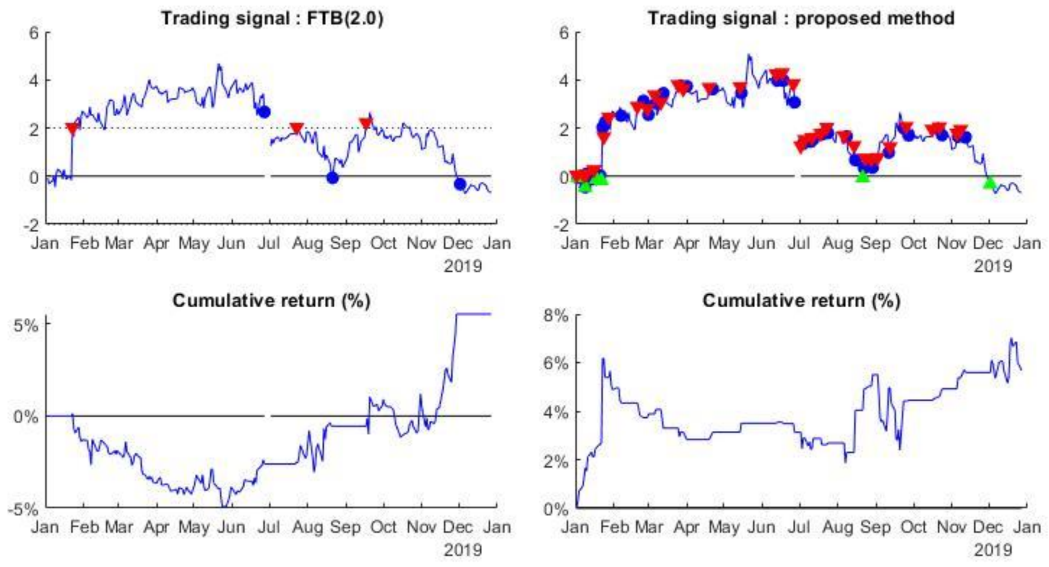

Figure 7 INTUCH-ADVANC trading chart during testing period

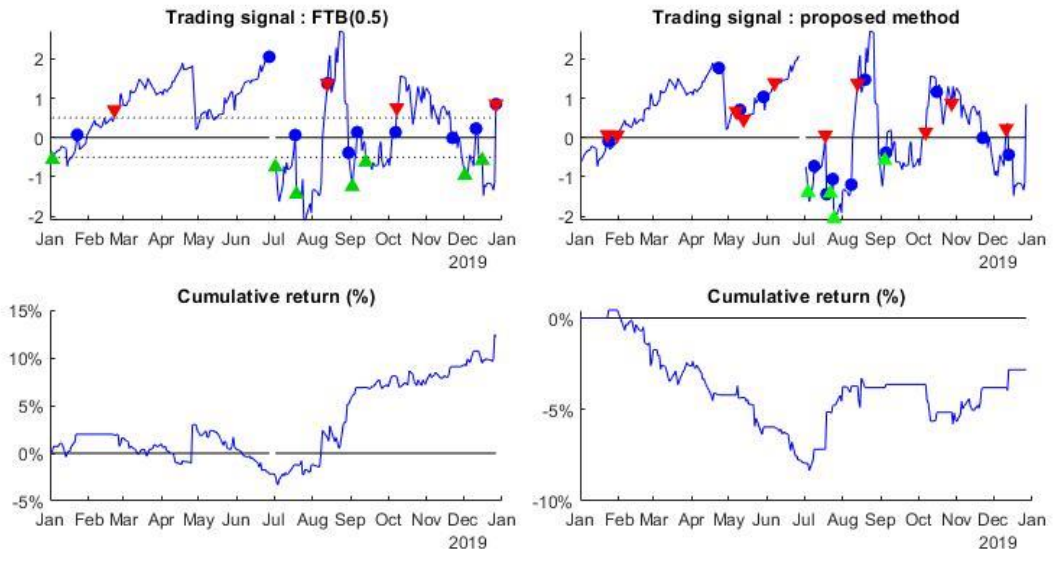

Figure 8 TISCO-TCAP trading chart during testing period

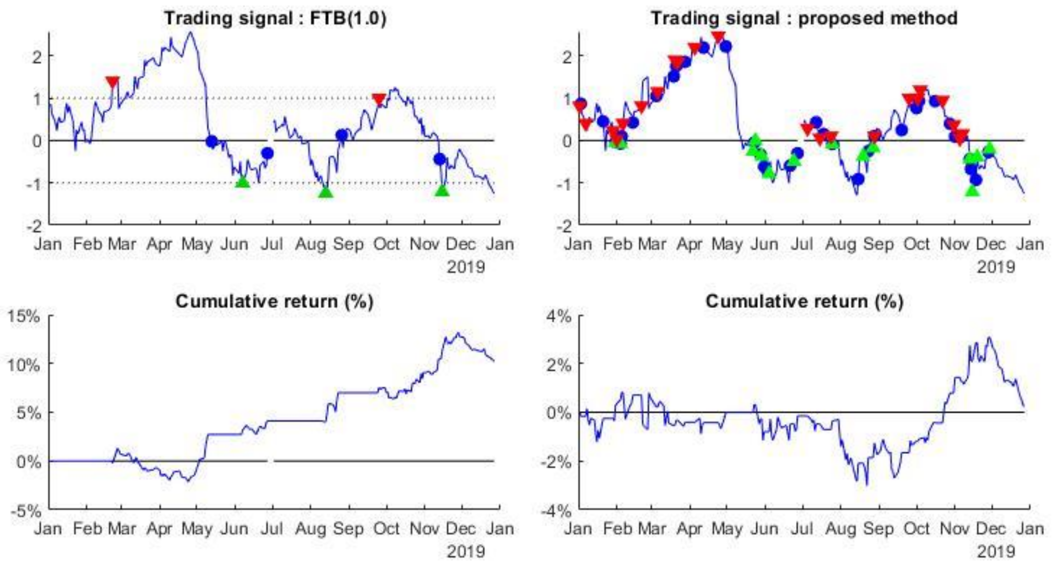

Figure 9 UNIQ-PREB trading chart during testing period

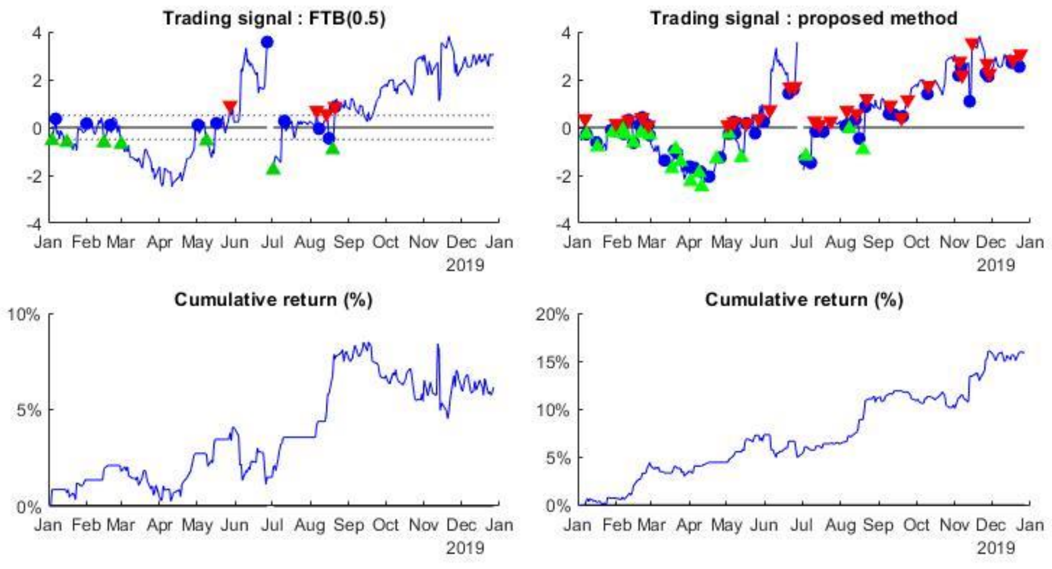

Figure 10 STEC-CK trading chart during testing period

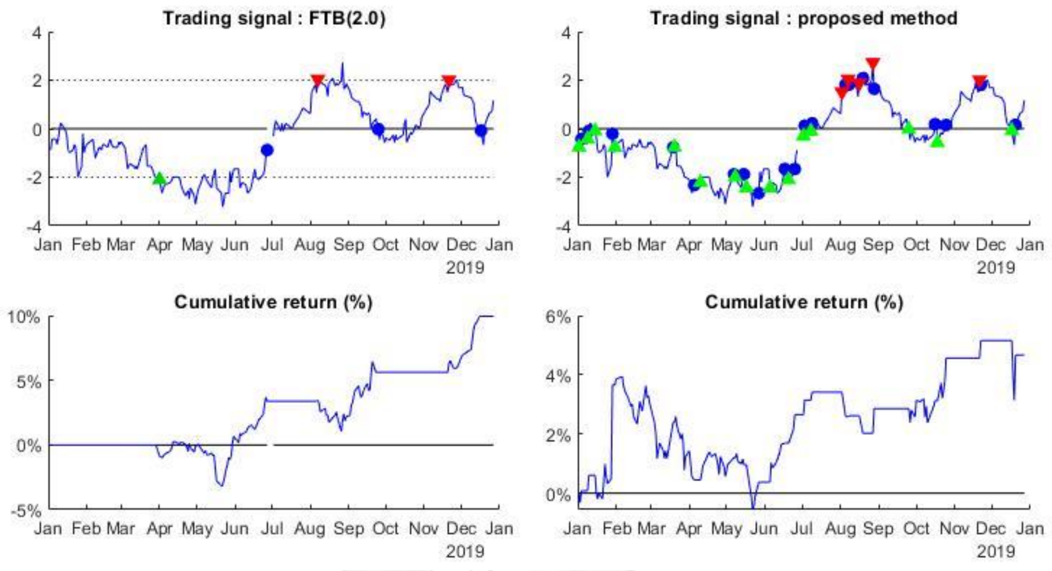

Figure 11 LH-BLAND trading chart during testing period

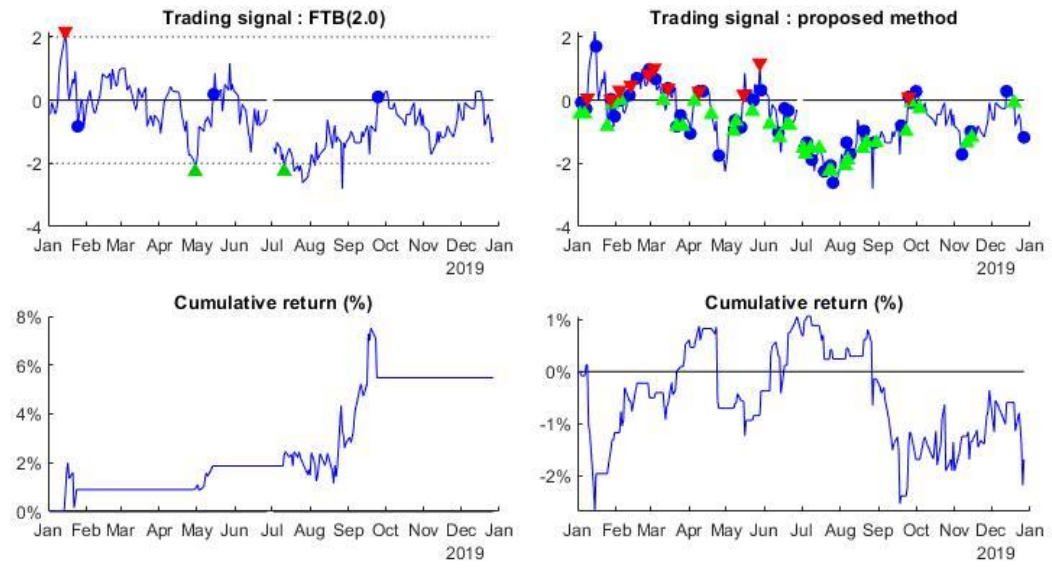

Figure 12 QH-LH trading chart during testing period

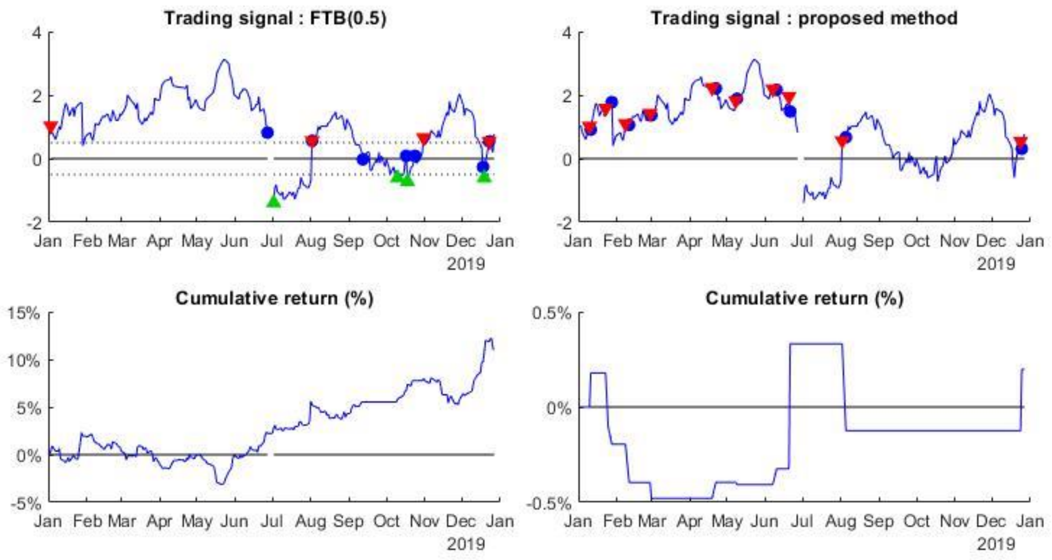

Figure 13 QH-BLAND trading chart during testing period

5. Conclusion

We proposed a pair trading optimization method using DQN. To improve the performance, we apply MODWT to the spread series. The wavelet extracted components represent the multi-timescale season series and long-term trend series. All components are used as input to the DQN. The experiments are carried out on the top seven correlated pairs of the stocks based on the Johansen cointegration test and Pearson correlation measure. In the training process, this model can solve a profit maximizing in the pair trading problem using wavelet inputs. since there is an increasing pattern in the cumulative reward for all training passages. The trained agents also be compared to a simple fixed trading boundary strategy using the out-of-sample data. On average, the trading performance of the proposed model underperforms the baseline method in terms of profitability and risk. Based on this model formulation, we have found that the yielding trading model faced an overfitting problem in both cross validation process and out-of-sample results. The accuracy of predictions becomes a critical risk due to the data-driven nature of this approach. For further improvement, it is necessary to improve the model generalization. We suggest conducting noise screening, an efficient feature selection, or cutting some high-frequency wave components which may relate to noises from the feature set. The model can be improved to be more precise in stock prediction by adding some additional related basic features. The formation window size and trading window size can be another set of hyperparameters to vary in the extended experiments. Finally, by incorporating a local risk measure into the reward function, both the risk and return dimensions can be optimized.

REFERENCES

Bozic, J., Babic, D., 2015. EUR/RSD exchange rate forecasting using hybrid wavelet-neural model: A case study. Computer Science and Information Systems 12, 5-5

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

Haar, A., 1910. Zur Theorie der orthogonalen Funktionensysteme. Mathematische Annalen 69, 331-371

He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026-1034

Hsieh, T.-J., Hsiao, H.-F., Yeh, W.-C., 2011. Forecasting stock markets using wavelet transforms and recurrent neural networks: An integrated system based on artificial bee colony algorithm. Applied Soft Computing 11, 2510-2525

Kao, L.-J., Chiu, C.-C., Lu, C.-J., Chang, C.-H., 2013. A hybrid approach by integrating wavelet-based feature extraction with MARS and SVR for stock index forecasting. Decision Support Systems 54, 1228-1244

Kim, S.-H.D.-Y.P., Ki-Hoon, L., Kim, S.-H.D.-Y.P., Ki-Hoon, L., 2022. Hybrid Deep Reinforcement Learning for Pairs Trading. Applied Sciences 12 2022, 944

Kim, T., Kim, H.Y., 2019. Optimizing the Pairs-Trading Strategy Using Deep Reinforcement Learning with Trading and Stop-Loss Boundaries. Complexity 2019, 3582516

Kingma, D., Ba, J., 2014. Adam: A Method for Stochastic Optimization. arXiv:1412.6980

Kobayashi, T., Ilboudo, W.E.L., 2021. t-soft update of target network for deep reinforcement learning. Neural Networks 136, 63-71

Lahmiri, S., 2014. Wavelet low- and high-frequency components as features for predicting stock prices with backpropagation neural networks. Journal of King Saud University - Computer and Information Sciences 26, 218-227

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., 2015. Human-level control through deep reinforcement learning. Nature 518, 529-533

Percival, D., Mofjeld, H., 1997. Analysis of Subtidal Coastal Sea Level Fluctuations Using Wavelets. Journal of the American Statistical Association 92, 868-880

Van Hasselt, H., Guez, A., Silver, D., 2015. Deep Reinforcement Learning with Double Q-Learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 2094–2100

VITA

NAME	Panudate Nithinon
DATE OF BIRTH	13 October 1995
PLACE OF BIRTH	Bangkok
INSTITUTIONS ATTENDED	Chulalongkorn University
HOME ADDRESS	111/32 Thanakorn 2 Villa TerdPrakiet road Watchalor Bangkruai Nonthaburi 11130