An Automated Portfolio Trading System with Feature Preprocessing and Recurrent Reinforcement Learning

Lin Li
Southern University of Science and Technology
Shenzhen, China
linleeccnu@gmail.com

ABSTRACT

We propose a novel portfolio trading system, which contains a feature preprocessing module and a trading module. The feature preprocessing module consists of various data processing operations, while in the trading part, we integrate the portfolio weight rebalance function with the trading algorithm and make the trading system fully automated and suitable for individual investors, holding a handful of stocks. The data preprocessing procedures are applied to remove the white noise in the raw data set and uncover the general pattern underlying the data set before the processed feature set is inputted into the trading algorithm. Our empirical results reveal that the proposed portfolio trading system can efficiently earn high profit and maintain a relatively low drawdown, which clearly outperforms other portfolio trading strategies.

CCS CONCEPTS

• **Computing methodologies** → **Reinforcement learning**; • **Applied computing** → **Economics**; **Multi-criterion optimization and decision-making**.

KEYWORDS

portfolio trading, recurrent reinforcement learning, automated trading systems, feature preprocessing

ACM Reference Format:
Lin Li. 2021. An Automated Portfolio Trading System with Feature Preprocessing and Recurrent Reinforcement Learning. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Portfolio trading usually aims to maximize the return over time and minimize the investment risk simultaneously. Investors typically gain profits by dynamically allocating their wealth among selected assets at the initial period and rebalancing their wealth afterwards. With the fast development of machine learning in recent years, portfolio trading has been extensively studied in the machine learning community. [18, 19] proposed to use recurrent reinforcement learning (RRL) algorithms to optimize trading systems and the algorithm was extended to optimize portfolios consisting of the S&P 500 Stock Index and T-Bills [17]. Afterwards, other researchers followed their scheme with certain variations. [8] applied RRL to the strategic asset allocation on samples from various countries. They only consider a portfolio consisting of 2 assets, long-term bond and equity and let the portfolio weight of one asset w_t depend on the parameters of the RRL algorithm by a heuristic equation in the economic context. [15] constructed a portfolio consisting of 12 stocks and let the position of each stock, long (1) or short (-1), be decided by the trading signal from each stock trading system, which was driven by the proposed regime-switching RRL. Also, [2] combined the RRL trading algorithm with the Calmar ratio to do the portfolio rebalance. They used the RRL algorithm to optimize the Calmar ratio instead of Sharpe ratio to generate the trading signals or positions of each asset, then the portfolio weight was decided by an exogenous softmax function based on the trading signal of each asset. On the other hand, online learning has also been broadly applied to select portfolios. [13] proposed passive aggressive mean reversion strategy, which constructed portfolios to minimize the deviation from the last portfolio. [12] exploited the mean reversion and variance information of portfolios and proposed the confidence weighted mean reversion strategy. Additionally, [11] proposed the on-line portfolio selection with moving average reversion (OLMAR) strategy, which used the moving average reversion pattern of stock price relatives. A notable difference between the portfolio selection algorithm driven by RRL and online learning is that online learning techniques allow more assets to be included in a portfolio than the reinforcement learning algorithm does. Other machine learning techniques and models have also been applied in portfolio selection. For instance, [3] developed a novel portfolio selection approach using a machine learning algorithm, eXtreme Gradient Boosting with an improved firely algorithm, and the mean-variance portfolio choice theory [16]. A mixed portfolio formation approach was proposed in [22], which used the long-short term memory networks to preselect stocks with high potential return and then the mean-variance model was applied to do the portfolio construction with the chosen stocks. Additionally, support vector machine and mean-variance model were combined to select portfolios in [5].

Amongst these machine learning portfolio selection models, we are particularly interested in RRL models, because it lays the ground for automated trading where the trading signals of each asset over the investment horizon are directly generated and its characteristics make it more suitable for individual investors. However, all these previous work about portfolio selection or trading mainly focus on building various machine learning algorithms to boost the model performance. To the best of our knowledge, few work has been dedicated to improving the feature quality before implementing the

Conference'17, July 2017, Washington, DC, USA
© 2021 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

trading algorithms. In this paper, we build an automated portfolio selection system, named PCA&DWT RRL, based on RRL and two main data preprocessing approaches, Principal Component Analysis (PCA) and Discrete Wavelet Transform (DWT). The PCA&DWT RRL system is driven by the RRL algorithm that aims to maximize the risk-adjusted return and contains some auxiliary components for feature processing. Specifically, the data set containing the daily close price and volume from each stock is firstly preprocessed to extract various technical indicators which are then further preprocessed by PCA and DWT techniques to form the feature series for each stock. Then the daily feature streams of various indicators are combined into one dataframe for each stock. Then for each period, the daily feature vectors of various stocks are concatenated as one element of the final feature set which is inputted into the RRL algorithm to generate portfolio weights at each trading period.

Our contributions are: 1, we embed the portfolio rebalance function into the RRL trading algorithm and the adjustment of portfolio weight is internally automated, which differentiates our approach from other portfolio trading methods based on RRL. 2, we integrate several data preprocessing steps with the trading algorithm, which ensures a high quality of the feature set for trading. 3, we have done extensive experiments to test the performance of PCA&DWT RRL system and the empirical results demonstrate that the proposed system consistently outperforms other benchmark portfolio selection strategies. Moreover, we find that feature preprocessing plays a vital role in this setting.

2 DATA PREPROCESSING MODULE

2.1 Data Configuration Layer

In the financial market, the price of stocks is affected by multiple factors which include but not limited to unexpected events, government policies and company activities. This implies that there may be a few periods when trading activities are not available, which results in a couple of NA entries in the price time series. These NAs are useless for our trading system and can be removed directly or filled out with other values subject to specific learning task. In our case, we discard these NAs directly for simplicity. Note that removing NA entries may result in irregular samples in timeline for different stocks. Hence, here we make sure the length of sample points is the same for all stocks and assume they are of the same timeline.

Technical analysis in the financial market has a long history amongst investment practitioners. However, it was omitted by academics over a few decades in the past probably due to the belief of efficient market hypothesis. As more and more evidence shows that markets are not as efficient as once believed. Technical analysis were applied to algorithmic trading [4, 20]. It is generally believed that technical analysis indicators can summarize the general pattern of the time series and avoid local noise in the data stream, which can further be utilized by the trading system to make profitable decisions. There are quite a few technical indicators developed by financial professionals [1]. One can choose various technical indicators to use depending on specific tasks. While without enough number of indicators, it may be tough to reveal the pattern of the data stream comprehensively, including too many technical indicators may also affect the trading decision negatively and increase the computation burden, since the calculated value of some technical indicators are not always consistent with one another. Given this, the indicators selected in this paper can be categorized into four groups with the hope to reflect the overall history information of specific stocks, momentum indicators: Momentum (MOM), Moving Average Convergence Divergence (MACD), Money Flow Index (MFI), Relative Strength Index (RSI), volatility indicators: Average True Range (ATR), Normalized Average True Range (NATR), cycle indicators: Hilbert Transform Dominant Cycle Phase (HTDCP), Hilbert Transform Sinewave (HTS), Hilbert Transform Trend Market Mode (HTTMM) and volume indicators: Chaikin Oscillator (CO), On Balance Volume (OBV). We select these technical indicators due to Let TA = {MOM, MACD, MFI, RSI, ATR, NATR, HTDCP, HTS, HTTMM, CO, OBV} denote the set of 11 technical indicators used for the following exposition. The calculation of technical indicators was done via the python library TA-Lib [6].

Additionally, to make the value of each feature on the same scale, we normalize each technical indicator stream with the standard normalization, i.e. z-score to process each technical indicator series, which is represented by the following equation.

$$X = \frac{X - \mu(X)}{\sigma(X)} \tag{1}$$

where X is the time series of each extracted feature, $\mu(X)$ and $\sigma(X)$ is the mean and standard deviation of X, respectively.

2.2 Principal Component Analysis Layer

The PCA layer of data preprocessing module receives the normalized feature set, including 11 technical indicators data streams, as input. The development of PCA originates from the curse of dimensionality which claims that data points in high dimension lie far away from each other, statistically speaking [7]. The curse of dimensionality not only makes the training of machine learning algorithm expensive, but also casts a shadow over the predictions of the trained algorithm, since data points in-sample and out-of-sample are so far away. In order to alleviate the effect of curse of dimensionality, a natural way is to reduce the dimension of data sets. That is where PCA comes into effect. To be more specific, PCA firstly fits the input, identifying the main components that represent the directions of maximum variance of the input. Then the main components are ordered according to the variance they explain and one can choose how many components are preserved. Afterwards, the original input is projected onto the retained components, resulting in a data set of lower dimension. In our case, the normalized technical indicators is a 11 dimensional data set at first. After being decomposed by PCA, there will be less than 11 indicators, reducing the probability of correlation and inconsistency amongst different technical indicators. In this way, the processed TA set results in a new $TA' \subset TA$. Note that different stocks may have different components of technical indicators retained in TA'. Here we only need to make sure that the cardinality of TA's is the same for all stocks to meet the logic of the RRL trading algorithm afterwards. We use the well developed Scikit-learn [21] library to implement the PCA operation. Additionally, we set the hyperparameter, explained variance ratio, to 95% which means that the sum of the variance explained by all retained principal components takes up at least 95% of the total variance of the original data set.

2.3 Discrete Wavelet Transform Layer

Although the data set returned by the PCA layer is simplified by removing less relevant features in the feature domain, some outliers or irrelevant data points, representing local noise, may still exist in each feature series. They may affect the training and trading of the RRL algorithm. To remove these local noise in the time domain of each indicator, we apply the discrete wavelet transform to the feature data after being processed by PCA.

Reference [14] proposed to calculate the DWT coefficients, including the approximation and the detail coefficients, using a pair of high pass and low-pass filter. In the algorithm, the father $\Phi(t)$ and mother $\Psi(t)$ basis functions are introduced to generate their corresponding son $\Phi_{j,k}(t)$ and daughter $\Psi_{j,k}(t)$ wavelets which are further utilized to approximate the original signal. The corresponding coefficients of son and daughter wavelets as a result of decomposing a function $f(x)$ is defined in the following inner product form [14].

$$a_{j,k} = \langle f(t), \Phi_{j,k}(t)\rangle, \quad d_{j,k} = \langle f(t), \Psi_{j,k}(t)\rangle \tag{2}$$

where $a_{j,k}$ and $d_{j,k}$ are approximation and detail coefficients, respectively, and $k = 0, 1, 2, \ldots$ and $j = 0, 1, 2, \ldots$. Though there are various types of wavelets, in this work, we use Haar wavelets with periodization padding mode, since Haar wavelets are useful to capture fluctuations between adjacent observations, recorded by [9], which would be heuristically useful to spot evident drawdowns in the financial market. Additionally, the decomposition can be iterated for multiple times, subject to the inherent decomposition level of the wavelets and the length of the signal or data series.

Eventually, the DWT leaves us one set of the approximation coefficients and a couple of sets of the detail coefficients depending on the given decomposition level. In this paper, we set the DWT decomposition level equal to 4, since too high decomposition level would destroy the general pattern, while too low would still leave too much noise in the data [10]. After the decomposition finishes, the general trend of the original signal is preserved in the approximation set, while the detail coefficients sets contain the local noise of the signal [14], which we aim to clean. At this point, we apply soft thresholding technique with an empirical threshold value equal to two times standard deviations of coefficients to each detail coefficients set. Eventually, the inverse DWT method is used to reconstruct the signal which is the final denoised version of the original signal. By discarding the irrelevant coefficients, the reconstructed signal represents the essential characteristics of the original signal, which is further adopted by the RRL trader in the following trading module. The DWT process for each technical indicator series of each asset is implemented with the open source python package PyWavelets [10]. Note that here parameters for DWT are kept the same for the PCA-transformed technical indicators.

3 RECURRENT REINFORCEMENT LEARNING FOR PORTFOLIO TRADING MODULE

3.1 Portfolio Rebalance Function

In this paper, we assume that the trader takes only long positions and there is no income or consumptions. At the beginning of each period, the trader rebalances the portfolio which is composed of several securities with corresponding weights. Assuming there are m securities with price series $\{\{p_t^a\} : a = 1, ..., m\}$, the market rate of return r_t^a for price series p_t^a for the period ending at time t is defined as $r_t^a = \frac{p_t^a}{p_{t-1}^a} - 1$ and thus the return vector of m securities is defined as $\boldsymbol{r}_t = [r_t^1, r_t^2, ..., r_t^m]^\top$. Defining portfolio weight of the a^{th} security at period t as F_t^a, $\boldsymbol{F}_t = [F_t^1, F_t^2, ..., F_t^m]^\top$ and the vector $\mathbf{1} = [1, 1, ..., 1]^\top$, then the trader that takes only long positions must have portfolio weights that satisfy:

$$\boldsymbol{F}_t^\top \cdot \mathbf{1} = 1, \quad \boldsymbol{F}_t \geq 0 \tag{3}$$

Given these conditions, we use the following normalized outputs as portfolio weights:

$$\boldsymbol{F}_t = \frac{\exp[\boldsymbol{f}_t(\boldsymbol{Y}_t)]}{\mathbf{1}^\top \cdot \exp[\boldsymbol{f}_t(\boldsymbol{Y}_t)]} \tag{4}$$

which is suggested by [18, 19] and $\boldsymbol{f}_t$ is defined using hyperbolic tangent activation function as:

$$\boldsymbol{f}_t(\boldsymbol{Y}_t) = \tanh(\boldsymbol{Y}_t) \tag{5}$$

$$\boldsymbol{Y}_t = (\boldsymbol{X}_t \otimes \boldsymbol{\Theta}) \cdot \mathbf{1} \tag{6}$$

where $\boldsymbol{X}_t = [\boldsymbol{x}_t^1, \boldsymbol{x}_t^2, ..., \boldsymbol{x}_t^m]^\top$ is the input feature matrix to the trading system, while $\boldsymbol{\Theta} = [\boldsymbol{\theta}^1, \boldsymbol{\theta}^2, ..., \boldsymbol{\theta}^m]^\top$ is the system parameter matrix to be learned during the training process, and $\otimes$ represents the element-wise product between matrices. Note that all elements of matrices $\boldsymbol{X}_t$ and $\boldsymbol{\Theta}$ are vectors per se, in particular, $\boldsymbol{x}_t^a = [1, ta_t^1, ta_t^2, \ldots, ta_t^n, F_{t-1}^a]$, where $ta_t^1, ta_t^2 \ldots, ta_t^n$ are the value of technical indicators remained in TA' at period t for security a and n is the cardinality of the TA' set.

3.2 Profit of Portfolio Tradings

Since F_t^a represents the holdings of security a at period t, then F_t^a should be re-adjusted at each time step according to Equation 4. Thus, generally speaking, a transaction cost rate δ should be applied to each security weight adjustment between two consecutive periods. One can arguably expect that higher transaction costs would discourage the excessive rebalance actions [17]. Since our focus is the systematic construction of the trading system, we here assume the influence of price series movements on portfolio weights is negligible for the ease of exposition and analysis. In this case, the wealth of the trading at time T is:

$$\begin{aligned} W_T &= W_0 \prod_{t=1}^{T} (1 + R_t) \\ &= W_0 \prod_{t=1}^{T} (1 + \boldsymbol{F}_{t-1}^\top \boldsymbol{r}_t)(1 - \delta \cdot \mathbf{1}^\top |\boldsymbol{F}_t - \boldsymbol{F}_{t-1}|) \end{aligned} \tag{7}$$

where W_0 is the initial wealth of the investment account, which we set as \$1 for simplicity, and R_t is the return of the portfolio at time t i.e. one-stage profit, defined as:

$$R_t = (1 + \boldsymbol{F}_{t-1}^\top \boldsymbol{r}_t)(1 - \delta \cdot \mathbf{1}^\top |\boldsymbol{F}_t - \boldsymbol{F}_{t-1}|) - 1 \tag{8}$$

In this case, the cumulative profit obtained from the investment after T periods is:

$$P_T = W_T - W_0 \tag{9}$$

3.3 Sharpe Ratio

In this paper, we aim to optimize the risk-adjusted return of the portfolio (e.g. Sharpe ratio) among other performance criteria, since it is relatively simple and widely adopted by various investors. The Sharpe ratio is commonly defined as the ratio between the average and standard deviation of a period of historical returns, $R_{1,\dots,T}$ [17], where R_t is the return of investment at trading period t. Intuitively, Sharpe ratio rewards investment strategies that are less volatile to make profits.

3.4 Gradient Ascent

To obtain the optimal portfolio rebalance strategy, the RRL algorithm needs to learn the optimal parameters via maximizing Sharpe ratio of the portfolio. Therefore, one needs to evaluate the influence of Sharpe ratio on the portfolio trading system during training. We attain this goal by computing the first order derivative of Sharpe ratio with respect to (w.r.t.) $\mathbf{\Theta}$. Furthermore, we adopt the gradient ascent to update the model parameters learned during training. Although automatic differentiation is easily available, we include technical details here for the reference of implementation, especially for readers with little background in machine learning.

First of all, with the estimate of the first and second moments of returns distributions, one has the Sharpe ratio formula of a portfolio as follows [17]:

$$S_T = \frac{E[R_{1,\dots,T}]}{\sqrt{E[R^2_{1,\dots,T}] - (E[R_{1,\dots,T}])^2}} = \frac{A}{\sqrt{B - A^2}} \tag{10}$$

Where $A = \frac{1}{T}\sum_{t=1}^{T} R_t$, $B = \frac{1}{T}\sum_{t=1}^{T}(R_t)^2$ and R_t is the return of the portfolio at time t. Then, the first order derivative of S_T w.r.t. the system parameters is computed using the chain rule:

$$\begin{aligned}\frac{dS_T}{d\mathbf{\Theta}} &= \frac{d}{d\mathbf{\Theta}}\left\{\frac{A}{\sqrt{B-A^2}}\right\}\\ &= \frac{\partial S_T}{\partial A}\cdot\frac{\partial A}{\partial \mathbf{\Theta}} + \frac{\partial S_T}{\partial B}\cdot\frac{\partial B}{\partial \mathbf{\Theta}}\\ &= \sum_{t=1}^{T}\left\{\frac{\partial S_T}{\partial A}\cdot\frac{\partial A}{\partial R_t} + \frac{\partial S_T}{\partial B}\cdot\frac{\partial B}{\partial R_t}\right\}\cdot\frac{dR_t}{d\mathbf{\Theta}}\\ &= \sum_{t=1}^{T}\left\{\frac{\partial S_T}{\partial A}\cdot\frac{\partial A}{\partial R_t} + \frac{\partial S_T}{\partial B}\cdot\frac{\partial B}{\partial R_t}\right\}\\ &\quad\cdot\left\{\mathrm{diag}(\frac{\partial R_t}{\partial \boldsymbol{F}_t})\frac{\partial \boldsymbol{F}_t}{\partial \mathbf{\Theta}} + \mathrm{diag}(\frac{\partial R_t}{\partial \boldsymbol{F}_{t-1}})\frac{\partial \boldsymbol{F}_{t-1}}{\partial \mathbf{\Theta}}\right\}\end{aligned} \tag{11}$$

where $\mathrm{diag}(\frac{\partial R_t}{\partial \boldsymbol{F}_t})$ and $\mathrm{diag}(\frac{\partial R_t}{\partial \boldsymbol{F}_{t-1}})$ stand for square matrices whose main diagonal entries are from vectors $\frac{\partial R_t}{\partial \boldsymbol{F}_t}$ and $\frac{\partial R_t}{\partial \boldsymbol{F}_{t-1}}$, respectively, and all other entries are 0. Note that the difference between the direct portfolio optimization as our method and single security automated trading optimization [17] is that here $\frac{dS_T}{d\mathbf{\Theta}}$ is no longer a vector, but a matrix. Since we are trading several risky assets simultaneously, this means $\boldsymbol{F}_t$ and $\boldsymbol{F}_{t-1}$ are both vectors, and $\mathbf{\Theta}$ is a matrix. The Jacobian matrices should be calculated for partial derivatives, $\frac{\partial R_t}{\partial \boldsymbol{F}_t}$, $\frac{\partial \boldsymbol{F}_t}{\partial \mathbf{\Theta}}$, $\frac{\partial R_t}{\partial \boldsymbol{F}_{t-1}}$ and $\frac{\partial \boldsymbol{F}_{t-1}}{\partial \mathbf{\Theta}}$. To be more specific:

$$\begin{aligned}\frac{\partial R_t}{\partial \boldsymbol{F}_t} &= \frac{\partial}{\partial \boldsymbol{F}_t}\left\{(1+\boldsymbol{F}_{t-1}^{\top}\boldsymbol{r}_t)(1-\delta\cdot\mathbf{1}^{\top}|\boldsymbol{F}_t-\boldsymbol{F}_{t-1}|)-1\right\}\\ &= -\delta\cdot(1+\boldsymbol{F}_{t-1}^{\top}\boldsymbol{r}_t)\cdot\mathrm{sgn}(\boldsymbol{F}_t-\boldsymbol{F}_{t-1})\end{aligned} \tag{12}$$

$$\begin{aligned}\frac{\partial R_t}{\partial \boldsymbol{F}_{t-1}} &= (1-\delta\cdot\mathbf{1}^{\top}|\boldsymbol{F}_t-\boldsymbol{F}_{t-1}|)\boldsymbol{r}_t\\ &\quad+\delta\cdot(1+\boldsymbol{F}_{t-1}^{\top}\boldsymbol{r}_t)\cdot\mathrm{sgn}(\boldsymbol{F}_t-\boldsymbol{F}_{t-1})\end{aligned} \tag{13}$$

$$\begin{aligned}\frac{\partial \boldsymbol{F}_t}{\partial \mathbf{\Theta}} &= \frac{\partial \boldsymbol{F}_t}{\partial \boldsymbol{f}_t}\cdot\frac{\partial \boldsymbol{f}_t}{\partial \boldsymbol{Y}_t}\cdot(\frac{\partial \boldsymbol{Y}_t}{\partial \mathbf{\Theta}} + \frac{\partial \boldsymbol{Y}_t}{\partial \boldsymbol{F}_{t-1}}\cdot\frac{\partial \boldsymbol{F}_{t-1}}{\partial \mathbf{\Theta}})\\ &= \boldsymbol{DF}_t(\boldsymbol{f}_t)\cdot\boldsymbol{Df}_t(\boldsymbol{Y}_t)\cdot(\boldsymbol{X}_t + \mathrm{diag}(\boldsymbol{\theta}'^{\top})\frac{\partial \boldsymbol{F}_{t-1}}{\partial \mathbf{\Theta}})\end{aligned} \tag{14}$$

Note that here we choose $\boldsymbol{f}_t$ in the form of tanh function, the logistic function form of $\boldsymbol{f}_t$ can also be easily calculated. Moreover, $\boldsymbol{DF}_t(\boldsymbol{f}_t)$ and $\boldsymbol{Df}_t(\boldsymbol{Y}_t)$ are the Jacobian matrices of $\boldsymbol{F}_t$ w.r.t. $\boldsymbol{f}_t$ and $\boldsymbol{f}_t$ w.r.t. $\boldsymbol{Y}_t$, respectively, and $\boldsymbol{\theta}'$ is the vector of the last column of parameter matrix $\mathbf{\Theta}$, corresponding to $\boldsymbol{F}_{t-1}$ in the feature matrix, $\boldsymbol{X}_t$.

$$\boldsymbol{DF}_t(\boldsymbol{f}_t) = \begin{bmatrix}\frac{\partial F_t^1}{\partial f_t^1} & \frac{\partial F_t^1}{\partial f_t^2} & \cdots & \frac{\partial F_t^1}{\partial f_t^m}\\ \frac{\partial F_t^2}{\partial f_t^2} & \ddots & \frac{\partial F_t^i}{\partial f_t^j} & \cdots\\ \vdots & \vdots & \vdots & \vdots\\ \frac{\partial F_t^m}{\partial f_t^1} & \frac{\partial F_t^m}{\partial f_t^2} & \cdots & \frac{\partial F_t^m}{\partial f_t^m}\end{bmatrix} \tag{15}$$

Define $S_i = \frac{\exp(f_t^i)}{\sum_i \exp(f_t^i)}$, then entries of the Jacobian $\boldsymbol{DF}_t(\boldsymbol{f}_t)$ can be simplified as:

$$\frac{\partial F_t^i}{\partial f_t^j} = \begin{cases} S_i(1-S_j) & \text{if} \quad i=j\\ -S_iS_j & \text{if} \quad i\neq j\end{cases} \tag{16}$$

Similarly, the Jacobian $\boldsymbol{Df}_t(\boldsymbol{Y}_t)$ is calculated as follows:

$$\boldsymbol{Df}_t(\boldsymbol{Y}_t) = \begin{bmatrix}\frac{\partial f_t^1}{\partial Y_t^1} & 0 & \cdots & 0\\ 0 & \frac{\partial f_t^2}{\partial Y_t^2} & \ddots & 0\\ \vdots & \vdots & \vdots & \vdots\\ 0 & 0 & \cdots & \frac{\partial f_t^m}{\partial Y_t^m}\end{bmatrix} \tag{17}$$

where

$$\frac{\partial f_t^i}{\partial Y_t^i} = 1 - \tanh^2(Y_t^i) \tag{18}$$

It is straightforward that the derivative $\frac{\partial \boldsymbol{F}_t}{\partial \mathbf{\Theta}}$ is recurrent and depends on all its previous value within the time window T. This is also the reason why this reinforcement learning method is recurrent. Once the term $\frac{dS_T}{d\mathbf{\Theta}}$ has been calculated, the system parameters $\mathbf{\Theta}$ is updated according to the gradient ascent rule with consideration of the ℓ_2 regularization to avoid overfitting the noise in the data,

$$\begin{aligned}\mathbf{\Theta}_{n+1} &= \mathbf{\Theta}_n + \rho\cdot(\frac{dS_T}{d\mathbf{\Theta}_n} - \lambda\cdot\mathbf{\Theta}_n)\\ &= (1-\rho\cdot\lambda)\cdot\mathbf{\Theta}_n + \rho\cdot\frac{dS_T}{d\mathbf{\Theta}_n}\end{aligned} \tag{19}$$

where $\rho, \lambda \geq 0$ are the given learning rate and ℓ_2 regularization hyperparameter. The process is repeated for N epochs, where N can be chosen such that Sharpe ratio has converged during training.

Note that optimizing Sharpe ratio requires to calculate the gradient of Sharpe ratio w.r.t. $\boldsymbol{\Theta}$ which depends on the total derivative of $\frac{\partial F_t}{\partial \Theta}$. Therefore, we adopt an efficient recurrent algorithm similar to backpropagation through time (BPTT) to train the model, as in [17] for single stock trading using RRL.

3.5 Algorithm

Algorithm 1: Training of the PCA&DWT RRL.

```
1  Input: $\boldsymbol{X}_t$: Feature matrix; $\boldsymbol{r}_t$: Stocks return vector; $T$:
     Training window size; $\rho$: Learning rate; $\lambda$: Parameter for $\ell_2$
     regularization; $\delta$: Transaction cost rate; $N$: The number of
     epochs; $\kappa$: Random seed for generation of initial portfolio
     weight; $\epsilon$: Iteration stopping threshold;
2  Output: $\boldsymbol{\Theta}_n^*$: Optimized system parameter matrix;
3  Procedure: Initialization: $\boldsymbol{\Theta}_0 \longleftarrow \mathcal{N}(0,1)$, $\boldsymbol{F}_0 = \boldsymbol{0}$, $\frac{\partial F_0}{\partial \Theta_0} = \boldsymbol{0}$,
     $W_0 = 1$;
4  for $n = 0, 1, ..., N-1$ do
5      for $t = 1, ..., T-1$ do
6          Receive feature matrix: $\boldsymbol{X}_t$, stocks return vector: $\boldsymbol{r}_t$;
7          Calculate $\boldsymbol{Y}_t = (\boldsymbol{X}_t \otimes \boldsymbol{\Theta}_n) \cdot \boldsymbol{1}$;
8          Calculate $\boldsymbol{f}_t = \tanh(\boldsymbol{Y}_t)$;
9          Calculate $\boldsymbol{F}_t = \text{softmax}(\boldsymbol{f}_t)$;
10         Calculate $R_t$, $\frac{\partial R_t}{\partial F_t}$, $\frac{\partial R_t}{\partial F_{t-1}}$ and $\frac{\partial F_t}{\partial \Theta_n}$, respectively;
11     end for
12     Calculate Sharpe ratio: $S_T^n$;
13     if $n \geq 2$ and $|S_T^n - S_T^{n-1}| \leq \epsilon$ then
14         Stop iteration;
15     end if
16     Calculate $\frac{dS_T^n}{d\Theta_n}$;
17     Update weight according to:
         $\Theta_{n+1} = (1 - \rho \cdot \lambda) \cdot \Theta_n + \rho \cdot \frac{dS_T^n}{d\Theta_n}$;
18 end for
```

Based on what we discussed before, we design Algorithm 1 to train our model, which aims to obtain the optimized system parameter matrix $\boldsymbol{\Theta}_n^*$. Then, $\boldsymbol{\Theta}_n^*$ is directly applied to the same algorithm within one training epoch with the test/trading window size M, which gives us the out-of-sample result of the PCA&DWT RRL method. Afterwards, the training and trading processes are repeated forward until the last batch of trading periods. A graphical representation of the rolling training and test is given in Figure 1. When training, the agent is trained for N epochs and the process is early stopped if the objective value is not improved in two consecutive epochs. This is also to avoid overfitting of the RRL algorithm and ensure a better generalization ability of the algorithm. Note the value of T and M can be fine-tuned to fit the market structure underlying different stock price patterns, which should improve the performance theoretically, if there is a similar market pattern in the rolling training and trading windows.

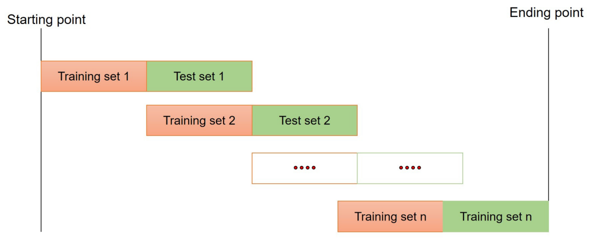

Figure 1: The rolling training and test processes of the RRL algorithm for portfolio trading.

4 EXPERIMENTS

4.1 Data Sets

We test the proposed portfolio trading system on real data sets composed of different number of stocks. As to the number of stocks in a portfolio, we notice that many studies considered a portfolio with less than ten stocks [2, 3, 22] and holding too many different stocks is tough to manage for individual investors. [5] argued that a portfolio with 7 assets is more appropriate than others for portfolio selection under the machine learning context. Therefore, we construct portfolios with cardinality $k = 4, 5, 6, 7, 8$ using different stocks as different data sets which are then inputted to the trading algorithm to uncover sequential portfolio weights. Specifically, we randomly choose 8 different stocks with ticker symbols XOM, VZ, NKE, AMAT, MCD, MSFT, AAP and NOV from S&P500 index which is arguably representative of the general stock market condition in the US, then we form portfolios with different cardinalities using these stocks. Note that due to the RRL trading algorithm where Jacobian matrices are calculated, the order of different stocks in a portfolio matters. However, our empirical results find that the effect of stock orders is not significant. Therefore, we form portfolios with the listed stocks order one by one without loss of generality. For each stock, five data streams are collected from Yahoo Finance [1], which consist of the daily prices (Open, High, Low, Close) and Volume over the period of 31/12/2009 to 29/12/2017. We allow each stock price series to exhibit a unique behavior or pattern to ensure the universality of the data sets. Table 1 exhibits the summary statistics of the close prices for the 8 stocks.

4.2 Performance Metrics

The metrics used to measure the performance of the proposed portfolio trading system in the real financial market are: Net Profit (NP) which is final wealth W_T accumulated by the RRL trading over all the trading periods minus the initial wealth W_0; Annualized Percentage Yield (APY), representing the annualized percentage gain; Annualized Sharpe Ratio (ASR), representing the annualized risk adjusted return and we assume the risk-free return is 4% per annum and there are 252 trading days each year; Maximum Drawdown

[1]accessible from https://finance.yahoo.com/.

Table 1: Summary statistics for the selected stocks ($)

Symbols	Mean	Std.	Max.	Min.	Range
XOM	83.73	9.66	104.38	56.57	47.81
VZ	44.02	7.39	56.53	25.26	31.27
NKE	38.18	15.37	67.17	15.33	51.84
AMAT	20.26	10.69	58.80	9.85	48.95
MCD	101.25	23.53	174.20	61.45	112.75
MSFT	41.47	15.52	86.85	23.01	63.84
AAP	106.08	42.34	200.38	39.16	161.22
NOV	53.19	16.28	86.43	26.34	60.09

Table 2: Metrics used to measure the performance of different trading systems

NP	APY	ASR	MDD	CR
$W_T - W_0$	$(\frac{W_T}{W_0})^{\frac{252}{T}} - 1$	$\frac{\text{APY}-0.04}{\text{Std}(R_{1,\ldots,T})\cdot\sqrt{252}}$	$\frac{\text{PV}-\text{LV}}{\text{PV}}$	$\frac{\text{APY}}{\text{MDD}}$

(MDD) which measures the profit decline percentage from peak value (PV) before largest drop and lowest value (LV) before new high established of an investment during a specific period; Calmar Ratio (CR), indicating the level of risk taken to achieve a return and a higher CR suggests that the system's return is not at the risk of large drawdowns and vice verse. The higher the numerical value of all these metrics are, the better the performance of the trading system is except for MDD for which a lower value is preferred, since most investors are risk-averse. The definitions for these metrics are summarized in Table 2.

4.3 Benchmark Strategies

A way to show the efficiency of the proposed portfolio trading system would be to compare the performance of the system with other benchmark methods. In this paper, we compare the proposed trading method with the baseline, Uniform Constant Rebalanced Portfolios (UCRP), which rebalances to a uniform portfolio with equal weight of each stock at the beginning of every period [11]; OLMAR, which is representative of the notable online portfolio selection techniques in recent years [11]; the mean-variance portfolio selection model (MV), which utilizes the Monte Carlo method to generate different portfolios, that is, randomly create a set of weights and calculate the mean and variance of each portfolio under the weight, then choose the corresponding weight with the highest Sharpe ratio to allocate wealth for each period over the horizon [2, 3, 5, 22]. Note that the mean-variance portfolio choice theory [16] usually aims for single period portfolio selection, researchers sometimes assume that the obtained portfolio weight is optimal for the entire investment horizon in hindsight, which may be not true. We also compare the proposed method with the original RRL trading method (LAG RRL) which simply uses the lagged historical daily return of each stock as features plus a RRL trading module [17]. Furthermore, in order to evaluate the effect of using PCA and DWT techniques, we also present the results of the proposed method without PCA and DWT layers (TA RRL).

4.4 Hyperparameters

A common feature of most machine learning models is that their performance highly depends on the setting of hyperparameters which are parameters set before the training process begins. In our case, the performance of the trading system is similarly affected by hyperparameters from each module of the system. Besides the hyparameters described before in the data preprocessing module of the system, hyperparameters of the RRL trading module also matter. Furthermore, theoretically speaking, for each feature set, there is an optimal set of hyperparameters associated. However, due to multiple factors, such as the big number of hyperparameters, the large value space of each hyperparameter and the interdependence amongst different hyperparameters, determination of the optimal set of hyperparameters is almost impossible. Even if one could find the optimal set of hyperparameters for one data set, these hyperparameters is most likely to be sub-optimal for other data sets. Therefore, in this paper, the value of hyperparameters of the proposed portfolio trading system is set empirically to try to ensure an overall good performance on all data sets, unless otherwise stated. Specifically, besides the hyperparameter in the data preprocessing module, we empirically fix $T = 100, \rho = 0.1, \lambda = 0.01, \delta = 0.001, N = 100, \kappa = 42, \epsilon = 0$ and the rolling trading window size $M = 100$. Note that in terms of the transaction cost, this paper only considers brokerage cost as it is directly controlled by individual investors. Referring to parameter setting of several empirical research [2, 22], we decide to simulate the transaction cost as $\delta = 0.001$ or 10 bps without loss of generality for all listed strategies. Moverover, to ensure a fair comparison, we make the shared hyperparameters same for three RRL based strategies and the LAG RRL is additionally fine-tuned in its lagged length of return series, while OLMAR is also fine-tuned in its hyperparameters, reversion threshold and look-back window and we use Monte Carlo method to simulate 50,000 different sets of portfolio weights for the MV strategy, which would arguably cover most possible portfolios, from statistical point of view [3, 22].

4.5 Numerical Results

While UCRP and MV are passive investment strategies whose profits are mostly determined by the movement of the underlying assets' price, the other methods are active in taking actions in markets. The numerical results of all strategies' performance are presented in Table 3 with the best results obtained by all strategies on each portfolio stressed in bold face. Additionally, Figure 2 presents these numerical results with boxplots.

By examining the obtained results, it is apparent that the proposed trading system clearly outperforms all other strategies in term of almost all metrics on portfolios with different cardinalities. Specifically, the PCA&DWT RRL earns the highest NP, APY, ASR, CR and lowest MDD, except for the MDD of the portfolio containing 5 stocks, than others on various portfolios, which shows that this portfolio trading system is not only effective but superior to the benchmark in most cases. Secondly, PCA&DWT RRL substantially outperforms other two RRL based strategies, meaning that combining PCA and DWT techniques implemented on technical indicators plays an important role in improving the performance of the trading system. Especially, feature preprocessing is vital for RRL trading algorithm. Thirdly, it is found that UCRP and MV are

Table 3: Numerical value of the performance of different strategies on different portfolios

	k	UCRP	OLMAR	MV	LAG RRL	TA RRL	PCA&DWT RRL
NP							
	4	1.64	0.64	2.32	0.97	1.44	**3.17**
	5	1.60	0.54	1.87	0.48	1.02	**3.07**
	6	1.73	0.80	1.80	0.50	1.03	**3.02**
	7	1.62	1.19	1.75	0.55	0.86	**2.80**
	8	1.40	1.32	1.80	0.35	0.93	**2.43**
APY							
	4	0.15	0.07	0.19	0.10	0.13	**0.22**
	5	0.15	0.06	0.16	0.06	0.10	**0.22**
	6	0.15	0.09	0.16	0.06	0.10	**0.21**
	7	0.15	0.12	0.16	0.06	0.09	**0.21**
	8	0.13	0.13	0.16	0.04	0.10	**0.19**
ASR							
	4	0.71	0.19	0.85	0.38	0.59	**1.10**
	5	0.76	0.14	0.88	0.11	0.44	**1.21**
	6	0.81	0.30	0.87	0.13	0.45	**1.22**
	7	0.76	0.49	0.86	0.17	0.35	**1.14**
	8	0.63	0.54	0.85	0.02	0.38	**0.98**
MDD							
	4	0.20	0.25	0.19	0.22	0.16	**0.13**
	5	0.17	0.26	**0.14**	0.20	0.17	0.15
	6	0.17	0.24	0.13	0.18	0.19	**0.12**
	7	0.14	0.24	0.13	0.16	0.16	**0.11**
	8	0.17	0.25	0.13	0.23	0.18	**0.12**
CR							
	4	0.76	0.29	1.00	0.45	0.82	**1.66**
	5	0.85	0.24	1.20	0.28	0.61	**1.49**
	6	0.93	0.36	1.21	0.32	0.55	**1.85**
	7	1.06	0.50	1.18	0.40	0.57	**1.94**
	8	0.77	0.52	1.24	0.19	0.53	**1.61**

relatively good strategies except PCA&DWT RRL, revealing following the market is always a viable strategy, while OLMAR is less competitive on these data sets, which seems to verify that OLMAR is more adapted to large portfolios management.

Finally, we test the performance of all strategies w.r.t. transaction costs. Given many online brokers provide free stock trading, we set δ as 0 bps, 10 bps and 50 bps, respectively, per share for more general settings. Figure 3 presents the trend of gross cumulative profits W_T of different strategies w.r.t. various portfolios with $\delta = 0$, while Figure 4 and 5 reveal the performance with $\delta = 30$ bps and 50 bps, respectively. It is clear that the proposed system is negatively affected by transaction costs, since on each particular portfolio, the higher the transaction cost rate is, the lower the final cumulative profit obtained by PCA&DWT RRL strategy. On the contrary, passive investment strategies, UCRP and MV, are less affected by transaction costs, since they just need to rebalance the portfolio affected by the underlying stocks price movement which is normally tiny. However, on most portfolios, especially when the transaction cost rates are low, PCA&DWT RRL system significantly outperforms all other benchmark strategies in gross profits, which reveals that the proposed trading system is robust, consistent and competitive in making profits compared to other strategies. Nevertheless, the proposed strategy seems to be unsuitable for large portfolios trading since all figures show that the more stocks contained in a portfolio, the less the profit gained by our strategy with other hyperparameters fixed.

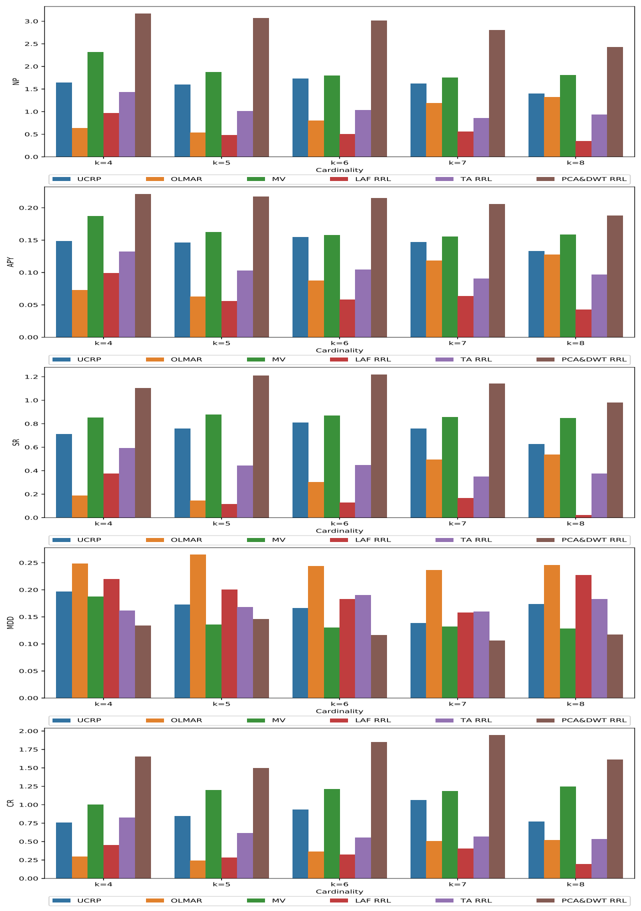

Figure 2: Boxplots of the performance of different strategies on various portfolios.

5 CONCLUSION

This paper proposes a novel portfolio trading system, PCA&DWT RRL, which not only embeds the portfolio rebalance function into the algorithm to trade portfolios at each period directly, but also combines PCA and DWT to preprocess the technical indicators extracted from the original stock price and volume data. The experimental results demonstrate that the proposed system consistently outperforms other portfolio selection strategies from previous literature. Moreover, we find that feature preprocessing is vital for the RRL trading algorithm in this setting. Future research could exploit more about the portfolio rebalance function to further improve the performance and make the system more adapted to large-scale portfolio selections for financial institutions.

REFERENCES

[1] Steven B Achelis. 2001. Technical Analysis from A to Z.
[2] Saud Almahdi and Steve Y Yang. 2017. An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown. *Expert Systems with Applications* 87 (2017), 267–279.
[3] W. Chen, H. Zhang, M. K. Mehlawat, and L. Jia. 2021. Mean–variance portfolio optimization using machine learning-based stock price prediction. *Applied Soft Computing* 100, 1 (2021), 106943.

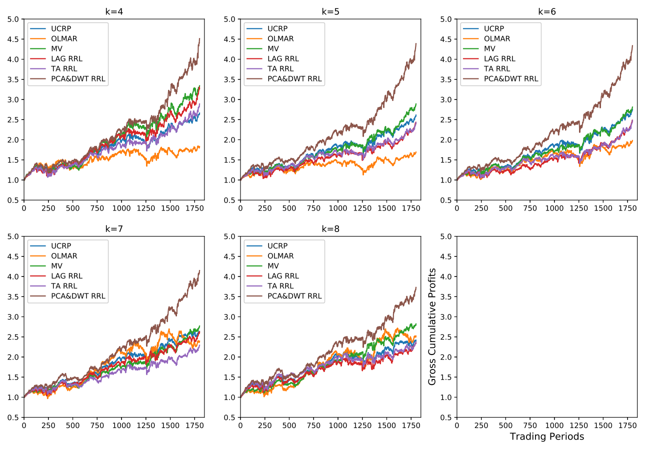

Figure 3: Gross cumulative profits of different strategies on various portfolios with $\delta = 0$. Repetitive labels are placed in the last empty panel.

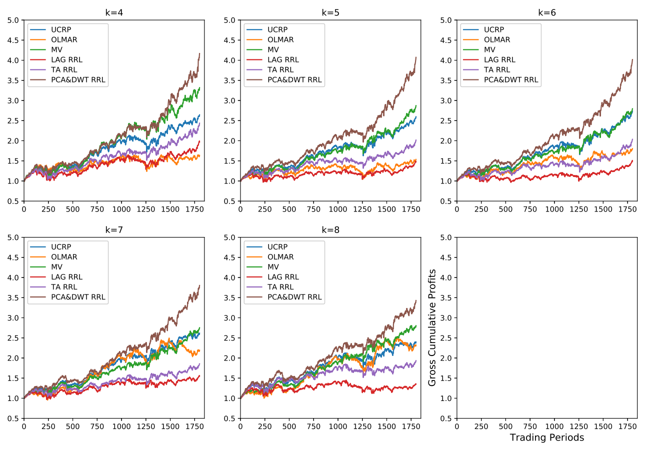

Figure 4: Gross cumulative profits of different strategies on various portfolios with $\delta = 10$ bps. Repetitive labels are placed in the last empty panel.

[4] Michael AH Dempster, Tom W Payne, Yazann Romahi, and Giles WP Thompson. 2001. Computational learning techniques for intraday FX trading using popular technical indicators. *IEEE Transactions on neural networks* 12, 4 (2001), 744–754.

[5] P. F. Dias, Crt Nogueira, H. G. Peixoto, and D. W. Moreira. 2018. Decision-Making for Financial Trading: A Fusion Approach of Machine Learning and Portfolio Selection. *Expert Systems with Applications* 115 (2018), S0957417418305037–.

[6] M. Fortier. 2007. TA-Lib: Technical Analysis Library. *http://www.ta-lib.org/* [Online; accessed March-2021] (2007).

[7] Aurélien Géron. 2019. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media.

[8] Thorsten Hens and Peter Wöhrmann. 2007. Strategic asset allocation and market timing: a reinforcement learning approach. *Computational Economics* 29, 3-4 (2007), 369–381.

[9] Lahmiri and Salim. 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, 2 (2014), 218–227.

[10] Gregory Lee, Ralf Gommers, Filip Waselewski, Kai Wohlfahrt, and Aaron O'Leary. 2019. PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software* 4, 36 (2019), 1237.

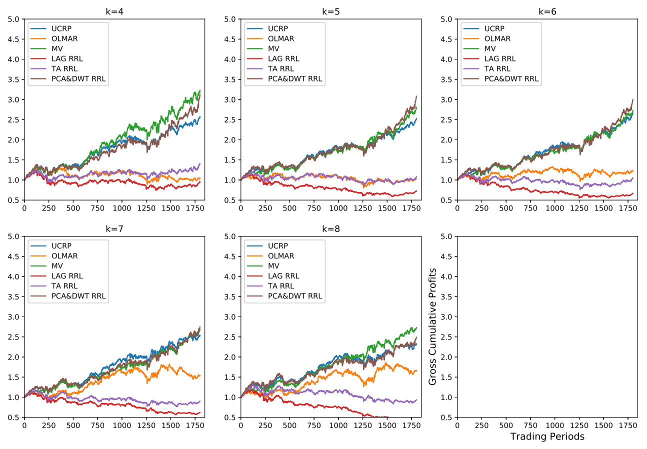

Figure 5: Gross cumulative profits of different strategies on various portfolios with $\delta = 50$ bps. Repetitive labels are placed in the last empty panel.

[11] Bin Li, Steven CH Hoi, Doyen Sahoo, and Zhi-Yong Liu. 2015. Moving average reversion strategy for on-line portfolio selection. *Artificial Intelligence* 222 (2015), 104–123.

[12] Bin Li, Steven CH Hoi, Peilin Zhao, and Vivekanand Gopalkrishnan. 2013. Confidence weighted mean reversion strategy for online portfolio selection. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 1 (2013), 4.

[13] Bin Li, Peilin Zhao, Steven CH Hoi, and Vivekanand Gopalkrishnan. 2012. PAMR: Passive aggressive mean reversion strategy for portfolio selection. *Machine learning* 87, 2 (2012), 221–258.

[14] Stephane G Mallat. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence* 11, 7 (1989), 674–693.

[15] Dietmar Maringer and Tikesh Ramtohul. 2012. Regime-switching recurrent reinforcement learning for investment decision making. *Computational Management Science* 9, 1 (2012), 89–107.

[16] Harry Markowitz. 1952. Portfolio selection. *The journal of finance* 7, 1 (1952), 77–91.

[17] John Moody and Matthew Saffell. 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks* 12, 4 (2001), 875–889.

[18] John Moody and Lizhong Wu. 1997. Optimization of trading systems and portfolios. In *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFEr)*. IEEE, 300–307.

[19] John Moody, Lizhong Wu, Yuansong Liao, and Matthew Saffell. 1998. Performance functions and reinforcement learning for trading systems and portfolios. *Journal of Forecasting* 17, 5-6 (1998), 441–470.

[20] João Nobre and Rui Ferreira Neves. 2019. Combining principal component analysis, discrete wavelet transform and XGBoost to trade in the financial markets. *Expert Systems with Applications* 125 (2019), 181–194.

[21] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[22] W. Wang, W. Li, N. Zhang, and K. Liu. 2020. Portfolio formation with preselection using deep learning from long-term financial data. *Expert Systems with Application* 143, Apr. (2020), 113042.1–113042.17.