Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)
Special Track on AI in FinTech

Multi-scale Two-way Deep Neural Network for Stock Trend Prediction

Guang Liu[1,2,†], **Yuzhao Mao**[1,2,†], **Qi Sun**[1],
Hailong Huang[1], **Weiguo Gao**[1,*], **Xuan Li**[1], **JianPing Shen**[1],
Ruifan Li[2] and **Xiaojie Wang**[2]
[1]PingAn Life Insurance Company of China, Ltd.
[2]School of Computer Science, Beijing University of Posts and Telecommunications
{liuguang230, maoyuzhao258,sunqi149}@pingan.com.cn
{huanghailong590,gaoweiguo801,lixuan208,shenjianping324}@pingan.com.cn
{rfli,xjwang}@bupt.edu.cn

Abstract

Stock Trend Prediction(STP) has drawn wide attention from various fields, especially Artificial Intelligence. Most previous studies are single-scale oriented which results in information loss from a multi-scale perspective. In fact, multi-scale behavior is vital for making intelligent investment decisions. A mature investor will thoroughly investigate the state of a stock market at various time scales. To automatically learn the multi-scale information in stock data, we propose a Multi-scale Two-way Deep Neural Network(MTDNN). It learns multi-scale patterns from two types of scale information, wavelet-based and downsampling-based, by eXtreme Gradient Boosting and Recurrent Convolutional Neural Network, respectively. After combining the learned patterns from the two-way, our model achieves state-of-the-art performance on FI-2010 and CSI-2016[1], where the latter is our published long-range stock dataset to help future studies for STP task. Extensive experimental results on the two datasets indicate that multi-scale information can significantly improve the STP performance and our model is superior in capturing such information.

1 Introduction

Stock Trend Prediction (STP), which automatically predicts future direction of the stock price movement, is of great importance for investors. It is challenging because stock data is non-stationary time series dominated by chaotic. It has attracted many researchers to explore such stochastic data [Tsai and Hsiao, 2010; Kara *et al.*, 2011; Li *et al.*, 2016].

To reduce the chaos in stock data, previous studies smooth the data with a single specific time scale to analyze the behavior of stock price movement (e.g. 5-minute moving average). However, the single-scale analysis ignores the multi-scale behavior within stock data. As depicted in Figure 1, stock trend moves toward different direction with multiple time scales, where s_1,s_2,s_3 indicate the wrong direction and s_4 conveys information towards the correct direction. Single-scale is insufficient to predict the moving trend.

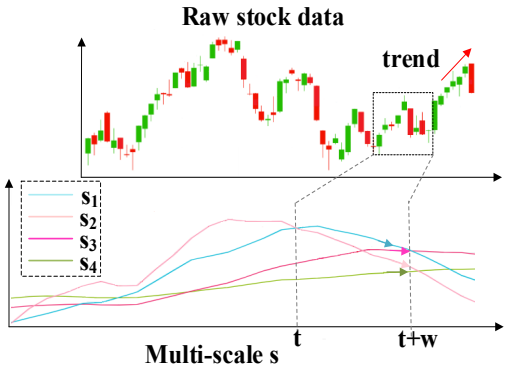

Figure 1: Intuitionistic view of multi-scale patterns within a stock data.

Notice that time scale is just one type of scale-information. [Hu and Qi, 2017] proposed a state-frequency memory network that uses Fourier transform to decompose memory state into multi-frequency components. [Lahmiri, 2014] used Discrete Wavelet Transform (DWT) to decompose a stock time series into multi-scale components of different resolutions. [Cui *et al.*, 2016] obtained multi-scale patterns directly by downsampling with different time scales. It is worth mentioning that all the above methods are not for STP task. In this paper, we insist that multi-scale information refers to stock price behavior not only at multiple scales but also in multiple types of scale-information. To explore the multi-scale patterns from two types of scale-information for the STP task, we propose a novel Multi-scale Two-way Deep Neural Network (MTDNN). One way is DWT-based. It uses eXtreme Gradient Boosting (XGBoost) to automatically ensemble the DWT-based multi-scale patterns. The other is downsampling-based. It uses Recurrent Convolutional Neural Network (RCNN) structure with a key operation to tempo-

[1]https://github.com/marscrazy/MTDNN
[2]†:equal comtribution
[3]∗:corresponding author

rally cascade the downsampling-based multi-scale patterns. Finally, we fuse the two types of multi-scale patterns by a fully connected layer to make predictions.

We evaluate our model on a benchmark dataset FI-2010. However, FI-2010 only has 10-day stock events which easily result in over-fitting. To address the above concerns, we collect and build a one-year range of one-minute stock dataset, China Stock Index 2016 (CSI-2016). Our model achieves state-of-the-art performance on both datasets.

The major contributions of this paper are summarized as follows:

(1) We propose a model that achieves state-of-the-art STP performance on a benchmark dataset and a long-range dataset, which strongly demonstrates the superiority of our model.

(2) We conduct a series of experiments to 1) compare different approaches of using multi-scale patterns and 2) analyze the scale characteristics of different types.

(3) We publish a new minute-level stock index dataset to help future studies on the task of STP.

The rest of this paper is organized as follows. Section 2 discusses the related works. Section 3 generally formalizes the STP task. Section 4 describes the architecture of MTDNN. Section 5 first introduces the dataset and experimental settings, then analyze the results. Section 6 is the conclusion and future works.

2 Related Works

2.1 Multi-scale for Time Series

Many studies concentrate on extracting the multi-scale pattern from time-series, to describe time-series more precisely. The multi-scale information of financial time-series has been extensively investigated [Dacorogna *et al.*, 1996]. By the similarity measured on multiple scales, the future price of given security can be estimated by finding a similar history price sequence across different financial markets [Papadimitriou and Yu, 2006]. In AI, some studies have explored the multi-scale information of time-series. The most prior work, ScaleNet [Geva, 1998], decomposes the time-series into different scales by Wavelet transform and extracts features from each scale by different Neural networks to obtain a prediction. More recently, Cui et al. [Cui *et al.*, 2016] use Convolutional Neural Network (CNN) to enhance the feature extraction ability, [Fernández *et al.*, 2019] apply extreme learning machine (ELM) and a Discrete Wavelet Transform (DWT) to capture the scaling-properties. The above methods with multi-scale information achieved remarkable improvement compared to the single-scale methods.

2.2 Stock Trend Prediction

Stock Trend Prediction (STP) is a typical classification task. Traditionally, Support Vector Machine (SVM) and Neural Network (NN) are thought to be very effective for STP [Kara *et al.*, 2011]. Due to the excessive parameter size, models are easily over-fitting to the training set. Ensemble-based methods, such as Random Forest (RF) [Patel *et al.*, 2015] which ensembles multiple trees to achieve better prediction and generalization performance, are introduced in STP. Recently, some pioneer researches have explored the effectiveness of deep learning models in STP [Deng *et al.*, 2017; Lin *et al.*, 2017]. The above researches indicate that STP task is lacking all kinds of publicly available benchmark dataset and only focusing on single-scale models.

3 Task Formulation

STP takes stock data as input to predict its moving trend. A stock data is a stock events time-series of T length, which we denote as $\mathbf{x} = \{x_t\}_T$ where $x_t \in \mathbb{R}^d$ is one stock event at t-th timestep with d dimensions (e.g. prices, volumes). The stock dataset is a collection of paired data $\mathbb{D} = \{(\mathbf{x}_n, y_n)\}_N$ where N is the number of samples in the dataset. y_n is the category given the n-th stock data $\mathbf{x}_n$, where

$$y_n = \begin{cases} -1 & \Delta p_T \leq -\alpha \\ 0 & -\alpha < \Delta p_T < \alpha \\ 1 & \Delta p_T \geq \alpha \end{cases} \tag{1}$$

represent the downward, stationary and upward stock price moving trend, respectively. The α is a threshold for trend direction judgement. Δp_T is the percentage change of the future mid-price compared with the current price, which is calculated as follows,

$$\Delta p_T = \frac{m_T(k) - p_T}{p_T}\,, \tag{2}$$

where $m_T(k) = \frac{1}{k}\sum_{i=1}^{k} p_{T+i}$, k is the prediction horizon.

STP is to construct a nonlinear function that can map an input stock data $\mathbf{x}_n$ to a category y_n as follows:

$$\hat{y_n} = f(\mathbf{x}_n; \theta), \tag{3}$$

where $f(\cdot)$ is the nonlinear mapping function, θ is the parameters and $\hat{y_n}$ is the predicted category. The objective is to learn a set of parameters θ that best fit $f(\cdot)$ to map an input $\mathbf{x}_n$ to the correct category y_n.

4 Model

4.1 Overview

The architecture of MTDNN is depicted in Figure 2. Our MTDNN model is a two-way end-to-end model. It comprises one wavelet-based way and one downsampling-based way. The two ways convey discriminative information, where the multi-scale information is the dominant force to help enhance the prediction of the stock trend. In the following of this section, we first define the wavelet-based way, then describe the downsampling-based way, at last, explain output and objective of the MTDNN.

4.2 Wavelet-based Way

In this way, we explore the multi-scale behavior of the stock data from a signal processing perspective. A set of stock data is regarded as a non-stationary and discrete signal. After a recursive decomposition of the signal by DWT, we can obtain a series of transformed multi-scale components. We first concatenate those components, then feed them to an XGBoost model to automatically ensemble the multi-scale information, finally output the category scores.

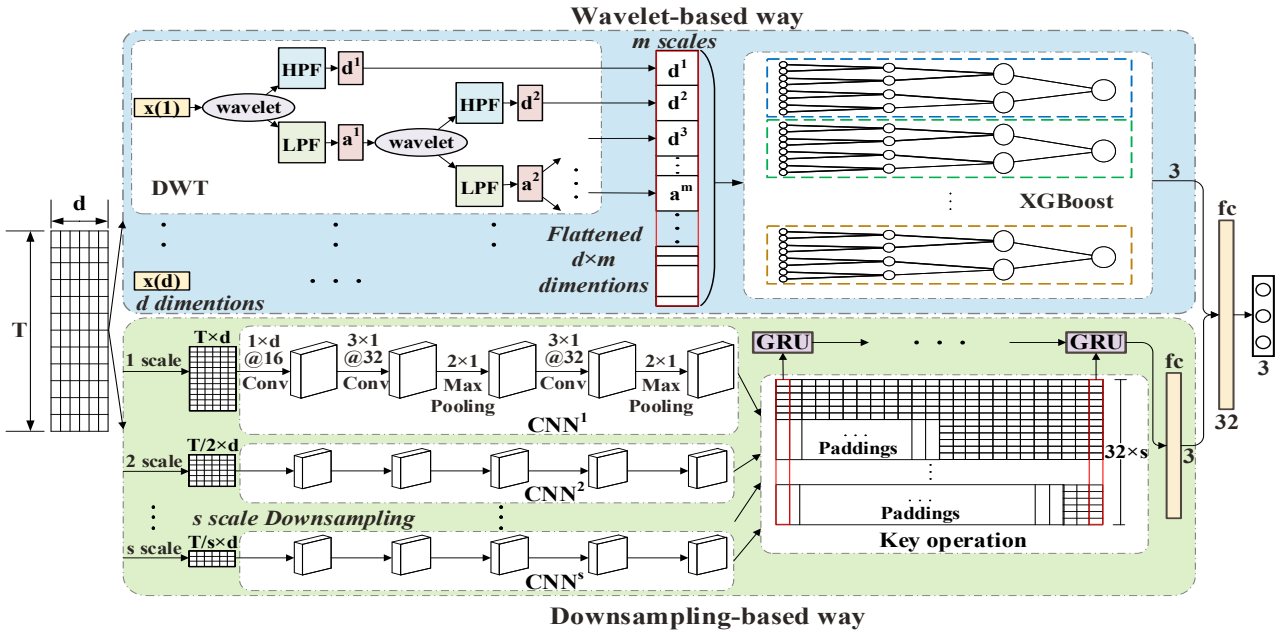

Figure 2: The architecture of MTDNN.

Discrete Wavelet Transform

DWT is the discrete version of the wavelet transform. It transfers the decomposition of a discrete signal into multi-scale components. Top-left of Figure 2 depicts the decomposition process of DWT. At the first level, given original signal $\mathbf{x}(i) = \{x_{t,i}\}_{t\in[1,T]}$ on the i-th dimensional, it is decomposed into approximation components $\mathbf{a}^1(i) = \{a_n^1(i)\}_{\frac{T}{2}}$ and detail components $\mathbf{d}^1(i) = \{d_n^1(i)\}_{\frac{T}{2}}$, by passing the signal through a Low-Pass Filter (LPF) and a High-Pass Filter (HPF), respectively. In this way, signals are downsampled by 2 so that frequency resolution is increased. For simplicity, we drop the index i whenever it is unambiguous from the context. The decomposition of the original signal can be formulated as,

$$a_n^1 = \sum_t h[2n-t]x_t\,, \tag{4}$$

$$d_n^1 = \sum_t g[2n-t]x_t\,, \tag{5}$$

where the superscript of a and d indicate the level of DWT. h and g are the LPF and HPF, respectively. n and t are the index of the corresponding components. The second level of DWT decompose the first level output a^1 into a^2 and d^2, then the third level till a specified level has been reached. The recursive iteration of wavelet decomposition can be illustrated as,

$$a_n^m = \sum_t h[2n-t]a_t^{m-1}\,, \tag{6}$$

$$d_n^m = \sum_t g[2n-t]a_t^{m-1}\,, \tag{7}$$

where m is the level index. $\mathbf{a}^{m-1} = \{a_t^{m-1}\}_{t\in[1,\frac{T}{2^{m-1}}]}$ is approximation components obtained from previous level. For a set of stock data, the approximation components $\mathbf{a}^m$ (low-frequency) maintain the information of the long-term moving trend within the historical data, and the detail component $\mathbf{d}^m$ (high-frequency) maintains its short-term moving trend information. Levels of DWT represent different resolutions of the original signal, which capture information about long-short term moving trends of different scales. We concatenate those components into a single vector. Thus given $\mathbf{x}$, the output is,

$$\mathbf{v}_i = [\mathbf{d}^m(i), \mathbf{d}^{m-1}(i), ..., \mathbf{d}^1(i), \mathbf{a}^1(i)]\,, \tag{8}$$

where $\mathbf{v}_i$ is the multi-scale feature vector for i-th feature dimension. We use $\mathbf{V}$ to represent output $[\mathbf{v}_0, \mathbf{v}_1, ..., \mathbf{v}_d]$ for simplify.

XGBoost

XGBoost is a scalable machine learning system for tree boosting [Chen and Guestrin, 2016]. Based on the gradient enhancement decision tree, it produces a prediction model with an ensemble of weak tree-based prediction models. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The tree-based nature is suitable for extracting features from mixed multi-scale information.

$$\hat{s}_{wavelet} = f_{xgb}(v) \tag{9}$$

where $\hat{s}_{wavelet} \in \mathbb{R}^3$ denotes the category score from wavelet-based way, f_{xgb} represents the XGBoost model.

4.3 Downsampling-based Way

In this way, we propose a novel strategy to temporally cascade a sequence of increasing multi-scale information by a RCNN structure. Firstly, we use a simple downsampling technique to transform stock data into multi-scale formations, then they

are fed to CNNs obtaining multi-scale spatial features. After a **key operation**, we obtain a sequence of increasing multi-scale information. Finally, we use GRU to temporally cascade those information and output categories.

Downsampling
Downsampling technique is a straightforward way to transform the original stock data $\mathbf{x}$ into multi-scale formations. Let s be the scale for downsampling. Then every s-th data point in $\mathbf{x}$ is kept to construct the new data $\mathbf{x}^s = \{x_{1+ls}\}_{l\in[0,L^s]}$, where $L^s = \lfloor T/s - 1 \rfloor$ is the length of $\mathbf{x}^s$. By setting different scales, we obtain a collection of down sampled multi-scale stock data $\mathbf{X} = \{\mathbf{x}^s\}_S$.

RCNN Structure
To extract multi-scale features from $\mathbf{X}$, we propose a RCNN structure to cascade outputs from a series of CNNs by RNN, where each independent CNN captures the spatial information from one scale formation of stock data and RNN temporally cascade such multi-scale spatial information. The key operation is how to construct and cascade the information.

We follow the structure of CNNpred [Ehsan and Saman, 2019] as our CNN method. CNNpred is a stock data-oriented CNN whose structure outperforms the other CNNs[Gunduz *et al.*, 2017; Di Persio and Honchar, 2016] for STP task. The configuration is depicted in the bottom of Figure 2. It is a 5-layer CNN. Given an input of stock data $\mathbf{x}^s \in \mathbb{R}^{L\times d}$, the first layer is a 1D convolution over features with 16 filters of $1 \times d$, after which is stacked with two convolutional layers with 32 filters of 1×3, each followed by a 2×1 max-pooling layer. The calculation of CNN can be simply represented as,

$$\mathbf{u}^s = f^s_{cnn}(\mathbf{x}^s) \tag{10}$$

where $\mathbf{u}^s = \{u^s_i\}_{i\in[1,\dot{L}^s]}$. Here, $u^s_i \in \mathbb{R}^{32}$ is the i-th of all $\dot{L}^s$ spatial feature vector obtained by CNN $f^s_{cnn}(\cdot)$ which is for stock data in scale s.

The **key operation** is to concatenate these multi-scale spatial features in such a way,

$$v_i = \begin{cases} [u^1_i, \mathbf{0}, ..., \mathbf{0}], & i \in [0, \dot{L}^1 - \dot{L}^2] \\ [u^1_i, u^2_{i-\dot{L}^1+\dot{L}^2}, ..., \mathbf{0}], & i \in (\dot{L}^1 + \dot{L}^2, \dot{L}^1 + \dot{L}^3] \\ \quad \vdots & \\ [u^1_i, u^2_{i-\dot{L}^1+\dot{L}^2}, ..., u^S_{i-\dot{L}^1+\dot{L}^S}], & i \in (\dot{L}^1 - \dot{L}^S, \dot{L}^1] \end{cases} \tag{11}$$

where $[, ...,]$ concatenates multiple vectors into single vector v_i. $\mathbf{0}$ is zero padding ensuring the same dimension as $\{v_i\}_{\dot{L}^1}$. Such operation can 1) make $\{v_i\}$ contains multi-scale information at each time-step; 2) let the multi-scale information increases over time.

We use Gated Recurrent Unit (GRU) to temporally cascade $\{v_i\}$, which can be represented as

$$h_i = f_{gru}(v_i, h_{i-1}) \,. \tag{12}$$

where h_i is the hidden state at the i-th time-step and f_{gru} is the GRU cell. The last hidden state $h_{\dot{L}^1}$ is passed to a fully connected neural network to make prediction,

$$\hat{s}_{sample} = f_{nn}(h_{\dot{L}^1}) \,. \tag{13}$$

where $\hat{s}_{sample} \in \mathbb{R}^3$ is the category score from downsampling-based way, f_{nn} denotes the fully connected neural network.

4.4 Output and Objective

We use a network with two fully connected layers to fuse category scores from both ways, and to output the category prediction results.

$$\hat{y} = f_{logit}(\hat{s}_{sample}, \hat{s}_{wavelet}) \,, \tag{14}$$

where $\hat{y}$ is the output score of our model, $f_{logit}(\cdot)$ denotes the output layer.

We use cross-entropy as our loss function to measure the difference between our predicted classification distribution $\hat{y}_n$ and real distribution y_n:

$$J = -\frac{1}{N}\sum_{n=1}^{N} y_n log(\hat{y}_n), \tag{15}$$

where θ represents all the parameter of the model, N is the total number of samples.

5 Experiments

5.1 Datasets and Settings

We test our model on two datasets: FI-2010 [Ntakaris *et al.*, 2018] and CSI-2016. The statistics of the two datasets are presented in Table 1.

Dataset		Train		Test	
		(%)	samples	(%)	samples
FI-2010	↓	32.03	352,300	31.18	31,837
	–	36.91		40.66	
	↑	31.06		28.16	
CSI-2016	↓	38.34	143,262	25.99	30,000
	–	25.21		48.99	
	↑	36.45		25.02	

Table 1: Dataset statistics.

FI-2010 is the first publicly available benchmark dataset of high-frequency Limit Order Book (LOB)[1] data. It comprises approximately 4.5 million events of 5 stocks from 10 consecutive days. Every 10 non-overlapping events are officially represented as a 144-D feature vector.

Experimental settings on this dataset are as follows. Setting the label threshold $\alpha = 0.002$, prediction horizon $k = 50$ and the input window size $T = 100$. The dataset provides 3 off-the-shelf normalised data: z-score, min-max and decimal precision normalisation. We follow the most previous work [Zhang *et al.*, 2019; Tran *et al.*, 2018] that use the first 40 z-score normalized dimensions as the feature vector $x_t = [p^i_a(t), v^i_a(t), p^i_b(t), v^i_b(t)]_{i\in[1,10]}$ which represent the top 10 prices and volumes of both ask and bid orders.

CSI-2016 is our collected dataset from three one-minute stock index data, including the Shanghai Stock Exchange

[1] A limit order book is a record of unexecuted limit orders maintained by the security specialist who works at the exchange. A limit order is a type of order to purchase or sell a security at a specified price or better, which is opposed to orders that match immediately.

(SSE) Composite Index SH000001, Shenzhen Stock Exchange Small & Medium Enterprises (SME Boards) Price Index SZ399005 and ChiNext Price Index SZ399006. It has over 170, 000 samples spanning a year from January 1st, 2016, to December 30th, 2016. Each sample $x_t = [p_h(t), p_l(t), p_o(t), p_c(t), v(t), a(t)]$ is a one minute data of 6 dimensions which are high, low, open, close, volume and amount, respectively.

Experimental settings on this dataset are as follows. The datasets are splited in strictly temporal order. Setting the label threshold $\alpha = 0.01$, prediction horizon $k = 5$, the input window size $T = 100$ and the feature dimension $d = 6$. All features are normalized by z-score. We firstly train the wavelet-based way and freeze its parameters, then train the rest of the model using the SGD algorithm with a learning rate of 0.0001 and weight decay 0.9.

5.2 Results and Analysis

We conduct a series of experiments to evaluate the performance of our model. We choose not only classical methods but also recently proposed an advanced model for comparison. In this section, we first analyze benchmark performance on FI-2010, then, analyze the results on our CSI-2016, finally give an ablation study to help understand the modules in our MTDNN. [Ntakaris *et al.*, 2018] suggests to use F1 as the major metrics, while we also present ACC performance for reference.

Model	ACC %	F1 %
SVM [Tsantekidis *et al.*, 2017b]	-	49.42
MLP [Tsantekidis *et al.*, 2017b]	-	55.95
CNN-I [Tsantekidis *et al.*, 2017a]	-	59.44
LSTM [Tsantekidis *et al.*, 2017b]	-	61.43
CNN-II [Tsantekidis *et al.*, 2018]	-	47.00
B (TABL) [Tran *et al.*, 2018]	75.58	73.64
C (TABL) [Tran *et al.*, 2018]	79.87	78.44
DeepLOB [Zhang *et al.*, 2019]	80.51	80.35
BL-GAM-RHN-7 [Luo and Yu, 2019]	82.00	80.88
Downsampling RCNN	80.79	80.72
DWT XGBoost	80.81	80.74
MTDNN	81.12	81.05

Table 2: Results of predicting the mid-price movements in the next 50 events on FI-2010 dataset.

Results on FI-2010

The model performance on FI-2010 is presented in Table 2, in which all the results are quoted from the original paper. All of the listed models for comparison are single-scale oriented methods.

Our two-way model achieved SOTA performance with 81.05% F1 score and 81.12% accuracy. In STP, a tiny improvement in classification would lead to a dramatic rise in profits. MTDNN achieves a higher F1 score than the previous SOTA model BL-GAM-RHN-7. We analyze the result from three aspects, 1) The two-way structure of MTDNN is more effective in extract multi-scale patterns than the one-way models. The one-way model can promote trend prediction performance to the same level (80+%). By combining the output score of two single-way models, our model achieves higher performance. 2) As we can see, DWT, Neural Tensor Network [Luo and Yu, 2019] and CNN are useful feature extractors in 80%-club models. Besides, most of the 80%-club models have the RNN structure, except DWT XGBoost. 3) Our key operation can effectively utilize the multi-scale patterns for STP. The Downsampling RCNN and DeepLOB has a similar structure, the major difference is the multi-scale transform and key-operation, which help Downsampling RCNN outperform the strong baseline DeepLOB.

Model	ACC %	F1 %
SVM [Kim, 2003]	51.50	51.81
RF [Kara *et al.*, 2011]	52.30	51.96
TreNet [Lin *et al.*, 2017]	52.38	52.50
FDNN [Deng *et al.*, 2017]	52.32	52.45
CNNPred [Ehsan and Saman, 2019]	56.63	52.93
SFM [Hu and Qi, 2017]	52.96	52.97
DWT MLP [Lahmiri, 2014]	57.29	54.19
Downsampling RCNN	62.74	61.35
DWT XGBoost	62.19	60.74
MTDNN	63.07	61.65

Table 3: Results on CSI-2016.

Results on CSI-2016

We choose seven models for comparison, where SFM is officially implemented and the others are implemented by ourselves. The middle two models are original multi-scale models for regression of stock prices, we modify them into STP models. The first five models are single-scale models originally for STP.

We present both ACC and F1 results in the table 3. Our MTDNN achieves the highest accuracy 63.07% and F1 score 61.65%. Compared with single-scale models. CNN was the strongest model for STP, however, it falls behind a simple MLP with just DWT multi-scale features. Both our single-way and two-way models outperform the other multi-scale models. It reveals that our models are superior existing multi-scale models in extracting and utilizing multi-scale features. Compared with the two adapted multi-scale models, our model obtains higher scores. It's noticeable that the DWT MLP uses the same input as our DWT XGBoost. The only difference between the aforementioned methods is the model used for extracting multi-scale features. The results suggest that the XGBoost is more effective than MLP in extracting multi-scale features from data after DWT.

Ablation Study

To further understand the multi-scale behavior in stock data, we make several variations of our model. The variations are tested under single- and multi-scale environment. The results are presented in Table 4. In single-scale environment, variations are fed with only raw data. The results are listed in

Model		CSI-2016		FI-2010	
		ACC	F1	ACC	F1
Single-scale	XGBoost	54.16	53.52	45.56	41.85
	CNN	56.63	52.93	56.73	58.22
	RNN	57.56	51.21	57.29	56.65
	RCNN	57.11	54.12	65.16	64.72
Multi-scale	DWT XGBoost	62.19	60.74	80.81	80.74
	DWT CNN	57.90	54.96	57.56	59.24
	DWT RNN	57.14	54.17	44.02	40.35
	DWT RCNN	57.58	50.21	44.58	37.71
	Downsampling XGBoost	54.18	53.51	45.60	41.82
	Downsampling RCNN*	57.21	51.95	59.45	58.03
	Downsampling RCNN	62.74	61.35	80.79	80.72
	Downsampling RCNN XGBoost	62.42	60.73	80.75	80.69
	Multi-kernel RCNN	58.14	54.40	67.16	66.85

Table 4: Ablation study

single-scale rows, which reveal the capability of each single model in STP task. In multi-scale environment, variations are fed with three types of scale-information. The results are listed in multi-scale rows, which demonstrate that models are sensitive to the type of scale-information. In this section, we give our analysis from two aspects as follows.

The single-scale rows presents the performance of four variations, where XGBoost directly regards the raw data as features to make predictions, CNN has a convolutional feature extraction before making predictions, RNN makes predictions by considering the temporal information over time, RCNN captures the temporal information from a series of CNN receptive fields (local spacial information). From the results, RCNN achieves the best performance on most of the indices, and XGBoost gets weak performance. It means 1) CNN features can significantly imporve the prediction of RNN, and raw data is hard to predict the moving trend even by a strong classifier (XGBoost achieves many SOTA performance in Kaggle tasks). Note that the structure of RCNN is very similar to [Zhang *et al.*, 2019], however we cannot reproduce their results in FI-2010.

In multi-scale rows, the first four rows are aforementioned variations fed with DWT-based multi-scale features. The following four variations are fed with downsampling-based multi-scale features. The last variation is fed with a new type of scale-information obtained by CNN with multiple kernel size.

Comparing the first four variations with their counterpart, XGBoost obtains significant increase, from the worst to the best, about 8 points in CSI-2016 and over 30 points in FI-2010. While the other variations get negative or slightly positive results. We analyze that DWT is a recursive convolution over low-frequency components with specified filters, which is similar to the operation in CNN. Thus CNN is hard to extract more information from DWT convolved features. The bad performance of RNN is due to the broken temporal structure in DWT features.

Comparing downsampling RCNN* and downsampling RCNN, where RCNN* is an alternative way of using output features from different CNNs and RCNN use our proposed key operation. In RCNN*, each CNN is followed with an independent RNN to temporally fuse the features of one scale. The output of each RNN is concatenated to make predictions. The results shows the superior of our key operation in using multi-scale information.

Comparing XGBoost, DWT XGBoost, downsampling XGBoost, downsampling RCNN and downsampling RCNN XGBoost, we can understand that 1) downsampling features valid on RCNN is useless on XGBoost, and 2) downsampling do not provide any new features which result in an equivalent performance between XGBoost and downsampling XGBoost, and 3) features from dowsampling RCNN cannot further boost the performance XGBoost.

Last, but not the least, Comparing DWT RCNN, downsampling RCNN and multi-kernel RCNN, we can understand that RCNN is better at capturing downsampling-based features. We think that downsampling can weaken the nonstationary in the raw data by explicitly droping out points in the raw data, and the denoised raw data can help CNN to extract more useful features.

6 Conclusion and Future Works

This paper proposes a multi-scale oriented model, MTDNN, for STP. Our motivation is to fully explore the potential of multi-scale information within the stock data. We explore two types of multi-scale information extracted from two modules, downsampling with RCNN and DWT with XGBoost, as the two way of our MTDNN. By combining the two modules, MTDNN achieves state-of-the-art performance on the benchmark FI-2010 dataset. We publish a one-minute dataset CSI-2016 and present the result for further study on the STP task. The results on both datasets reveal the effectiveness of multi-scale information and the superior of our model in using such information. In the future, we prepare to introduce an attention mechanism to dynamically choose the most relevant scale of information.

References

[Chen and Guestrin, 2016] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *SIGKDD 2016*, pages 785–794. ACM, 2016.

[Cui *et al.*, 2016] Zhicheng Cui, Wenlin Chen, and Yixin Chen. Multi-scale convolutional neural networks for time series classification. *arXiv preprint arXiv:1603.06995*, 2016.

[Dacorogna *et al.*, 1996] Michel M Dacorogna, Cindy L Gauvreau, Ulrich A Müller, Richard B Olsen, and Olivier V Pictet. Changing time scale for short-term forecasting in financial markets. *Journal of Forecasting*, 15(3):203–227, 1996.

[Deng *et al.*, 2017] Yue Deng, Zhiquan Ren, Youyong Kong, Feng Bao, and Qionghai Dai. A hierarchical fused fuzzy deep neural network for data classification. *IEEE Transactions on Fuzzy Systems*, 25(4):1006–1012, 2017.

[Di Persio and Honchar, 2016] Luca Di Persio and Oleksandr Honchar. Artificial neural networks architectures for stock price prediction: Comparisons and applications. *International journal of circuits, systems and signal processing*, 10:403–413, 2016.

[Ehsan and Saman, 2019] Ehsan and Saman. Cnnpred: Cnn-based stock market prediction using a diverse set of variables. *Expert Systems with Applications*, 129:273–285, 2019.

[Fernández *et al.*, 2019] César Fernández, Luis Salinas, and Claudio E Torres. A meta extreme learning machine method for forecasting financial time series. *Applied Intelligence*, 49(2):532–554, 2019.

[Geva, 1998] Amir B Geva. Scalenet-multiscale neural-network architecture for time series prediction. *IEEE Transactions on neural networks*, 9(6):1471–1482, 1998.

[Gunduz *et al.*, 2017] Hakan Gunduz, Yusuf Yaslan, and Zehra Cataltepe. Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations. *Knowledge-Based Systems*, 137:138–148, 2017.

[Hu and Qi, 2017] Hao Hu and Guo-Jun Qi. State-frequency memory recurrent neural networks. In *ICML*, pages 1568–1577, 2017.

[Kara *et al.*, 2011] Yakup Kara, Melek Acar Boyacioglu, and Ömer Kaan Baykan. Predicting direction of stock price index movement using artificial neural networks and support vector machines: The sample of the istanbul stock exchange. *Expert systems with Applications*, 38(5):5311–5319, 2011.

[Kim, 2003] Kyoung-jae Kim. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2):307–319, 2003.

[Lahmiri, 2014] Salim Lahmiri. 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):218–227, 2014.

[Li *et al.*, 2016] Xiaodong Li, Haoran Xie, Ran Wang, Yi Cai, Jingjing Cao, Feng Wang, Huaqing Min, and Xiaotie Deng. Empirical analysis: stock market prediction via extreme learning machine. *Neural Computing and Applications*, 27(1):67–78, 2016.

[Lin *et al.*, 2017] Tao Lin, Tian Guo, and Karl Aberer. Hybrid neural networks for learning the trend in time series. In *IJCAI-17*, pages 2273–2279, 2017.

[Luo and Yu, 2019] Wei Luo and Feng Yu. Recurrent highway networks with grouped auxiliary memory. *IEEE Access*, 7:182037–182049, 2019.

[Ntakaris *et al.*, 2018] Adamantios Ntakaris, Martin Magris, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods. *Journal of Forecasting*, 37(8):852–866, 2018.

[Papadimitriou and Yu, 2006] Spiros Papadimitriou and Philip Yu. Optimal multi-scale patterns in time series streams. In *SIGMOD 2006*, pages 647–658. ACM, 2006.

[Patel *et al.*, 2015] Jigar Patel, Sahil Shah, Priyank Thakkar, and K Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert Systems with Applications*, 42(1):259–268, 2015.

[Tran *et al.*, 2018] Dat Thanh Tran, Alexandros Iosifidis, Juho Kanniainen, and Moncef Gabbouj. Temporal attention-augmented bilinear network for financial time-series data analysis. *IEEE transactions on neural networks and learning systems*, 30(5):1407–1418, 2018.

[Tsai and Hsiao, 2010] Chih-Fong Tsai and Yu-Chieh Hsiao. Combining multiple feature selection methods for stock prediction: Union, intersection, and multi-intersection approaches. *Decision Support Systems*, 50(1):258–269, 2010.

[Tsantekidis *et al.*, 2017a] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Forecasting stock prices from the limit order book using convolutional neural networks. In *CBI 2017*, volume 1, pages 7–12. IEEE, 2017.

[Tsantekidis *et al.*, 2017b] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning to detect price change indications in financial markets. In *EUSIPCO 2017*, pages 2511–2515. IEEE, 2017.

[Tsantekidis *et al.*, 2018] Avraam Tsantekidis, Nikolaos Passalis, Anastasios Tefas, Juho Kanniainen, Moncef Gabbouj, and Alexandros Iosifidis. Using deep learning for price prediction by exploiting stationary limit order book features. *arXiv preprint arXiv:1810.09965*, 2018.

[Zhang *et al.*, 2019] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, 2019.