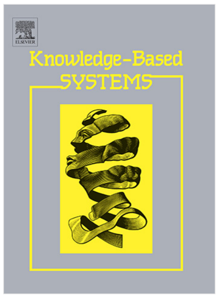
Accepted Manuscript



Deep learning-based feature engineering for stock price movement prediction

Wen Long, Zhichen Lu, Lingxiao Cui

PII: S0950-7051(18)30526-4
DOI: https://doi.org/10.1016/j.knosys.2018.10.034
Reference: KNOSYS 4553

To appear in: *Knowledge-Based Systems*

Received date : 9 April 2018
Revised date : 21 October 2018
Accepted date : 24 October 2018

Please cite this article as: W. Long, Z. Lu and L. Cui, Deep learning-based feature engineering for stock price movement prediction, *Knowledge-Based Systems* (2018), https://doi.org/10.1016/j.knosys.2018.10.034

Deep Learning-Based Feature Engineering for Stock Price Movement Prediction

Wen Long[a,b,c], Zhichen Lu[a,b,c,*], Lingxiao Cui[a,b,c]

[a]*School of Economics & Management, University of Chinese Academy of Sciences, Beijing 100190 P.R.China*
[b]*Research Center on Fictitious Economy & Data Science, Chinese Academy of Sciences, Beijing, 100190 P.R.China*
[c]*Key Laboratory of Big Data Mining & Knowledge Management, Chinese Academy of Sciences, Beijing, 100190 P.R.China*

Abstract

Stock price modeling and prediction have been challenging objectives for researchers and speculators because of noisy and non-stationary characteristics of samples. With the growth in deep learning, the task of feature learning can be performed more effectively by purposely designed network. In this paper, we propose a novel end-to-end model named multi-filters neural network (MFNN) specifically for feature extraction on financial time series samples and price movement prediction task. Both convolutional and recurrent neurons are integrated to build the multi-filters structure, so that the information from different feature spaces and market views can be obtained. We apply our MFNN for extreme market prediction and signal-based trading simulation tasks on Chinese stock market index CSI 300. Experimental results show that our network outperforms traditional machine learning models, statistical models, and single-structure(convolutional, recurrent, and LSTM) networks in terms of the accuracy, profitability, and stability.

Keywords: Stock Price Prediction, Feature Engineering, Deep Learning
2010 MSC: 00-01, 99-00

*Corresponding author.
Email addresses: `longwen@ucas.ac.cn` (Wen Long), `luzhcihen16@mails.ucas.ac.cn` (Zhichen Lu), `clxdsjyx@163.com` (Lingxiao Cui)

1. Introduction

Financial time series forecasting, particularly stock price forecasting, has been one of the most difficult problems for researchers and speculators. It plays a key role in trading strategies to identify opportunities to buy and sell a stock. Difficulties are mainly caused by the uncertainty and noises of samples. The generation of samples is not only a consequence of historical market behaviors, but also affected by information such as macroeconomy and investor sentiments. Several categories of methods and data sources were used in stock market prediction[1, 2, 3, 4]; commonly used methods were modeling the relationship between the historical behavior and future movement of the price, and using historical market samples to predict the future trend or value of the price[1]. Traditional statistical methods such as linear regression, auto-regression and moving average (ARMA), and GARCH were much favorable for financial time series forecasting, because of their interpretability. The key of the prediction process was the feature engineering part; technical analysis[5] was mostly performed to extract features from the original market data. Features were subjectively designed assuming that future movements of the price were results of historical behaviors. Models built on features of technical analysis were based on some assumptions on patterns of the market, and the success of models mostly depended on the correctness of these assumptions. Apart from the traditional statistical models, machine learning algorithms such as naive Bayes(NB), logistic regression(LR), random forest(RF), and k-nearest neighbors(KNN) were also used to learn the relationship between the features from technical analysis and future price[6, 7], because of their stronger capability of learning, ease of interpretation, and absence of presumptions. Support vector machine(SVM) and artificial neural networks (ANNs)[8, 9, 10] were leading algorithms in the application of traditional machine learning methods on financial studies owing to their remarkable capability of nonlinear mapping and fitting. However, because of the "black-box" property, mapping relationship learned by models are lacking of interpretation and their performance is directly related to quality of the features. Moreover, owing to their capability of nonlinear mapping and fitting, over-fitting presents one of the biggest obstacles in practical applications.

Among deep learning methodologies, neural networks play key roles in the feature extraction process, and feature engineering can be performed by purposely designing an integrated network structure using a series of available neural network feature extractors for specific samples. In each layer of a network, nonlinear mapping is implemented, and with the depth of the network growing, features are transformed by a high-layer nonlinear mapping, so that deeper feature maps can be more suitable for the final task. Existing works have exploited most types of neural networks for financial time series modeling[11, 12, 13]. Most of these work performed two-stage predictions, which means defining and extracting features first and inputing them into models for prediction. However, the highlighted difference between deep learning methodologies and traditional neural networks is that deep learning provides a group of units(convolutional, recurrent, LSTM, etc.) for designing network structures according to specific data formation and objective tasks. And the feature engineering part can be integrated into classification model to build an end-to-end model. Although some networks have been designed to learn the trading rule[14] and fit the distribution of the limit order book[15], in the task of feature extraction for price trend prediction, deep learning methodologies are barely utilized to design a specialized network that adequately considers characteristics of samples and objective tasks.

In this study, we propose a novel end-to-end model named multi-filters neural network (MFNN) specifically for feature extraction on financial time series samples and price movement prediction task. By performing feature transformation on sequential market data with MFNN, samples are mapped into feature space where different patterns are more discriminable for eventually classification-based prediction. In our proposed network structure, the convolutional and recurrent units are combined as an integral network, by which features transformed from different filters are merged and information from different views can be obtained. We use our MFNN to extract features from high-frequency market samples of the Chinese market index(CSI 300) and then recognize extreme markets by extracted features to exploit the suitable trading signal. Six types of extreme markets in five different prediction windows are defined in our experiments to obtain a suitable definition of trading signals. A simple prediction-based trading strategy is tested based on signals from our prediction model to evaluate profitability and stability of our network.

Contributions of our work can be summarized as follow: First, a novel end-to-end network designed with deep learning methodologies is proposed specifically for feature extraction from multi-variate financial time series data and classification-based prediction. Features containing different types of information are extracted by recurrent and convolutional units, and are integrated for further task of classification. Unlike most two-stage feature extraction and prediction works, the whole feature extraction and prediction processes are integrated in our end-to-end model. The second contribution of our work is that we apply our network to an extreme market prediction-based trading strategy, and tune the prediction windows and definition of the extreme market to find the most suitable trading signals in the high-frequency market data of CSI 300 index. To present the superiority of our proposed model in terms of the prediction, profitability, and stability, traditional machine learning, statistical models, and neural networks are also involved in our experiments as baseline, and our model shows better performance for both market prediction and simulation.

The remainder of this paper are organized as follows. Section 2 summarizes related works of methodologies and application of machine learning and deep learning. It also presents existing works on stock market prediction. Section 3 specifies details of our proposed MFNN and data preprocessing methodology. Section 4 describes the training methodology, tricks of prediction, and experimental results of the classification and market simulation. Finally, Section 5 concludes this work and presents some brief comments.

2. Related Work

2.1. Feature Engineering of Financial Time Series

Unlike picture, text and speech samples, whose raw inputs already contain most information needed for final objectives, stock price movements are results of multiple factors such as macroeconomy, financial situation of a company, investors' sentiments, etc. And financial time series contain high noise. To predict stock price movement, features containing useful information are needed, so feature extraction and selection play significant roles in stock price movement prediction. Existing studies have applied traditional statistical and machine learning methods for financial time series modeling and prediction. These models are built on features extracted by some specific methodologies.

Commonly used feature engineering methodologies contain technical analysis and statistical methodologies. Technical analysis presumes that future behaviors are correlated to some historical patterns[16], and a few technical indicators are defined to describe these patterns such as the moving average(MA)[17] , momentum[18] , Bollinger band[19], etc. Most of these indicators are mathematical expressions of historical price series defined to describe specific characteristics of presumed patterns. Nevertheless, because features extracted by designed indicators are based on presumed patterns, some information may be lost in this approach. Statistical methodologies focus on information compression and dimension reduction, and features extracted by them are mostly used in the machine learning methods. The universally used methods include the principal component analysis(PCA)[12], independent component analysis(ICA)[20], and locally linear embedding(LLE)[21], etc. They focus on the distribution of the samples, and the extraction operation is performed on data sets. Some transformation or kernel-based operation is needed to ensure an identically distributed condition of samples, and the whole process may be sensitive to hyper parameters. In recent years, a tensor-based method called the signature of path[22, 23] was proposed to calculate a unique vectorized representation of a multi-variate stream, and its attempt at depicting a financial time series[24] yielded considerable results.

2.2. Models for Financial Time Series Modeling

Both statistical and machine learning methods are universally used in financial time series modeling and prediction, most of which are implemented on features extracted by technical analysis and statistical methodologies. Because of their good capability of interpretation, statistical models are used to verify assumptions of market behind features, and predictions are based on verified assumptions. Commonly used statistical models include ARMA[25], ARCH[26], GARCH[27] and VAR[28],etc. Machine learning methods provide considerable capability of learning potential relationships between patterns in features and labels. They learn parameters of models by fitting training samples and presume that the distribution of the training set and test set in the feature space are identical[29, 30, 31, 10].

However, performance of either statistical models or machine learning models mostly depends on quality of features, which makes inappropriate features possibly lead to underperformance of models. Technical analysis based feature extraction are based on assumptions and subjective insights on market, but some of assumptions may not be rigorously proved, and some may be efficient only in specific market situation and lack of generalization. Statistical methodologies based feature engineering, on the other hand, simply perform feature transformation based on the distribution of original features, but does not consider the relationship between features and objective. Also, both of these two kinds of feature engineering methods may be sensitive to hyper parameters.

Recently, decision tree C4.5 and C5.0 combined with an improved filter feature selection method were proposed to predict the listing statuses of Chinese-listed companies[32]. The experiment on 23,497 company-year observations demonstrated that the proposed method showed better performance than genetic algorithm based wrapper method. Inspired by their works, we got an idea of further integrating the feature engineering process into models, so that features can be learnt with consideration of objectives. That leads us to deep learning methodologies, by which network structure can be tailored for feature engineering on specific data formation with consideration of objective task.

2.3. Deep Learning for feature learning

In recent years, deep learning methodologies have achieved remarkable results in computer vision[33], speech recognition[34], and some path recognition such as handwritten character recognition[35]. The main difference between traditional methods and deep learning is that the network structure can be tailored for specific data formation and objective task. Deep learning provides a series of units such as convolutional unit[36], recurrent unit[37], gate recurrent unit[38], and long-short memory term unit[39] for feature extraction on samples with different characteristics. The feature extraction process can be integrated into networks, and structures of networks can be purposely designed according to characteristics of samples. GoogLeNet[40] and AlexNet[33] were designed to enhance abilities of learning information from the pixel data. Attention-over-attention neural network[41] was designed for extracting features from text data for a reading comprehension task, CNN-bLSTM[42] was designed for speech recognition, and R-CNN[43] was proposed for scalable objection detection.

Learning ability of networks can be enhanced with the depth of the networks growing. However, problems such as exploding and vanishing gradient[44, 45, 46] may be incurred. Tricks such as dropout[47], batch normalization[48], and residual[49] were proposed to deal with these problems and enhance the efficiency of training.

Some of deep learning methodologies have been used in stock price prediction. The classical deep brief network was updated to model continuous data by using the restricted Boltzmann machine and its good performance was verified by application on exchange-rate forecasting[11]. DNN with the $(2D)^2$PCA feature extractor[12] was proposed to predict stock price and yield a better profitability and accuracy of the hit rate than radial basis function neural networks and recurrent neural networks. Recurrent neural network was applied to the classification task on limit order book samples[50] for a trading signal, and it exhibited its ability to capture the nonlinear relationship between the near-term price-flips and spatio–temporal representation of the limit order book.

However, most of these works are concentrated on using single type networks to make two-stages predictions of stock price. Although some of them performed feature engineering before feeding samples to networks, the feature learning ability of tailored and integrated network structure has been barely explored on financial time series. The motivation of our work is to bridge this gap and explore the performance of deep learning based feature engineering on financial time series.

3. Deep Feature Engineering on Financial Time Series

3.1. Data sets

In the task of stock price prediction, we define $x_1, x_2, x_3...x_t, ...$ as indicators sequences, in our works, six indicators $x_t = (o_t, c_t, h_t, l_t, v_t, a_t)$ are obtained at each time step in 1-minute frequency, including open price(stock price at the start of each time step), close price(stock price at the end of each time step), highest price(the highest price among each time step), lowest price(the lowest price among each time step), volume(the number of shares traded in a security during each time step), amount(the amount of money traded in a security during each time step). At each time step, features of each sample is composed of the six indicator sequences over past 120 minutes, which can be denoted as $X_t = (x_{t-119}^T, x_{t-118}^T, ..., x_t^T)^T = (O_t, C_t, H_t, L_t, V_t, A_t)$. Features of each sample are then scaled as follow:

$$\widetilde{Z_t} = \frac{Z_t - mean(Z_t)}{std(Z_t)} \tag{1}$$

where $Z_t \in \{O_t, C_t, H_t, L_t, V_t, A_t\}$ denotes each univariate time series of each segmented sequence. Each univariate series of each sample is composed of 120 historical indicators,

$mean(Z_t)$ and $std(Z_t)$ denote the mean value and standard deviation of the 120 historical indicators, respectively.

In our works, market data of CSI 300 in 1-min frequency from December 9th, 2013 to December 7th, 2016 are used for training and testing. Sequences X_t are sampled from the raw data at each minute and scaled, so the t of X_t ranges from 11:00 am(120 minutes after the market opening) December 9th, 2013 to 15:00 pm(time when the market closes), December 7th 2016. Samples before and after December 31th, 2015 are used for training set and test set, which in proportion of 7:3. After sampling, the labeling methodology described in Eq 2 is performed on the data set.

$$L_t = \begin{cases} +1 & r_t > r_\theta \\ 0 & \text{Others} \\ \text{-1} & r_t < r_{1-\theta} \end{cases} \tag{2}$$

L_t denotes the label of sample X_t, $r_t = ln\frac{close_{t+t_{forward}}}{close_t}$ denotes the logarithm return of the stock index in $t_{forward}$ minutes after t, r_θ and $r_{1-\theta}$ denotes thresholds by which samples are labeled different categories($\theta \leq 0.3$). Samples in training set are first ranked in descending order of their future return r_t, r_θ and $r_{1-\theta}$ denote future returns of samples at the $(100*\theta)$th percentile and the $100*(1-\theta)$th percentile, respectively. Samples above the $(100*\theta)$th percentile are labeled +1, samples below the $100*(1-\theta)$th percentile are labeled -1, and samples between the $(100*\theta)$th percentile and the $100*(1-\theta)$th percentile are labeled 0. Samples in test set are labeled according to the thresholds r_θ and $r_{1-\theta}$ calculated among training set. For example, when $\theta = 0.1$ and $t_{forward} = 10$, samples of training set are ranked in descending order of their future-10-minutes-return, and samples above the 10th percentile are labeled +1, samples below the 90th percentile are labeled -1, others are labeled 0; test set are labeled +1, 0, -1 according to the thresholds $r_{0.1}$ and $r_{0.9}$ of training set. Training set and test set are then rebalanced by randomly drop samples with label 0 to insure that percentages of each categories are about to equal.

The smaller θ is, the higher r_θ is(also the lower $r_{1-\theta}$ is), and samples labeled +1 will have higher future returns(lower returns for samples labeled -1). With five kinds of $\theta = 0.1, 0.15, 0.2, 0.25, 0.3$ and six kinds of $t_{forward} = 5, 10, 15, 20, 25, 30$, we generate 30 data sets, aiming to explore the relationship between predictability of samples and θ, as well as

Table 1: Statistic of data sets

(a) Numbers of samples in each category with different θ and $t_{forward} = 5$

θ	Training sets			Testing sets		
	+1	0	-1	+1	0	-1
0.1	12239	12277	12194	2454	2412	2370
0.15	18355	18397	18315	4511	4386	4261
0.2	24470	24504	24433	6880	6761	[illegible]
0.25	30588	30622	30551	9667	9521	9375
0.3	36699	36738	36665	12982	12652	12322

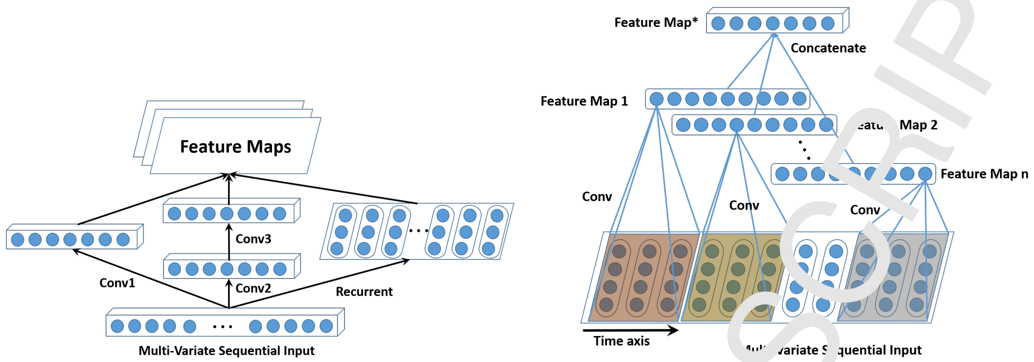
(b) tuples $(r_\theta, r_{1-\theta})$ in different θ and $t_{forward}$

θ	$t_{forward} = 5$	$t_{forward} = 10$	$t_{forward} = 15$	$t_{forward} = 20$	$t_{forward} = 25$	$t_{forward} = 30$
0.1	(0.0026, -0.0025)	(0.0036, -0.0035)	(0.0044, -0.0042)	([illegible], -0.0049)	(0.0057, -0.0054)	(0.0063, -0.0059)
0.15	(0.0019, -0.0018)	(0.0027, -0.0026)	(0.0033, -0.0031)	(0.0039, -0.0036)	(0.0044, -0.0039)	(0.0048, -0.0043)
0.2	(0.0014, -0.0013)	(0.0022, -0.002)	(0.0026, -0.00[illegible])	(0.003, -0.0027)	(0.0034, -0.003)	(0.0038, -0.0033)
0.25	(0.0011, -0.001)	(0.0017, -0.0015)	(0.0021, -0.0019)	(0.0024, -0.0021)	(0.0027, -0.0023)	(0.003, -0.0025)
0.3	(0.0008, -0.0007)	(0.0013, -0.0011)	(0.0016, -0.0014)	(0.0019, -0.0016)	(0.0021, -0.0017)	(0.0023, -0.0019)

$t_{forward}$. Table 1 presents statistics of data sets. Table 1(a) presents numbers of samples labeled +1, 0, -1 in each of the five data sets, when $t_{forward} = 5$ and $\theta = 0.1, 0.15, 0.2, 0.25, 0.3$. Table 1(b) presents values of r_θ and $r_{1-\theta}$ of 30 data sets. For example, when $t_{forward} = 5$ and $\theta = 0.1$, samples whose prices rise for more than 0.26% in future 5 minutes are labeled +1, and whose prices fall for more than -0.25% in future 5 minutes are labeled -1, and the other samples are labeled 0. Samples labeled 0 are randomly dropped to insure numbers of samples labeled +1, 0, -1 are about to equal, which are 12239, 12277, 12194 in training set, and 2454, 2412, 2370 in test set.

3.2. Multi-Filters for Feature Engineering on Financial Time Series

To extract features from samples, a network module is designed by integrating both convolutional neurons and recurrent neurons. Fig 1(a) demonstrates how these two kinds of neurons are integrated in our module. The left part and middle part of the multi-filters module perform convolution operation on raw inputs. The width of convolution filters is set to the number of indicators at each time step, which is 6 for our samples. By performing



(a) Multi-Filters module (b) Convolutional operation on sequential input

Figure 1: Multi-Filters module for feature engineering on sequential samples

convolution operation through time axis, discrete information are involved into feature maps. The way that convolutional filters extract features from sequential samples is shown in Fig 1(b), which can be formulated as

$$H_t = \sigma(\sum W * \widetilde{X_t} + b) \tag{3}$$

where $\widetilde{X_t} = (\widetilde{O_t}, \widetilde{C_t}, \widetilde{H_t}, \widetilde{L_t}, \widetilde{V_t}, \widetilde{A_t})$ denotes the input of the convolutional filters, which is scaled sequential sample in our work, H_t denotes the feature maps after the convolution operation, $*$ denotes the convolution operation, W denotes weights of the convolutional filter, b is a slack term, and a sigmoid $\sigma(\cdot)$ function is used for non-linear activation. By stacking convolutional filters, dimensions of features can be reduced and key information can be filtered and condensed into a lower-dimensional feature space. The multi-filters module integrates both single layer and multi-layers convolutional filters to obtain discrete information extracted through time axis.

Market data at each time step within a sample partly result from historical behaviors and contain different amounts of information in different sub-periods. So to capture these information contained in different sub-periods, the right part of multi-filters module extracts features by recurrent neurons on raw inputs, which can be formulated as:

$$h_t = F_h(\widetilde{x}_t, h_{t-1}) = \sigma(W_h \times \widetilde{x}_t + U_h \times h_{t-1} + b_h) \tag{4}$$

where $\widetilde{x}_t$ denotes the 6-D indicators vector at time step t. W_h denotes weights of connections between input and hidden states, U_h denotes weights of connections between hidden states h_t and h_{t-1}, and b_h denotes bias. Eq 4 is performed on scaled samples $\widetilde{X_t} = \{\widetilde{x}_{t-119}, \widetilde{x}_{t-118}, ..., \widetilde{x}_t\}$ so that feature maps $H_t = \{h_{t-119}, h_{t-18}, ..., h_t\}$ can be obtained from recurrent filters. By recurrently feeding hidden states from each time step to neurons of their next time step, feature maps composed of outputs from hidden states will contain information with memory of previous steps.

Features extracted by convolutional filters and recurrent filters are sub-sampled and padded to insure the length of feature maps are equal. Numbers of hidden nodes in recurrent layers are equal to numbers of convolutional filers of the left part, as well as that of the second layer of the middle part. Then feature maps with same size from multi-filers are concatenated as multi-channels feature maps and fed to the next layer.

3.3. Architecture of Multi-Filters Neural Networks

We further extend the multi-filters module to a deeper architecture, and name this network Multi-Filters Neural Networks(MFNN) since it integrates multiple kinds of filters(convolutional and recurrent filters) for feature learning. Fig 2 demonstrates details of MFNN. Two Multi-Filters modules are stacked to a deeper architecture for better feature learning ability. At the tail of the last stacked module, feature maps are flattened to a single vector, and concatenated with a fully connected layer. And a softmax output layer[51] is used for eventually classification based prediction. At the output layer we choose cross entropy as loss function, which is calculated by:

$$Loss = -\frac{1}{n}\sum_i [y_i ln z_i + (1-y_i) ln(1-z_i)] \tag{5}$$

where z_i denotes the output and y_i denotes the actual label.

When depth of a network increases, gradient issues of vanishing and explosion[44, 45, 46] may be incurred, which may interrupt the training process. To prevent our network from these issues, we introduce batch normalization after each filter, which can be performed as:

$$\hat{H}_t = \frac{H_t - E[H_t]}{\sqrt{Var[H_t]}} \tag{6}$$

$$y = \gamma \hat{H}_t + \beta \tag{7}$$

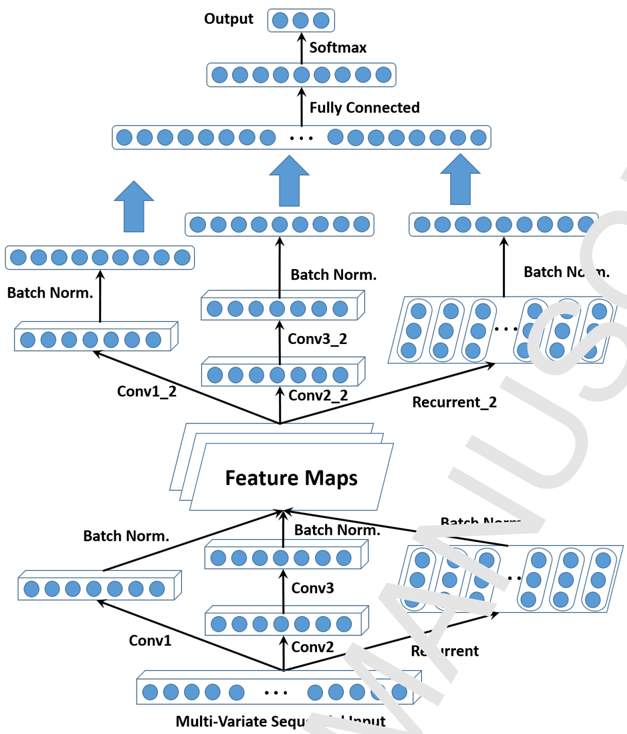


Figure 2: Structure of the Multi-Filters Neural Network

where H_t denotes activation of filters (convolutional and recurrent), y denotes features after batch normalization, γ and β are parameters of batch normalization to be learnt. Mean values $E[\cdot]$ and standard deviation $\sqrt{Var[\cdot]}$ are calculated among each batch. Also, the powerful nonlinear mapping ability of deep structures may lead to over-fitting on the training samples. To alleviate this issue, we implement dropout[47] on the final fully-connected layer and recurrent filters, whose weights tend to be dense. It is performed by randomly omitting a certain percentage of neurons while updating weights by back propagating[52] gradient.

4. Experiment

To evaluate effectiveness of the proposed MFNN, we design experiment process shown in Fig 3. Models are trained and evaluated by accuracy on test sets among 30 data sets, and the best performing one is used for market simulation by using prediction results as trading

Table 2: Training times(minutes) for MFNN on 30 data sets

θ	$t_{forward} = 5$	$t_{forward} = 10$	$t_{forward} = 15$	$t_{forward} = 20$	$t_{forward} = 25$	$t_{forward} = 30$
0.1	72.40	64.20	120.00	84.20	106.40	11[illegible].40
0.15	149.45	135.80	28.00	210.00	158.20	[illegible]10.00
0.2	182.92	165.42	250.00	233.33	220.83	170.00
0.25	184.97	268.67	310.00	276.93	310.[illegible]	310.00
0.3	247.00	390.00	85.80	390.00	33[illegible].75	390.00

signals.

4.1. Experimental Setting & Training Methodology

We evaluate performance of our MFNN on 30 data sets, traditional machine learning models and statistical models are used as baselines. Specifically, we build single-type-filter networks using recurrent unit, convolutional unit and Long Short Term Memory(LSTM) unit as comparisons to test the effectiveness of MFNN.

Back propagation with the stochastic gradient descent (SGD) optimizer is used to learn parameters of the network. Details of training methodology are presented in Algorithm 1. The learning rate ρ of SGD is initialized to 0.5 because we make a trade-off between training time and model performance. As shown in Fig 4, higher learning rate results in less epochs for convergence, but performance of model is restricted. Lower learning rate on the opposite, leads to more epochs for training but a better optimum. Batch size(N_{batch}) is set to 400 to correspond to the learning rate. We use learning rate decay and early stopping to adapt the training process, because it is proved that learning rate decay can prevent catastrophic events(sudden rocketing of training loss and gradient norm)[53]. The learning rate of SGD will be halved if the accuracy on the validation set hasn't been improved for 20 epochs. Training will stop if the accuracy on validation set hasn't been improved for 150 epochs(after 7 times learning rate decay) to prevent network from over-fitting. Other initialized parameters $T_{LastPeak} = 0$, $Acc_{val} = 0$, $Epoch = 0$ are temporary variable to record information such as for how many epochs accuracy on the validation set hasn't been improved. Table 2 gives training times(minutes) for MFNN on 30 data sets.

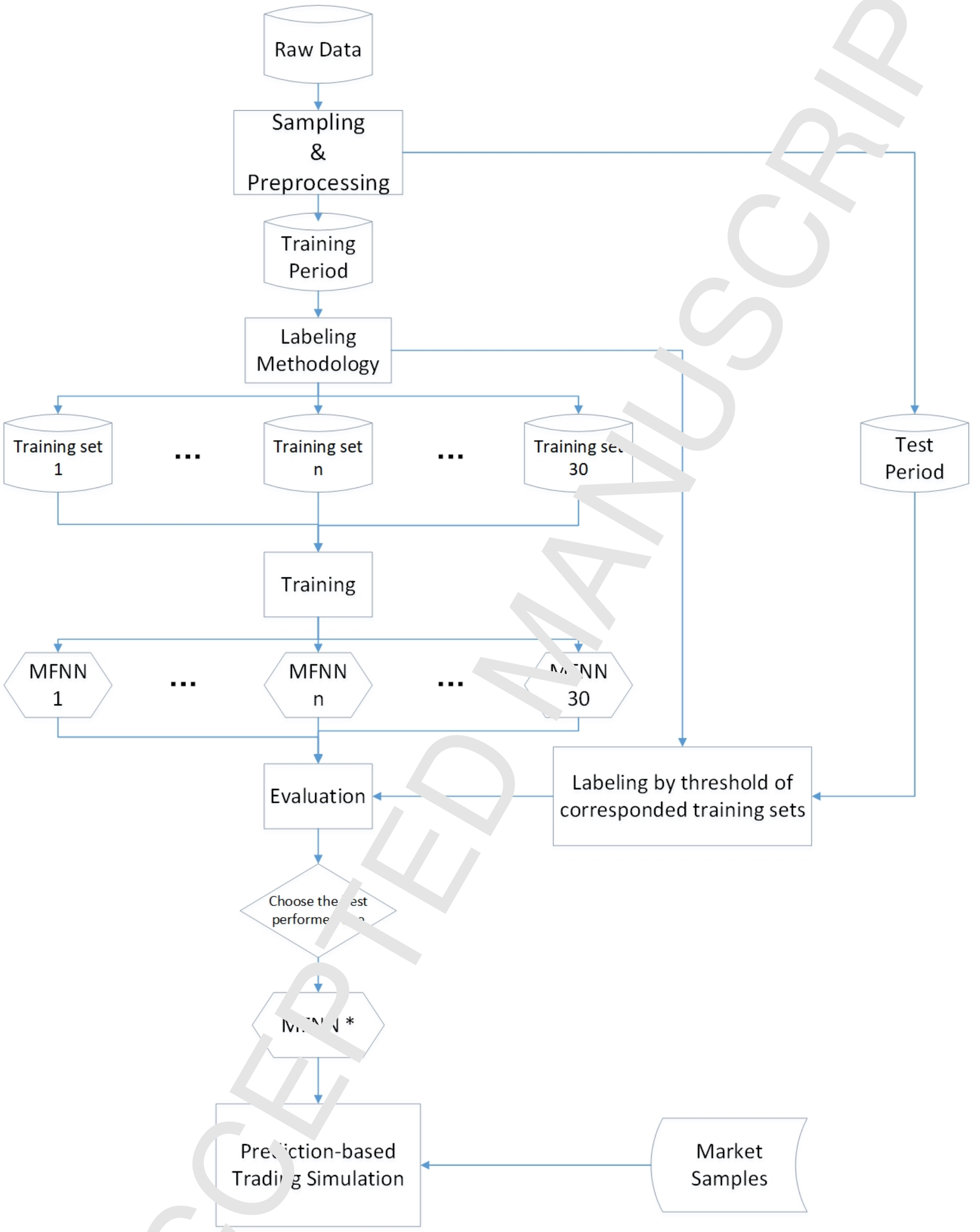


Figure 3: Experimental procedure for stock price movement prediction and prediction-based trading simulation.

Algorithm 1 Training Process For MFNN

Input: Preprocessed training sample Z_0,...,Z_N

Initialization: Initialize the parameters for the networks, $\rho = 0.5$, $\lambda = 0.5$, $T_{LastPeak} = 0$, $Acc_{val} = 0$, $N_{batch} = 400$, Epoch=0

```
repeat
    if T_LastPeak > 150 then Break
    else if T_LastPeak > 20 then
        Update the learning rate ρ_t=λ ∗ ρ_{t−1}
    end if
    for i=0,1,...,N%N_batch do
        Set batch B_i = {}
        for j=0,...,N_batch do
            Append Z_j to B_i
        end for
        Caculate ∇(U_t)_θ by averaging its gradient values among B_i
        Θ_t = Θ_{t−1} − ρ ∗ ∇(U_t)_θ / ‖∇(U_t)_θ‖
    end for
    Calculate the accuracy on validation test Acc_val by Θ_t
    if Acc_val > Acc_Peak then
        T_LastPeak=0
        Acc_Peak = Acc_val
    else
        T_LastPeak=T_LastPeak + 1
    end if
until Convergence
```

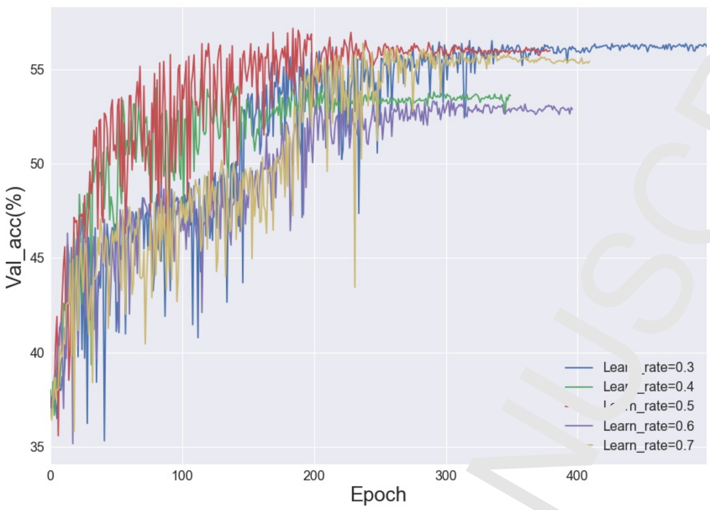
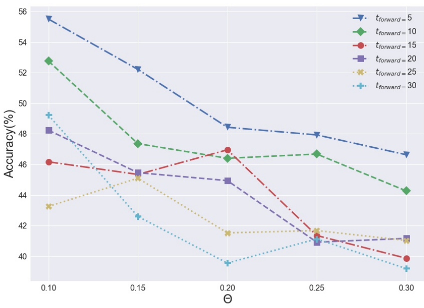


Figure 4: Training process with different initialized learning rate($t_{forward}$ = 5, θ=0.1).

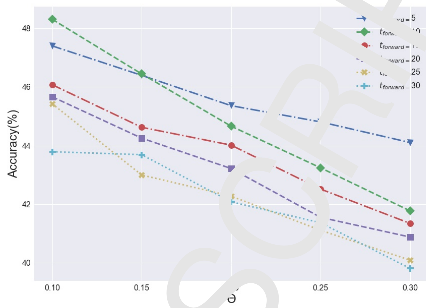
4.2. Result Discussion

Results of the accuracy on test sets of each model are presented in Fig 5. It can be concluded that the accuracy on the test set decreases whereas θ increases, which implies that samples with larger margins of a future rise or fall show stronger dependency between features and labels. However, changes on prediction windows $t_{forward}$ do not show obvious effects on model performance. For an in depth analysis of results of model performance, Table 3 presents the best five results of each model, average accuracy of the best 10 and 20 results of each model, from which it can be concluded that (1)models using deep learning methodologies have better capabilities of prediction since results of CNN, RNN, LTSM, and MFNN are all superior to those of other traditional methods. (2) Integration of multiple filters can enhance the efficiency of feature extraction since MFNN exhibits a better performance than both RNN and CNN, and it outperforms CNN (which shows a better performance than RNN and LSTM) by 6.28%.

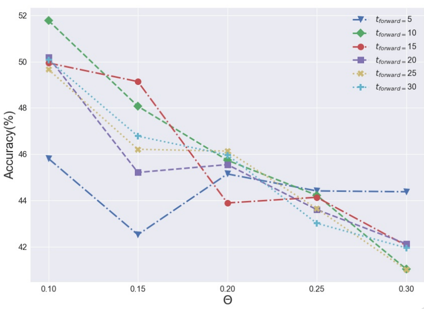
In previous part, training period and test period are split by the point December 31th, 2015, which makes the proportion of training period 70%, and 30 % for testing period. To



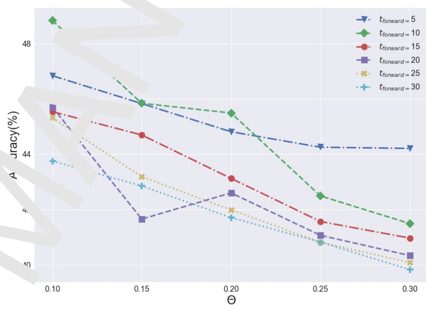
(a) MFNN



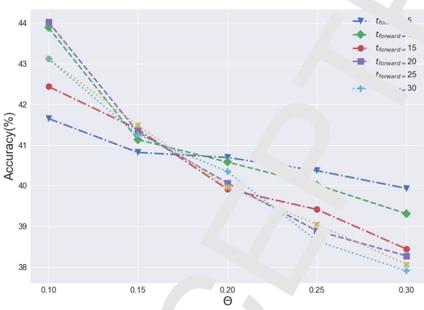
(b) RNN



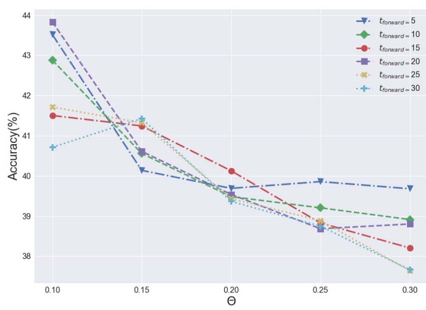
(c) CNN



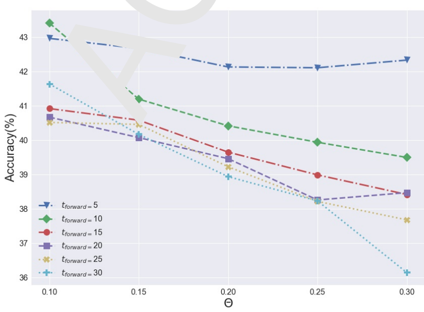
(d) LSTM



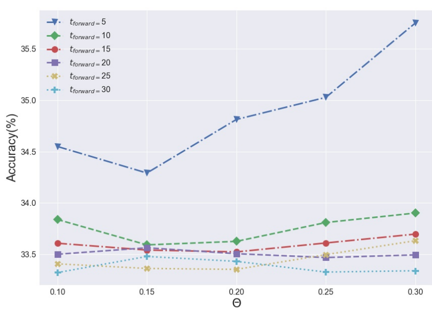
(e) SVM



(f) Random Forest



(g) Lofistic Regression



(h) Linear Regression

Figure 5: Performance of each model on 30 data sets.

Table 3: Best 5 results, average accuracy of the best 10 and the best 20 results of each model on 30 data sets

	1	2	3	4	5	Avg_Top10	Avg_Top20
MFNN	$t_{forward}$=5 $\theta = 0.1$ 55.50%	$t_{forward}$=10 $\theta = 0.1$ 52.76%	$t_{forward}$=5 $\theta = 0.15$ 52.21%	$t_{forward}$=30 $\theta = 0.1$ 49.22%	$t_{forward}$=5 $\theta = 0.2$ 48.43%	49.53%	47.27%
RNN	$t_{forward}$=10 $\theta = 0.1$ 48.31%	$t_{forward}$=5 $\theta = 0.1$ 47.40%	$t_{forward}$=10 $\theta = 0.15$ 46.45%	$t_{forward}$=5 $\theta = 0.15$ 46.40%	$t_{forward}$=15 $\theta = 0.1$ 45.67%	[illegible]	44.85%
CNN	$t_{forward}$=10 $\theta = 0.1$ 49.22%	$t_{forward}$=30 $\theta = 0.1$ 48.79%	$t_{forward}$=25 $\theta = 0.1$ 47.97%	$t_{forward}$=20 $\theta = 0.1$ 47.70%	$t_{forward}$=15 [illegible] 47.34%	48.80%	46.93%
LSTM	$t_{forward}$=10 $\theta = 0.1$ 48.283%	$t_{forward}$=5 $\theta = 0.1$ 46.84%	$t_{forward}$=10 $\theta = 0.15$ 45.84%	$t_{forward}$=5 $\theta = 0.15$ 45.83%	[illegible] $\theta = 0.1$ [illegible]	45.89%	44.45%
SVM	$t_{forward}$=20 $\theta = 0.1$ 44.03%	$t_{forward}$=10 $\theta = 0.1$ 43.89%	$t_{forward}$=25 $\theta = 0.1$ 43.13%	$t_{forward}$=30 $\theta = 0.1$ 43.12%	$t_{forward}$=15 $\theta = 0.1$ [illegible]	42.37%	41.38%
Logistic Regression	$t_{forward}$=10 $\theta = 0.1$ 43.41%	$t_{forward}$=5 $\theta = 0.1$ 42.97%	$t_{forward}$=5 $\theta = 0.15$ 42.67%	$t_{forward}$=5 $\theta = 0.3$ 42.33%	[illegible] $\theta = 0.2$ [illegible]	42.01%	41.04%
Random Forest	$t_{forward}$=20 $\theta = 0.1$ 43.83%	$t_{forward}$=5 $\theta = 0.1$ 43.52%	$t_{forward}$=10 $\theta = 0.1$ 42.88%	$t_{forward}$=25 $\theta = 0.1$ 41.71%	[illegible] $\theta = 0.1$ [illegible]	41.88%	40.83%
Linear Regression	$t_{forward}$=5 $\theta = 0.3$ 35.75%	$t_{forward}$=5 $\theta = 0.25$ 35.03%	$t_{forward}$=5 $\theta = 0.2$ 34.81%	$t_{forward}$=5 $\theta = 0.1$ 34.55%	$t_{forward}$=5 $\theta = 0.15$ 34.29%	34.33%	33.94%

Table 4: Performance of each model with different time windows for training period

	30%	40%	50%	60%	70%
MFNN	**38.51%**	**39.63%**	**[illegible]**	**44.65%**	**55.50%**
RNN	36.13%	35.83%	[illegible]	38.03%	48.31%
CNN	35.76%	37.17%	39.58%	42.29%	49.22%
LSTM	34.83%	35.97%	37.63%	41.15%	48.83%
SVM	35.62%	38.27%	38.32%	42.03%	44.03%
Logistic Regression	37.26%	38.25%	39.31%	41.97%	43.41%
Random Forest	35.34%	36.37%	38.03%	41.90%	43.83%
Linear Regression	33.34%	[illegible]	33.57%	33.79%	35.75%

evaluate the robustness of our model, we use different time windows to split the training period and test period. Training periods with proportion of 30%, 40%, 50%, 60%, 70%(test periods with proportion of 70%, 60%, 50%, 40%, 30% correspondingly) are used for sampling and generating data sets by the same process described in Section 3.1. Results are shown in Table 4, from which we can see that our MFNN still outperform each of other models with each proportion of training period.

Additionally, we utilize the Wilcoxon signed rank test[54] to further verify the statistical significance of MFNN and benchmark methods. The Wilcoxon signed rank test is a pairwise test that aims to detect significant differences between two algorithms. Table 5 presents the statistical significance of differences among models. It reports Wilcoxon signed rank

Table 5: Comparison of each two models using Wilcoxon signed rank test

	MFNN	RNN	CNN	LSTM	SVM	Logistic Regression	Random Forest	Linear Regression
MFNN		**1.78**	**1.15**	**1.36**	**1.57**	**1.57**	**1.78**	**2.61**
RNN			-0.73	-0.10	-0.73	-1.36	-0.1[illegible]	2.40
CNN				0.52	0.31	-0.10	0.52	2.40
LSTM					-0.52	-0.73	-0.31	1.98
SVM						0.10	0.73	2.19
Logistic Regression							0.73	2.61
Random Forest								2.19
Linear Regression								

test statistics for pairwise comparisons of a row model versus a column model. Our sign convention is that a positive statistic indicates the row model outperforms the column model. Results still verify that performance of networks can be improved by integrating multi-filters and our MFNN outperforms all other baseline models.

4.3. Market Simulation

To further evaluate the performance of MFNN on extreme market prediction, we simulate a prediction-based trading to test whether predictions made by MFNN can make profit. In the following simulation, a strategy is developed to mimic behaviors of a trader who makes trading decisions according to predictions of a model based on a very simple rule: if the model predicts that stock price is expected to have a rising trend, the system will buy the stock; if the model predicts that the stock price is expected to have a falling trend, the system will have a short position of the stock. To simulate this very simple strategy, we use models trained on the training sets to predict future trends of the CSI 300 in each minute from April 18 2016 to January 30 2017 and send trading signals according to predictions of models. As new market samples are updated every minute, predictions are made by models fed with new data, and subsequently, trading signals are sent.

If the model predicts a new sample to be positive category, our system will purchase 100,000 CNY worth of the stock in the next minutes with the opening price. We assume that 1,000,000 CNY are available at the start moment, and trading signals will not be executed when the cash balance is less than 100,000 CNY. Another assumption is zero transaction cost, which is common in similar evaluations. After a purchase, the system will hold the stock for $t_{forward}$ minutes corresponding to the prediction window of the model. If during that period the stock can be sold to make a profit of r_θ(threshold profit rate of labeling) or more, the system will sell immediately, otherwise, at the end of $t_{forward}$ minutes period, our system will sell the stock at the close price and takes a loss if necessary.

If the model predicts a new sample to be negative category, our system will have a short position of 100,000 CNY worth of stock ,which implies selling the stock we do not yet have in hopes of buying it later at a lower price. Similarly, the system will keep the position for $t_{forward}$ minutes. If during the period the system can buy the stock at $r_{1-\theta}$ lower than shorted, the system will close the short position by buying the stock to cover it. Otherwise, at the end of the period, the system will close the position similarly at the end of the period at the close price.

Accuracy of models can only measure the ability of the classification-based prediction, which correspond to ranges of future return, while what actually matters in market practice is the profitability, which is correlated to the amount of rise or fall. For example, profit made by two correctly predicted samples may be absorbed by loss caused by one incorrectly predicted sample, if the actual amount of the rise or fall in the future of the incorrectly predicted sample is sufficiently large. To evaluate performance of the MFNN in the market simulation, the total return (R) is measured by

$$R = (\frac{Portfolio_T}{Portfolio_{t_0}} - 1) \times 100\% \tag{8}$$

where $Portfolio_T$ denotes the value of all the assets in the account at the end of simulation period T and $Portfolio_{t_0}$ denotes the same at the start of simulation period t_0, which is initialized to 1,000,000 CNY as mentioned before. A higher return implies a better profitability performance of the model. Generally, the annual return rate (AR) is also measured by converting the total profit rate as

$$AR = (1 + R)^{\frac{244}{T_s}} - 1 \tag{9}$$

where T_s is the time span (days) over which the model is simulated and 244 is the average number of trading days for a year. Similarly, the daily winning rate (DWR) is also calculated to evaluate the stability of the model on prediction-based trading. In modern portfolio theory[55], the risk-adjusted profit is also widely used to evaluate stability of a trading system, so the Sharpe ratio(SR), which has been widely used in many trading related works[56], is also measured. The SR is defined as the ratio of average excess returns to the volatility of excess returns that is considered as risk, and SR is calculated as

$$SR = \sqrt{244} \times \frac{\overline{r_e}}{\sigma_e} \tag{10}$$

where

$$\overline{r_e} = \frac{1}{n}\sum_{i=1}^{n}[r_p(i) - r_f(i)] \tag{11}$$

$$\sigma_e = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}[r_p(i) - r_f(i) - \overline{r_e}]^2} \tag{12}$$

where $\overline{r_e}$ and σ_e denote the average daily excess returns and volatility of the daily excess returns during the simulation period, respectively, $r_p(i)$ and $r_f(i)$ denote the returns from the trading strategy and risk-free rate of interest on the i-th trading day, and n denotes the number of trading days during the simulation. A large SR corresponds to a high profit under a unit risk and higher stability. To evaluate the risk of the trading system, the maximum drawdown (MDD) is also introduced, which is defined as the maximum loss from a peak to a trough in the portfolio before a new peak is attained. The annual volatility (V) is also used to evaluate the risk, and is measured as

$$V = \sqrt{\frac{244}{n-1}\sum_{i=1}^{n}[r_p(i) - \overline{r_p}]^2}. \tag{13}$$

where $\overline{r_p}$ denotes the average return from the trading strategy.

Details of the portfolios' net value curves during simulations are shown in Fig 6. Results of each model in market simulations are presented in Table 6, where all the indicators mentioned before are compared.

First, we can see from Table 6 that all prediction-based simulations are significantly more profitable than the randomly buy and sell strategy. It implies that prediction models involved can capture suitable trading points to make profits. Among these prediction models, all simulations based on predictions from machine learning and deep learning models result in better returns and annual returns than linear regression. It indicates that non-linear models show better profitability than the traditional statistical one. Specifically, most deep

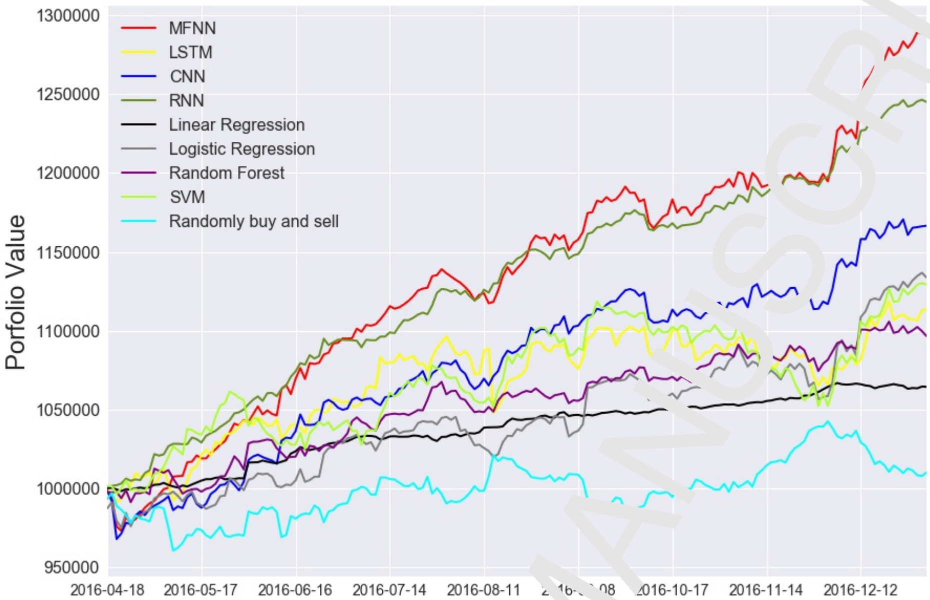


Figure 6: Prediction-based market simulations.

learning methodologies-based(except LSTM) simulations significantly outperform those of machine learning and statistical methods at profitability, and simulations based on the MFNN outperform the best result of machine learning (logistic Regression) by 15.41% and linear regression by 22.41% at returns.

Second, the traditional statistical model still shows a better performance in terms of the risk control, and the annual volatility of the linear regression-based simulation (1.86%) is only one-third of that of the random forest, which shows the least volatility among the machine learning models (5.89%). Similarly, the maximum drawdown of the simulation based on linear regression is also much less than any machine learning model. This implies that trading based on linear regression will be less risky than that based on machine learning models, but as demonstrated before, it is also a conservative strategy that may lead to less

Table 6: Market simulation results

	Hyper-parameter	R	AR	SR	V	MDD	DWR
MFNN	$\theta = 0.1$ $t_{forward} = 5$	**28.78%**	**42.28%**	4.49	7.41%	-2.56%	65.14%
RNN	$\theta = 0.1$ $t_{forward} = 10$	24.50%	35.74%	**6.42**	4.42%	-1.09%	**67.43%**
CNN	$\theta = 0.1$ $t_{forward} = 10$	20.50%	29.70%	3.3595	7.14%	-2.01%	65.14%
LSTM	$\theta = 0.1$ $t_{forward} = 10$	11.53%	16.17%	1.40	9.42%	-4.44%	58.29%
Linear Regression	$\theta = 0.3$ $t_{forward} = 5$	6.37%	8.99%	3.4056	**1.86%**	**-0.34%**	64.57%
Logistic Regression	$\theta = 0.1$ $t_{forward} = 10$	13.37%	19.12%	1.9792	7.84%	-3.11%	62.29%
Random Forest	$\theta = 0.1$ $t_{forward} = 20$	9.65%	13.71%	1.8221	5.89%	-1.80%	58.86%
SVM	$\theta = 0.1$ $t_{forward} = 20$	12.93%	18.47%	1.7140	8.79%	-5.94%	53.71%
random buy and sell	-- $t_{forward} = 10$	1.03%	1.44%	-0.0870	7.08%	-3.65%	48.57%

returns.

In the view of the stability, the most stable result from RNN reaches 6.42 Sharpe ratio and 67.43% daily winning rate, and MFNN yields the second best result. We can conclude from these results that models using deep learning methodologies have better capabilities of capturing profitable and stable signals than traditional methods. But when using integrated network structures, the quality of captured signals can be affected by the way that units are integrated.

5. Conclusion

In this paper, we proposed an end-to-end model named multi-filters neural network using deep learning methodologies for feature engineering on multivariate financial time series, and classification-based prediction. Feature maps extracted by multi-filters were used for classification-based extreme market prediction, and models using deep learning methodologies were verified to be superior to the traditional machine learning and statistical methods on the prediction task. The best prediction result from the MFNN outperformed the best machine learning method and statistical method for 11.47% and 19.75%. The integration of filters and purposely designing of the network enhanced the accuracy for 7.19% and 6.28% compared with RNN and CNN, respectively. In the market simulation, models with deep learning methodologies presented a better profitability than any other baseline. Our proposed MFNN was 15.41% better than the best result of traditional machine learning methods and 22.41% better than the statistical method at returns.

Although the effectiveness of deep learning methodologies on extreme-market prediction was verified, there are still some promising future directions. The proposed method is superior in terms of the profitability when simulated in prediction-based trading, but risk and stability are to be improved. The RNN achieved the best result of the Sharpe ratio, which implied that deep learning methodologies has the capability to capture stable signals, but the quality and characteristics of signals are potentially affected by the way filters are integrated. Therefore, in our future works, we hope to explore how the way of integration affects the quality of trading signals in terms of the risk, profitability, and stability. Second, we only use historical market data of stocks to predict future trends. Considering the extensibility of neural networks and deep learning methodologies, we can further try to design a specific network to extract information from multi-sources information (macroeconomic indicators, news, and market sentiment) to make prediction.

6. Acknowledgement

This research was partly supported by the grants from the National Natural Science Foundation of China (No. 71771204, 71331005, 91546201).

References

[1] K. J. Kim, I. Han, Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index, Expert Systems with Applications 19 (2) (2000) 125–132.

[2] H. Gunduz, Y. Yaslan, Z. Cataltepe, Intraday prediction of borsa istanbul using convolutional neural networks and feature correlations, Knowledge-Based Systems 137 (2017) 138–148.

[3] M. Hagenau, M. Liebmann, D. Neumann, Automated news reading: Stock price prediction based on financial news using context-capturing features, Decision Support Systems 55 (3) (2013) 685–697.

[4] X. Ding, Y. Zhang, T. Liu, J. Duan, Deep learning for event-driven stock prediction, in: Proceedings of the 24th International Conference on Artificial Intelligence, AAAI Press, 2015, pp. 2327–2333.

[5] Y. Shynkevich, T. McGinnity, S. Coleman, A. Belatreche, Y. Li, Forecasting price movements using technical indicators: investigating the impact of varying input window length, Neurocomputing.

[6] J. Patel, S. Shah, P. Thakkar, K. Kotecha, Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques, Expert Systems with Applications 42 (1) (2015) 259–268.

[7] J. Sen, T. D. Chaudhuri, A robust predictive model for stock price forecasting, in: International Conference on Business Analytics and Intelligence, 2017.

[8] G. S. Atsalakis, K. P. Valavanis, Surveying stock market forecasting techniques part ii: Soft computing methods, Expert Systems with Applications 36 (3) (2009) 5932–5941.

[9] Y. Pan, Z. Xiao, X. Wang, D. Yang, A multiple support vector machine approach to stock index forecasting with mixed frequency sampling , Knowledge-Based Systems 122 (2017) 90–102.

[10] T. Xiong, C. Li, Y. Bao, Z. Hu, L. Zhang, A combination method for interval forecasting of agricultural commodity futures prices, Knowledge Based Systems 77 (C) (2015) 92–102.

[11] F. Shen, J. Chao, J. Zhao, Forecasting exchange rate using deep belief networks and conjugate gradient method, Neurocomputing 167 (2015) 243–253.

[12] R. Singh, S. Srivastava, Stock prediction using deep learning, Multimedia Tools and Applications 76 (18) (2017) 18569–18584.

[13] K. Chen, Y. Zhou, F. Dai, A lstm-based method for stock returns prediction: A case study of china stock market, in: Big Data (Big Data), 2015 IEEE International Conference on, IEEE, 2015, pp. 2823–2824.

[14] Y. Deng, F. Bao, Y. Kong, Z. Ren, Q. Dai, Deep direct reinforcement learning for financial signal representation and trading, IEEE Transactions on Neural Networks & Learning Systems 28 (3) (2017) 653.

[15] J. Sirignano, Deep learning for limit order books, arXiv preprint arXiv:1601.01987.

[16] M. P. Taylor, H. Allen, The use of technical analysis in the foreign exchange market, Journal of international Money and Finance 11 (3) (1992) 304–314.

[17] A. Gunasekarage, D. M. Power, The profitability of moving average trading rules in south asian stock markets, Emerging Markets Review 2 (1) (2001) 17–33.

[18] C. Lee, B. Swaminathan, Price momentum and trading volume, the Journal of Finance 55 (5) (2000) 2017–2069.

[19] C. Lento, N. Gradojevic, C. Wright, Investment information content in bollinger bands?, Applied Financial Economics Letters 3 (4) (2007) 263–267.

[20] A. D. Back, A. S. Weigend, A first application of independent component analysis to extracting structure from stock returns, International journal of neural systems 8 (04) (1997) 473–484.

[21] S. T. Roweis, L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, science 290 (5500) (2000) 2323–2326.

[22] B. Hambly, T. Lyons, Uniqueness for the signature of a path of bounded variation and the reduced path group, Annals of Mathematics (2010) 109–167.

[23] H. Boedihardjo, X. Geng, T. Lyons, D. Yang, The signature of a rough path: uniqueness, Advances in Mathematics 293 (2016) 720–737.

[24] T. Lyons, Rough paths, signatures and the modelling of functions on streams, arXiv preprint arXiv:1405.4537.

[25] C. Kocak, Arma(p,q) type high order fuzzy time series forecast method based on fuzzy logic relations, Applied Soft Computing.

[26] G. Zumbach, L. Fernndez, Option pricing with realistic arch processes, Quantitative Finance 14 (1) (2014) 143–170.

[27] Z. Lin, Modelling and forecasting the stock market volatility of sse composite index using garch models, Future Generation Computer Systems.

[28] N. lk, D. Kuruppuarachchi, O. Kuzmicheva, Stock market's response to real output shocks in eastern european frontier markets: A varwal model, Emerging Markets Review.

[29] L. G. Valiant, A theory of the learnable, Communications of the ACM 27 (11) (1984) 1134–1142.

[30] R. L. Rivest, Learning decision lists, Machine learning 2 (3) (1987) 229–246.

[31] T. Zhou, S. Gao, J. Wang, C. Chu, Y. Todo, Z. Tang, Financial time series prediction using a dendritic neuron model, Knowledge-Based Systems 105 (C) (2016) 214–224.

[32] L. Zhou, Y. W. Si, H. Fujita, Predicting the listing statuses of chinese listed companies using decision trees combined with an improved filter feature selection method, Knowledge-Based Systems 128.

[33] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[34] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: Acoustics, speech and signal processing (icassp), 2013 ieee international conference on, IEEE, 2013, pp. 6645–6649.

[35] W. Yang, L. Jin, H. Ni, T. Lyons, Rotation-free online handwritten character recognition using dyadic path signature features, hanging normalization, and deep neural network, in: Pattern Recognition (ICPR), 2016 23rd International Conference on, IEEE, 2016, pp. 4083–4088.

[36] Y. Lcun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[37] R. J. Williams, D. Zipser, A learning algorithm for continually running fully recurrent neural networks, MIT Press, 1989.

[38] K. Cho, B. Van Merrienboer, D. Bahdanau, Y. Bengio, On the properties of neural machine translation: Encoder-decoder approaches, Computer Science.

[39] S. Hochreiter, J. Schmidhuber, Long short-term memory., Neural Computation 9 (8) (1997) 1735–1780.

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.

[41] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, G. Hu, Attention-over-attention neural networks for reading comprehension, arXiv preprint arXiv:1607.04423.

[42] K. J. Han, S. Hahm, B.-H. Kim, J. Kim, I. Lane, Deep learning-based telephony speech recognition in the wild, in: Proc. Interspeech, 2017, pp. 1323–1327.

[43] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[44] S. Hochreiter, The vanishing gradient problem during learning recurrent neural nets and problem solutions, International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6 (02) (1998) 107–116.

[45] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult., IEEE Transactions on Neural Networks 5 (2) (2002) 157–166.

[46] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: International Conference on Machine Learning, 2013, pp. 1310–1318.

[47] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research 15 (1) (2014) 1929–1958.

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

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

[50] M. Dixon, Sequence classification of the limit order book using recurrent neural networks, Journal of Computational Sciencedoi:https://doi.org/10.1016/j.jocs.2017.08.018.
URL http://www.sciencedirect.com/science/article/pii/S1877750317309675

[51] M. Rimer, T. Martinez, Softprop: softmax neural network backpropagation learning, in: Neural Networks, 2004. Proceedings. 2004 IEEE International Joint Conference on, Vol. 2, IEEE, 2004, pp. 979–983.

[52] HechtNielsen, Theory of the backpropagation neural network, Neural Networks 1 (1) (1988) 445–445.

[53] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, N. De Freitas, Learning to learn by gradient descent by gradient descent, in: Advances in Neural Information Processing Systems, 2016, pp. 3981–3989.

[54] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.

[55] H. Markowitz, Portfolio selection, The journal of finance 7 (1) (1952) 77–91.

[56] O. Ledoit, M. Wolf, Robust performance hypothesis testing with the sharpe ratio, Journal of Empirical Finance 15 (5) (2008) 850–859.