Forecasting stock price movement: new evidence from a novel hybrid deep learning model

Yang Zhao and Zhonglu Chen
Southwest Jiaotong University, Chengdu, China

Received 30 May 2021
Revised 2 July 2021
12 July 2021
Accepted 14 July 2021

Abstract

Purpose – This study explores whether a new machine learning method can more accurately predict the movement of stock prices.

Design/methodology/approach – This study presents a novel hybrid deep learning model, Residual-CNN-Seq2Seq (RCSNet), to predict the trend of stock price movement. RCSNet integrates the autoregressive integrated moving average (ARIMA) model, convolutional neural network (CNN) and the sequence-to-sequence (Seq2Seq) long–short-term memory (LSTM) model.

Findings – The hybrid model is able to forecast both linear and non-linear time-series component of stock dataset. CNN and Seq2Seq LSTMs can be effectively combined for dynamic modeling of short- and long-term-dependent patterns in non-linear time series forecast. Experimental results show that the proposed model outperforms baseline models on S&P 500 index stock dataset from January 2000 to August 2016.

Originality/value – This study develops the RCSNet hybrid model to tackle the challenge by combining both linear and non-linear models. New evidence has been obtained in predicting the movement of stock market prices.

Keywords Stock price movement, RCSNet, ARIMA, CNN, LSTM, S&P 500 index

Paper type Research paper

1. Introduction

It is widely acknowledged that predicting stock price movement trend is a difficult financial problem. The relatively precise prediction of a stock's future price movement will maximize profits of investors. However, for the variability and instability of stock market, it is still an open question.

Generally, traditional stock forecast methods mainly include two categories, which are the technical analysis and the fundamental analysis. Fundamental analysis analyzes the company financial environment, operations, macroeconomic and microeconomic indicators to predict stock price. In fact, in recent years, as the massive growth of internet content, the development of natural language processing (NLP) technique enables investors to capture market movement trends online. However, the quality of online content of stock market is not guaranteed and could not exclude low-quality content and even including fake news and comments. That is to say, fundamental analysis based on methods are difficult to model. Therefore, in this work, the technical analysis methods is concerned.

Technical analysis methods based on historical time-series data. Stock price movement prediction is deemed as a time-series forecast problem. Generally, rather than the original stock forecast, stock price time-series data are abstracted into two components, which are a linear and non-linear. Hence, we could research both linear and non-linear stock forecasts. Classical linear time-series models, including the vector autoregression (VAR) model and

JEL Classification — C52, G11, G12

Journal of Asian Business and Economic Studies
Vol. 29 No. 2, 2022
pp. 91-104
Emerald Publishing Limited
2515-964X
DOI 10.1108/JABES-05-2021-0061

autoregressive integrated moving average (ARIMA) model (Ltkepohl, 2005), have been proved effective for linear time-series forecast in many fields such as economic (Hamilton, 1989) and power price forecasting (Contreras *et al.*, 2003), meanwhile have poor performance for non-linear forecast. On the other hand, traditional non-linear time forecast models, like support vector machine (SVM) (Hossain *et al.*, 2009), back-propagation (BP) neural network (NN) (Maier and Dandy, 2000), specialize in describing the non-linear data, while performing worse in linear data and the long term in time-series forecast.

A successful stock time-series forecasting model should satisfy two points: Firstly, the model should be suited to both linear and non-linear data since the stock movement data contain both. Secondly, the model could capture multi-frequency patterns (short and long term) for accurate non-linear part predictions.

In recent years, based on the fast growth of the computing ability of computers and massive data, deep learning methodologies have been wildly adopted in speech recognition (Hinton *et al.*, 2012), image classification, machine transition (Cho *et al.*, 2014) and other area, which are composed of kinds of derivatives of artificial neural network (ANN), such as convolutional neural network (CNN) and recurrent neural network (RNN). CNN models perform outstanding image recognition performance by extracting local features at various granularity levels from input images. RNN (as shown in Figure 1(a)) is a type of non-linear model, and what is more, it has the ability to model long-term dependencies, which makes it is well suited to both non-linear data and long-term dependencies. However, long-term dependencies are hard to detect by traditional RNN, suffering from the gradient vanishing (Bengio *et al.*, 2002) problem. To solve this problem, long–short-term memory (LSTM) units (as shown in Figure 1(b)) (Hochreiter and Schmidhuber, 1997) and the gated recurrent unit (GRU) (Chung *et al.*, 2014) have achieved great success in various domains like computer version and NLP.

With the recent success of deep learning, we develop the residual-CNN-Seq2Seq (RCSNet) hybrid model to tackle the challenge by combining both linear and non-linear models. In Figure 2, the model consists of ARIMA layer, CNN layer, Seq2Seq LSTM layer and fully connected (FC) layer. Hence, the model is capable of forecasting both linear and non-linear time series and could also proceed the frequency patterns separately in the Seq2Seq LSTM layer. Overall, based on the above characteristics, we apply it to forecast the stock movement trend.

The rest of this work is planned as follows: In Section 2, we present the preliminary and related works. Section 3 provides the details of the RCSNet model proposed in this study. The

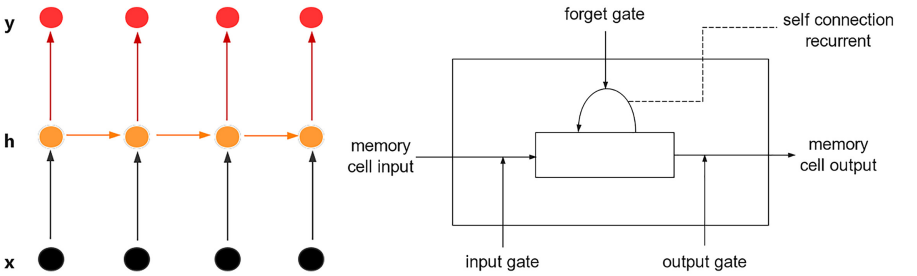

Figure 1.
The architecture of a standard RNN and LSTM

Note(s): (a) A standard RNN contains a single layer, in which x denotes an input vector; y denotes the output vector; h is the memory, which is computed by the past memory and current input. (b) A standard LSTM memory cell that can store information until inputs gate makes them forget or overwrite it with new input. The cell with gates can determine whether to output this information or just keep it

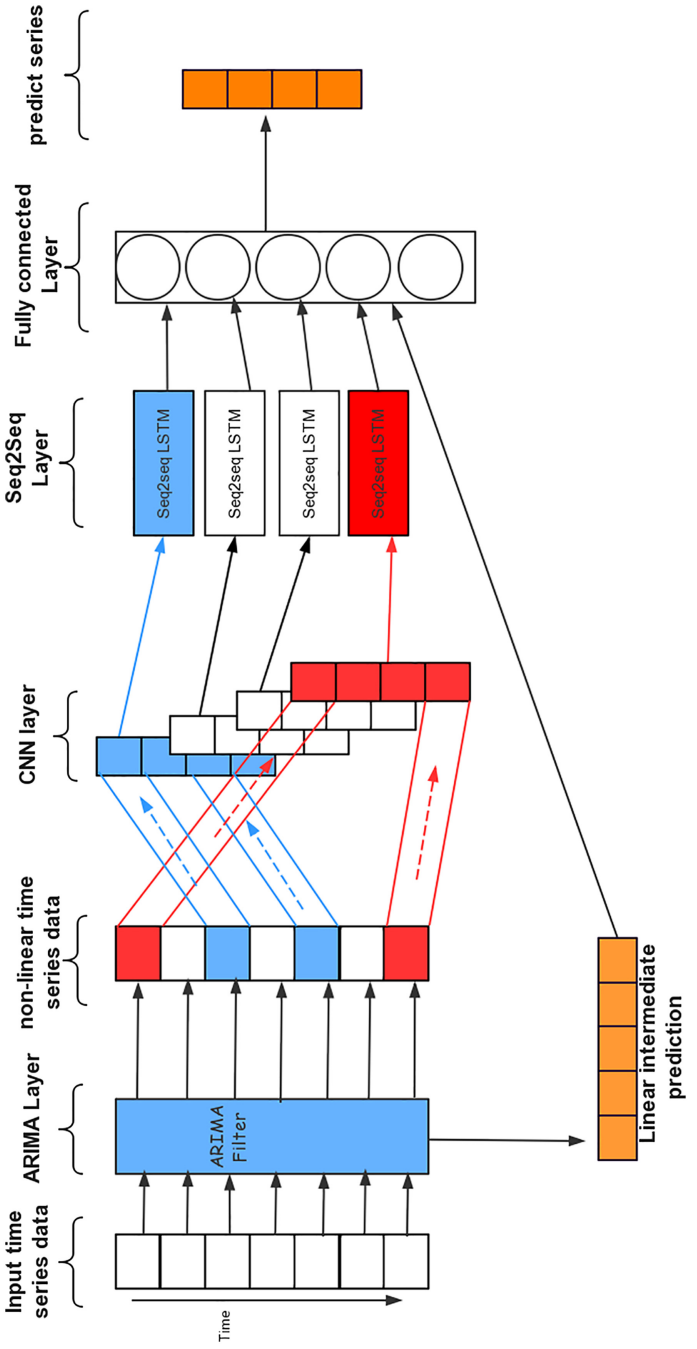

Figure 2.
An overview of RCCSNet NN, which has the ARIMA, CNN, Seq2Seq LSTM and FC layers

comparison between the results of our model and traditional models is shown in Section 4. The conclusion and future work are displayed in Section 5.

2. Preliminary and related works

This study involves three groups model of research primarily, that is, linear models, non-linear models and hybrid models. Therefore, the traditional linear models, non-linear models and hybrid models for stock movement forecast are mentioned in this section.

2.1 Autoregressive model

ARIMA model is one of the most commonly used autoregressive models, which has been extensively studied (Hamilton, 1989; Contreras *et al.*, 2003). These linear models can be applied effectively to forecast the behavior of economic and financial (Tsay, 2005; Sims, 1980). The ARIMA model is generally expressed as ARIMA (*p, d, q*), where *p, d* and *q* are parameters composed by non-negative integers. *p* is the set for the number of time lags of the autoregressive model (the order of AR terms); *d* denotes the order of differences, and *q* indicates the order of the moving-average model. The ARIMA model is formulated as:

$$\nabla^d x_t = \sum_{i=1}^{p} \phi_i \nabla^d x_{t-i} + \sum_{j=1}^{q} \theta_j \varepsilon_{t-j} + \varepsilon_t \tag{1}$$

where ϕ_i denotes the parameter of the autoregressive part of the model, θ_j is the parameter of the moving average part of the model. In addition, ϵ_t is the error term at time t, and the ϵ_{t-j} presents the error terms, which are usually supposed to be independent, sampled from a normal distribution where the mean is zero. x_t represents the forecasting result of autoregressive method.

2.2 Back-propagation neural network model

A back-propagation neural network (BPNN) is one of most essential ANNs proposed by Rumelhart *et al.* in the year 1986 (Rumelhart *et al.*, 1988). It is widely used in prediction and forecasting. JZ Wang *et al.* have proposed the wavelet de-noising-based back-propagation (WDBP) NN (Wang *et al.*, 2011) to predict the stock prices. M Gken *et al.* have provided a NN with selecting the most relevant technical indicators for stock market forecasting (Gken *et al.*, 2016) in 2016. In this study, we assume that the BPNN is composed of non-linear layers with sigma function $\sigma()$ as an active function. We design three layers, including input, hidden and output layer. Each inner layer is fully linked to the previous one.

$$h_i = \sigma(W_i h_{i-1} + b_i) \tag{2}$$

where W_i is a the ith weight matrix (parameter matrix) for ith layer, b_i is bias term. After training, h_i represents the forecasting result of the BPNN model.

2.3 Recurrent neural network model

RNN is a sort of ANN, which connects itself via sequential data and forms a directed circulation. This enables it to display dynamic temporal behavior. There are variety of types of RNN, including simple RNN, LSTM and so on. The RNN model is capable of capturing long-term and non-linear internal rule of the data with the gated memory cells. Hence, it could memorize the long-term content of time-series data. EW Saad *et al.* have exploited a type of time delay, recurrent and probabilistic NNs for stock price movement trend (Saad *et al.*, 1998). CM Kuan *et al.* have researched the out-of-sample forecasting performance of RNNs via empirical foreign exchange rate data (Kuan and Liu, 1995). An adaptive "forget gate," presented by F Gers, allows an LSTM cell to learn to rebuild itself at a suitable time, thereby releasing internal resources during forecasting (Gers *et al.*, 2000).

2.3.1 Simple recurrent neural network model. A simple RNN model can deal with time series of inputs by utilizing its internal memory. The input is propagated in the manner of a standard feed-forward at each time step. The fixed back connections cause the context units to hold a copy of the previously hidden units values all the time. The equations are given by:

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h)$$
$$y_t = \sigma(W_y h_t + b_y) \tag{3}$$

In the model, $X = (x_1, x_2, \ldots, x_t)$, h_t and y_t represent the input vector, hidden vector and output vector (prediction), respectively. (W, U, b) are parameter matrices and vector. Moreover, σ is the activation function. y_t denotes the output of prediction.

2.3.2 Simple long–short-term memory model. The simple LSTM model is composed of LSTM units. There are a cell, an input gate, an output gate and a forget gate in an LSTM unit. The equations are given by:

$$\begin{aligned} f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\ i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\ o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\ \widehat{c}_t &= \tan h(W_c x_t + U_c h_{t-1} + b_c) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \widehat{c}_t \\ h_t &= o_t \odot \tan h(c_t) \end{aligned} \tag{4}$$

where W_f, W_i, W_o and U_f, U_i, U_o gather the weights of the input and recurrent connections, respectively; the input gate i_t, output gate o_t, $\widehat{c}_t$ indicates the updated unit status value and forget gate f_t and the memory cell c_t count on the activation being calculated, which is the sigmoid function; h_t denotes the output of the model for forecast.

2.3.3 Hybrid model. A hybrid model is a model that combines two or more base models of various kinds to reach a better model, which has been extensively researched. Ping Feng Pai *et al.* researched the hybrid model combining ARIMA with SVMs based on time-series data (Pai and Lin, 2005). They proposed a hybrid methodology to predict stock prices via taking advantage of the superiority of the ARIMA and SVM models. Ashu Jain *et al.* have developed a hybrid time-series NN model, which can make use of the advantages of traditional time-series methods and ANNs (Jain and Kumar, 2007). A hybridized framework of SVM with K-nearest neighbor approach for the Indian stock market indices prediction proposed by Nayak *et al.* (2015). FA Gers *et al.* designed a hybrid model that combined a time window-based MLP and LSTM for forecasting (Gers *et al.*, 2001) in 2002. A time window-based MLP was primarily trained, then its weights were freezed, and finally, LSTM was employed to decrease the forecast error in this model.

As discussed above, overall, the related works ignore the following:

(1) The stock time-series data have both linear and non-linear dependency component.

(2) The non-linear component of stock time-series data contains long and short patterns.

Hence, we design a hybrid model for solving these problems to forecast the trend of the stock movement.

3. Residual-CNN-Seq2Seq model

This section discusses the details of the model for stock time-series prediction. RCSNet is comprised of the ARIMA, CNN, Seq2Seq LSTM and FC layers. The objective function and the optimization strategy also are discussed.

3.1 Problem statement and algorithm

The problem is typically described as follows: given a target series $X = (x_1, x_2, \cdots, x_t)$, we need to train a model that aims to learn the internal rule of mapping the current value of target series X_{target} to the value of predicted data X_{predict}, $X_{\text{predict}} = F(x_1, x_2, \cdots, x_t)$. To find the most suitable parameters for RCSNet is crucial for the stock movement forecasting.

RCSNet firstly extracts the linear dependence component by the ARIMA model. Then, the residual error time series (target data subtracts ARIMA's forecast output) is seen as the non-linear component. The CNN layer is used to extract short- and long-term trading patterns. After the CNN layer, the residual error time series of different patterns is predicted by the Seq2Seq layer to generate non-linear intermediate forecast results. Finally, the FC layer jointly outputs the final forecast results by using linear and non-linear intermediate results. Generally, the RCSNet can be described as below:

$$\begin{aligned} \widehat{x}_{t+h} &= \text{ARIMA}(x_t) \\ e_{t+h} &= \widehat{x}_{t+h} - x_{t+h} \\ \tilde{e}_{t+h} &= \text{CNN}(e_{t+h}) \\ \widehat{e}_{t+h} &= \text{Seq 2 Seq}(\tilde{e}_{t+h}) \\ \widehat{y}_{t+h} &= \text{Fully Connect}(\widehat{e}_{t+h}, \widehat{x}_{t+h}) \end{aligned} \tag{5}$$

The linear model takes input x_t up to time step t, producing the output $\widehat{x}_{t+h}$, which is what the ARIMA model predicts for a h horizon. x_{t+h} denotes the actual value $t + h$. We get the linear models residual e_{t+h} by subtraction of $\widehat{x}_{t+h}$ and the actual value. e_{t+h} has multi-frequency trading patterns, and we extract the sub-frequency trading pattern by the CNN layer. A Seq2Seq LSTM is utilized to model the non-linear residuals, whose input is the sub-frequency residual $\widehat{e}_{t+h}$. The objective of the Seq2Seq LSTM is to predict the error that the linear model will produce in its next forecast for time step $t + h$. The final model output is then generated by combing the forecast $\widehat{e}_{t+h}$ with the forecast of the linear models $\widehat{x}_{t+h}$. The overall algorithm for RCSNet:

Algorithm 1 RCSNet training algorithm

Require: Input: $X^{all} = (X^1, X^2, \cdots, X^N) \in R^{N \times T}$, Parameters

Ensure: To predict final result sets $y^{final} = ({y_1}^{final}, {y_2}^{final}, \cdots, {y_N}^{final})$,

The target result is $x^{target} = ({x_1}^{target}, {x_2}^{target}, \cdots, {x_N}^{target})$

while Input series is not finished ($i \leq N$) **do**
  linear_predict ← *ARIMA*(x^i)
  non_linear ← x^{target} - *linear_predict*
  if *non_linear component exists* **then**
    Input ← *non_linear*
    hidden_state ← *encoder* (*Input*)
    decode_state ← *decoder* (*hidden state*)
    To generate: *non_linear_predict*
  end if
  combine_prediction:
  linear_predict and *non_linear_predict* intermediate results to Fully connected layer to get y^{final} series
  To minimize the loss of $\sum({y_i}^{final} - {x_i}^{target})$
  update Parameters to decrease the loss
  $i \leftarrow i + 1$
end while

3.2 Autoregressive integrated moving average layer

RCSNet uses the ARIMA model as the linear filter. Seq2Seq LSTM model is trained by calculating the residual of the linear model (non-linear component). As Figure 3 shows, the ARIMA filter can distinguish the linear component from historical stock data series, and then we could obtain the non-linear data series.

3.3 Convolutional neural network layer

The second layer of RCSNet, designed to extract short and long patterns in the time dimension, is a convolutional network layer without pooling. The CNN layer is comprised of multiple filters of height w that is set to equal the number of variables. The k-th filter sweeps through the input matrix X. The long-term patterns reflect the trading frequency of season, month, while the short-term patterns express the trading frequency of week, day. Because of taking different kinds of trading frequency patterns into consideration, it is likely for us to forecast the time series precisely.

$$h_k = W_k \times X \tag{6}$$

where $\times$ indicates the convolution computation, and the output h_k is a vector that only has one column in our work. This work fills each vector h_k of length T with zeros to the left of input matrix X. The size of the output matrix of the convolutional layer is $m_c * T$, where m_c represents the amounts of filters.

3.4 Seq2Seq long–short-term memory layer

Inspired by the success of machine translation (Cho *et al.*, 2014), we have recognized the power of the Seq2Seq model in NLP. More specifically, two crucial components make up the standard Seq2Seq model, one is an encoder and the other is a decoder. The former maps the source input x to a vector representation, while the latter produces an output series based on the source. Both the encoder and decoder are LSTMs. By transmitting the last memory condition of the encoder to the decoder as the original memory condition, the encoder is

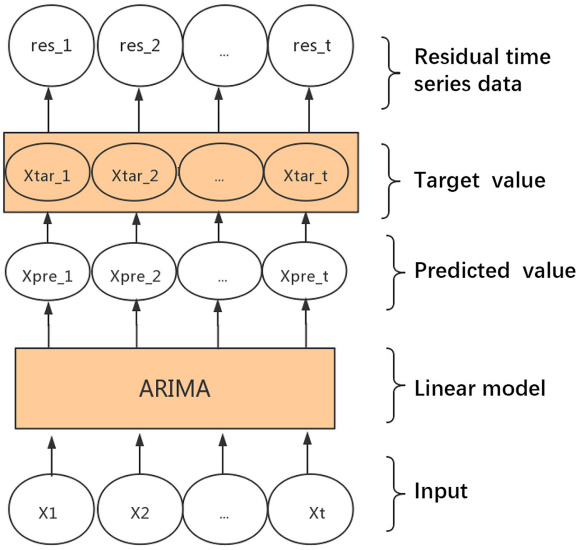

Figure 3.
ARIMA filter separates linear and non-linear time-series components

capable of accessing information from the encoder. Input and output generally apply various LSTMs that possess their own compositional parameters to capture various compositional patterns. We apply a Seq2Seq LSTMs model to address the non-linear time-series forecasting issue as the third layer. Figure 4 shows the model constituted by an encoder and a decoder. In the encoder part, the input LSTM mechanism is used for inputting into series data. In the decoder part, an output LSTM mechanism is employed to decode the hidden states of encoder across all time steps before.

3.4.1 Encoder. The encoder module is substantially an LSTM. It can encode the input series into a characteristic representation. For example, given N input series $X = (x_1, x_2, \cdots, x_t) \in R^{N \times T}$, in which t denotes the window size of the sequence. We make use of the encoder in learning a mapping function from x_t to h_t (at time step t), $h_t = f^{enc}(h_{t1}, x_t)$, where $h_t \in R^M$ is the encoder hidden state at time t, M indicates the size of the hidden state and moreover, f^{enc} represents a non-linear activation function. At time t, each LSTM unit owns a memory cell with the state $c^{enc}(t)$. $\widehat{c}^{\,enc}(t)$ indicates the updated unit status value for encoder. There will be three sigmoid gates that control access to the memory cell: forget gate $f^{enc}(t)$, input gate $i^{enc}(t)$ and output gate $o^{enc}(t)$. The LSTM unit update is summed up as follows:

$$
\begin{aligned}
f^{enc}(t) &= \sigma\left(W_f^{enc} x_t + U_f^{enc} h_{t-1}^{enc} + b_f^{enc}\right) \\
i^{enc}(t) &= \sigma\left(W_i^{enc} x_t + U_i^{enc} h_{t-1}^{enc} + b_i^{enc}\right) \\
o^{enc}(t) &= \sigma\left(W_o^{enc} x_t + U_o^{enc} h_{t-1}^{enc} + b_o^{enc}\right) \\
\widehat{c}^{\,enc\_c}(t) &= W_c^{enc} x_t + U_c^{enc} h^{enc}(t-1) + b_c^{enc} \\
\widehat{c}^{\,enc}(t) &= \tan h\left(\widehat{c}^{\,enc\_c}(t)^{enc}(t)\right) \\
\widehat{c}^{\,enc\_f}(t) &= f^{enc}(t) \odot c^{enc}(t-1))
\end{aligned}
\tag{7}
$$

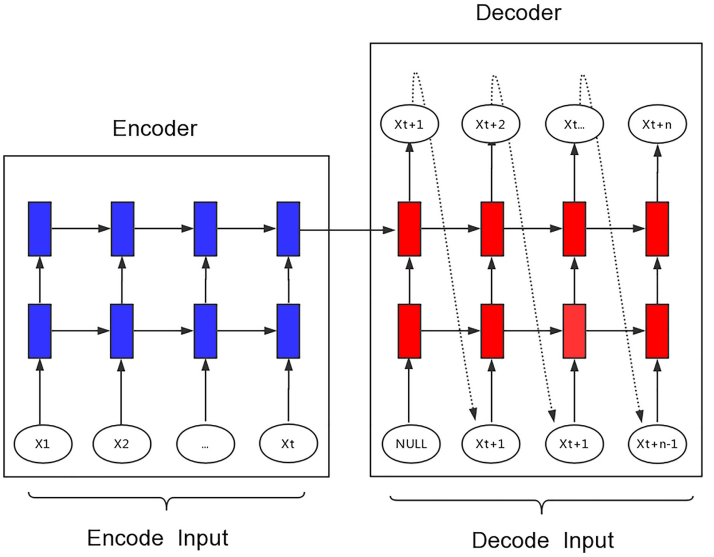

Figure 4.
The architecture of the Seq2Seq LSTM architecture, including encoder and decoder modules

$$c^{enc}(t) = \widehat{c}^{\,enc\_f}(t) + i^{enc}(t) \odot \widehat{c}^{\,enc}(t)$$
$$h^{enc}(t) = o_t \odot \tan h(c^{enc}(t))$$

We use h_t^{enc} as the intermediate state value. In the model, sigmoid and tanh are activation functions.

3.4.2 Decoder. The decoder is a feed-forward neural language network model that generates the next data series based on previous generated data and encoder states. The decoder layer is trained to generate the next data series (intermediate predicted result) for the FC layer, given previous state of the encoder. Importantly, the decoder uses the hidden state vector from the encoder as initial state, where the decoder gets initial information from. Effectively, the decoder learns to generate targets h_n^{dec}, given the input series $(\phi, h_1^{dec}, \cdots, h_{t-1}^{dec})$, cell state is c_t^{dec}, specially $c_0^{dec} = h_t^{enc}$, conditioned on the input series, $\widehat{c}^{\,dec}(t)$ indicates the updated unit status value for decoder, where t is the window size of the output series. And, h_1^{dec}, h_2^{dec},. . ., h_t^{dec} are the outputs of the Seq2Seq LSTM layer.

$$\begin{aligned}
f^{dec}(t) &= \sigma\left(W_f^{dec} x_t + U_f^{dec} h_{t-1}^{dec} + b_f^{dec}\right) \\
i^{dec}(t) &= \sigma\left(W_i^{dec} x_t + U_i^{dec} h_{t-1}^{dec} + b_i^{dec}\right) \\
o^{dec}(t) &= \sigma\left(W_o^{dec} x_t + U_o^{dec} h_{t-1}^{dec} + b_o^{dec}\right) \\
c^{dec}(0) &= h_t^{enc} \\
\widehat{c}^{\,dec}(t) &= \tan h\left(W_c^{dec} x_t + U_c^{dec} h^{dec}(t-1) + b_c^{dec}\right) \\
c^{dec}(t) &= f^{dec}(t) \odot c^{dec}(t-1) + i^{dec}(t) \odot \widehat{c}^{\,dec} \\
h^{dec}(t) &= o_t \odot \tan h\ \left(c^{dec}(t)\right)
\end{aligned} \tag{8}$$

3.5 Fully connected layer

The FC layer has two hidden layers comprised of N rectified linear units (ReLUs) presented by Nair and Hinton (2010) in 2010. Each unit in the hidden layers is fully connected to the previous layer.

$$h_i = \max(0,\ W_i h_{i-1} + b_i) \tag{9}$$

where W_1 is a weight matrix for the first hidden layer, and W_i are matrices for all subsequent layers. b_i is the bias.

The layer obtains linear forecast data series from the ARIMA filter, and non-linear forecast data series from the Seq2Seq LSTM layer. Then, it jointly generates the final forecast results from the linear and non-linear intermediate output forecasts: h_i.

3.6 Objective function and optimization strategy

To adjust the parameters and evaluate the results, we adopted the squared error as the loss function to forecast. In our model, the corresponding optimization objective is formulated as, minimize $\sum_{n=1}^{N} \|Y_n - \widehat{Y}_n\|^2$, where Y_n is actual data (target data), $\widehat{Y}_n$ is the final predict data. Our optimization strategy is similar to the traditional time-series prediction model. Thus, the objective problem of this research becomes a regression task of a group of feature-value pairs

$(\widehat{Y}_n, Y_n)$ and can be optimized by stochastic gradient decent (SGD) or its variants, such as Adam algorithm (Kingma and Ba, 2014).

4. Experiments

In this section, we lead a comprehensive set of experiments and present the experimental details and compare our results with baselines.

4.1 Dataset

We choose S&P 500 indices as our dataset, which is an American stock market index involving the market capitalizations of 500 major corporations. It has ordinary shares listed on the NYSE or NASDAQ and can represent the international stock price indices. The dataset is downloaded from Yahoo Finance database, including ranges 16 years trading price movement from 2000–01–02 to 2016–12–07 and consists 4,262 indices. It includes daily close prices, as showed in Figure 5. The study separates training data into two consecutive segments: one (70%) is for training and the other (30%) is for testing.

4.2 Evaluation

This research adopts two estimate metrics to assess performance of different methodologies for stock movement time-series prediction. They are root mean squared error (RMSE) (Plutowski *et al.*, 1996) and mean absolute error (MAE), which are two scale-dependent measures. Specifically, assuming y_t and $\widehat{y}_t$ are the target and predicted values at time t, respectively, RMSE and MAE are calculated as:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_t - \widehat{y}_t\right)^2}$$

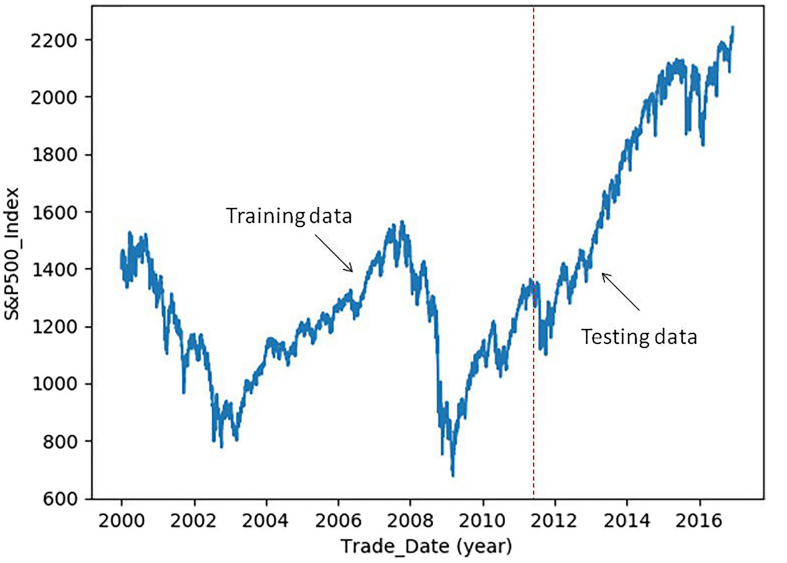

Figure 5.
Daily close price series of S&P 500 indices from 2000–01–02 to 2016–12–07

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_t - \widehat{y}_t\right| \tag{10}$$

4.3 Training and testing

The training proceeds as follows: at first, the dataset is normalized with zero mean and unit variance between 0 and 1. When training, we set batch size as 90. Hence, one sample of the training data contains 90 days' indices data. Next, we set ARIMA (p, d, q) as ARIMA (5, 1, 0), and use the ARIMA layer to extract linear component of the data as linear intermediate production. At the third step, the residual data are used for four CNN filters $filter_t \in R^{\rightarrow \mathbb{F} \times \mathbb{W}}$, to capture the one, three, seven (week), 14 days (half month). After that, we use a single-layer LSTM with 128 units for the Seq2Seq encoder, and a single-layer LSTM with 128 units for the Seq2Seq decoder to generate the non-linear intermediate prediction with four frequency data extracted from the CNN, respectively. At last, we integrate the linear intermediate production and four non-linear intermediate predictions to produce the final prediction using the FC layer, which has two hidden layers comprised of 32, 64 ReLUs. Typically, we define the learning rate as 0.001 and epoch (the time of training) as 1,000. All the parameter matrices are randomly initiated, including, W^{enc}, U^{enc}, b^{enc} and W^{dec}, U^{dec}, b^{dec}.

The test proceeds are as follows: at the initial step, the test dataset (30% dataset) is also normalized as training data did. Then, to prove the effectiveness of the RCSNet model, we compare it with three baseline models, including ARIMA (5, 1, 0), BPNN (three layers) and simple LSTM (128 units). We perform one, three, seven and 14 days ahead prediction, respectively, for all the test variables using four models mentioned above, which means the prediction horizon (length of time steps) of h is 1, 3, 7, 14 for the four models.

4.4 Results

Firstly, we compare the results of various methods with one, three, seven and 14 days (length of time steps) ahead prediction. As Table 1 shows, when the time steps h is set as 1, the ARIMA model achieves MAE at 10.939 and MAPE at 15.108, which perform better than other methods. However, with the increasement of time step h, RCSNet performs better than the average performance of baseline models. For long time step tasks: when the time steps h is set as 3, RCSNet performs 81.17% promote than the average performance of baseline models. When the time step h is set as 7, RCSNet performs with 78.72% accuracy compared with the average performance of baseline models. When the time step h is set as 14, RCSNet improves 74.2% than the average performance of the baseline models. Obviously, we can conclude that RCSNet performs better, which is shown in Figure 6. In summary, the efficiency of our model has been clearly confirmed by those experiments.

Model	Evaluation	Horizon1 = 1	Horizon = 3	Horizon = 7	Horizon = 14
ARIMA	MAE	10.939	370.47	469.86	479.417
	RMSE	15.108	424.28	533.16	533.074
BPNN	MAE	64.089	320.27	536.42	730.397
	RMSE	71.477	330.50	555.47	760.644
LSTM	MAE	139.06	41.156	136.44	297.350
	RMSE	162.59	50.997	158.31	344.391
RCSNet	MAE	39.456	53.176	80.777	129.269
	RMSE	46.471	62.466	98.474	151.624

Table 1. The results of one, three, seven, 14 days (length of time steps) ahead prediction with different methods

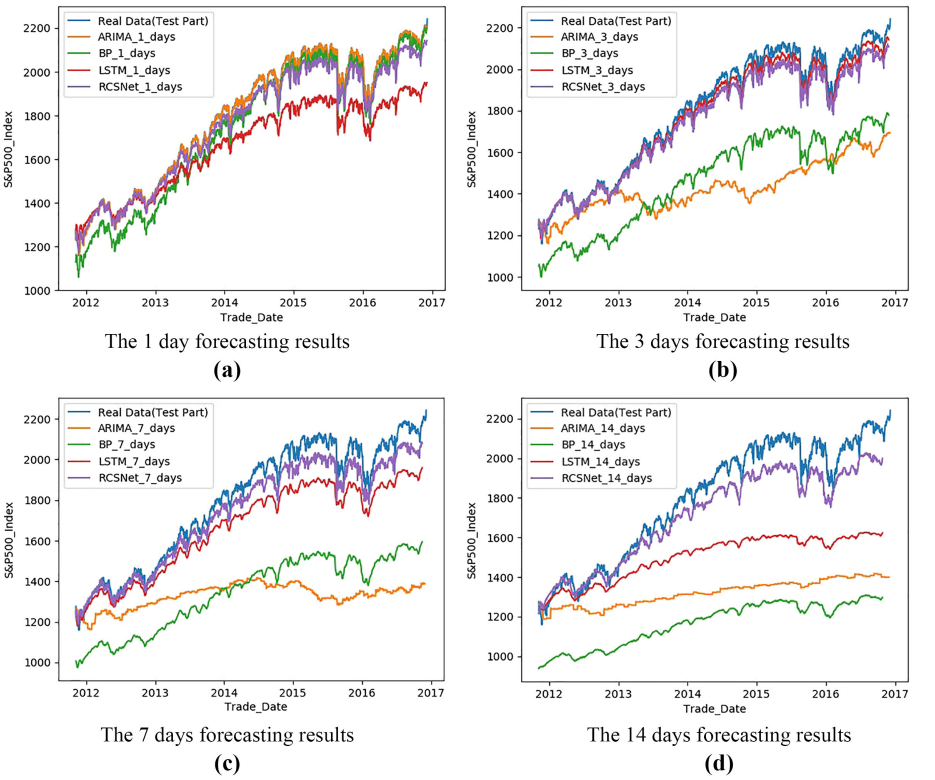

Figure 6.
The forecasting results contain 1, 3, 7, 14 days forecasting results

5. Conclusion and future work

This study presents a novel framework, RCSNet model, for stock time-series prediction by combining the traditional linear and non-linear time-series models. Our model integrates the power of existing classical models, which exploits ARIMA to capture the linear time-series component, and then applies the CNN and Seq2Seq LSTM to handle the non-linear residual component. Finally, combining the two parts of intermediate results in generating the final forecast result. The paper's experiment results demonstrate the following three points:

(1) The hybrid model is able to forecast both linear and non-linear time-series component of stock dataset.

(2) CNN and Seq2Seq LSTMs can be effectively combined for dynamic modeling of short- and long-term-dependent patterns in non-linear time-series forecast.

(3) The experimental results show our model that outperforms baseline models on S&P 500 index stock dataset from January 2000 to August 2016.

We will take into account the social media information that relates company's product and economic environment for future research and study how to filter "fake" information to make prediction not only by technical analysis but also by fundamental analysis using NLP tools. Besides, we will use time-series cross-validation rather than the simple data split for time-series experiments.

References

Bengio, Y., Simard, P. and Frasconi, P. (2002), "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, Vol. 5 No. 2, pp. 157-166, 2.

Cho, K., Merriënboer, B., Bahdanau, D. and Bengio, Y. (2014), "On the properties of neural machine translation: encoder-decoder approaches", *arXiv preprint arXiv:1409.1259*.

Chung, J., Gulcehre, C., Cho, K. and Bengio, Y. (2014), "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv preprint arXiv:1412.3555*.

Contreras, J., Espinola, R., Nogales, F.J. and Conejo, A.J. (2003), "Arima models to predict next-day electricity prices", *IEEE Transactions on Power Systems*, Vol. 18 No. 3, pp. 1014-1020, 1, 2.

Gers, F., Schmidhuber, J. and Cummins, F. (2000), "Learning to forget: continual prediction with LSTM", *Neural Computation*, Vol. 12 No. 10, pp. 2451-2471, 3.

Gers, F.A., Eck, D. and Schmidhuber, J. (2001), "Applying LSTM to time series predictable through time-window approaches", *International Conference on Artificial Neural Networks*, pp. 669-676, 4.

Gken, M., Zalc, M., Boru, A. and Dosdoru, A.T. (2016), "Integrating metaheuristics and artificial neural networks for improved stock price prediction", *Expert Systems with Applications*, Vol. 44 No. C, pp. 320-331, 2.

Hamilton, J.D. (1989), "A new approach to the economic analysis of nonstationary time series and the business cycle", *Econometrica*, Vol. 57 No. 2, pp. 357-384, 1, 2.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P. and Sainath, T.N. (2012), "Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups", *IEEE Signal Processing Magazine*, Vol. 29 No. 6, pp. 82-97, 2.

Hochreiter, S. and Schmidhuber, J. (1997), "Long short-term memory", *Neural Computation*, Vol. 9 No. 8, pp. 1735-1780, 2.

Hossain, A., Zaman, F., Nasser, M. and Mufakhkharul Islam, M. (2009), "Comparison of Garch, neural network and support vector machine in financial time series prediction", *Lecture Notes in Computer Science*, Vol. 5909, pp. 597-602, 1.

Jain, A. and Kumar, A.M. (2007), "Hybrid neural network models for hydrologic time series forecasting", *Applied Soft Computing*, Vol. 7 No. 2, pp. 585-592, 4.

Kingma, D.P. and Ba, J. (2014), "Adam: a method for stochastic optimization", *arXiv preprint arXiv: 1412.6980*.

Kuan, C.M. and Liu, T. (1995), "Forecasting exchange rates using feedforward and recurrent neural networks", *Journal of Applied Econometrics*, Vol. 10 No. 4, pp. 347-364, 3.

Ltkepohl, H. (2005), *New Introduction to Multiple Time Series Analysis*, Springer Science & Business Media.

Maier, H.R. and Dandy, G.C. (2000), "Neural networks for the prediction and forecasting of water resources variables: a review of modelling issues and applications", *Environmental Modelling Software*, Vol. 15 No. 1, pp. 101-124, 1.

Nair, V. and Hinton, G.E. (2010), "Rectified linear units improve restricted Boltzmann machines", *International Conference on International Conference on Machine Learning*, pp. 807-814, 7.

Nayak, R.K., Mishra, D. and Rath, A.K. (2015), "A Naïve SVM-KNN based stock market trend reversal analysis for Indian benchmark indices", *Applied Soft Computing*, Vol. 35, pp. 670-680.

Pai, P.F. and Lin, C.S. (2005), "A hybrid arima and support vector machines model in stock price forecasting", *Omega*, Vol. 33 No. 6, pp. 4974-505.

Plutowski, M., Cottrell, G. and White, H. (1996), "Experience with selecting exemplars from clean data", *Neural Networks*, Vol. 9 No. 2, pp. 273-294, 8.

Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1988), *Learning Internal Representations by Error Propagation*, Institute for Cognitive Science, University of California, San Diego.

Saad, E.W., Prokhorov, D.V. and Wunsch, D.C. (1998), "Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks", *IEEE Transactions on Neural Networks*, Vol. 9 No. 6, pp. 1456-70, 3.

Sims, C.A. (1980), "Macroeconomics and reality", *Econometrica*, Vol. 48 No. 1, pp. 1-48, 2.

Tsay, R.S. (2005), *Analysis of Financial Time Series*, John wiley & sons, Vol. 543.

Wang, J.Z., Wang, J.J., Zhang, Z.G. and Guo, S.P. (2011), "Forecasting stock indices with back propagation neural network", *Expert Systems with Applications*, Vol. 38 No. 11, pp. 14346-14355, 2.

Further reading

Ardalani-Farsa, M. and Zolfaghari, S. (2010), "Chaotic time series prediction with residual analysis method using hybrid elmannarx neural networks", *Neurocomputing*, Vol. 73 No. 1315, pp. 2540-2553.

Greff, K., Srivastava, R.K., Koutnik, J., Steunebrink, B.R. and Schmidhuber, J. (2017), "LSTM: a search space odyssey", *IEEE Transactions on Neural Networks Learning Systems*, Vol. 28 No. 10, pp. 2222-2232.

Kaastra, I. and Boyd, M. (1996), "Designing a neural network for forecasting financial and economic time series", *Neurocomputing*, Vol. 10 No. 3, pp. 215-236.

Lipton, Z.C., Berkowitz, J. and Elkan, C. (2015), "A critical review of recurrent neural networks for sequence learning", *Computer Science*.

Makridakis, S. and Hibon, M. (1997), "ARMA models and the Box -Jenkins methodology", *Journal of Forecasting*, Vol. 16 No. 3, pp. 147-163.

Rather, A.M., Agarwal, A. and Sastry, V.N. (2015), "Recurrent neural network and a hybrid model for prediction of stock returns", *Expert Systems with Applications*, Vol. 42 No. 6, pp. 3234-3241.

Zhang, G.P. (2003), "Time series forecasting using a hybrid arima and neural network model", *Neurocomputing*, Vol. 50 No. 1, pp. 159-175.

Corresponding author

Zhonglu Chen can be contacted at: cxsichuan@yeah.net