Science & Technology Development Journal – Economics - Law and Management, 4(1):500-515

Open Access Full Text Article

Research Article

Time series prediction: A combination of Long Short-Term Memory and structural time series models

Quoc Luu[1,*], Son Nguyen[2], Uyen Pham[1,2]

ABSTRACT

Stock market is an important capital mobilization channel for economy. However, the market has potential loss due to fluctuations of stock prices to reflect uncertain events such as political news, supply and demand of daily trading volume. There are many approaches to reduce risk such as portfolio construction and optimization, hedging strategies. Hence, it is critical to leverage time series prediction techniques to achieve higher performance in stock market. Recently, Vietnam stock markets have gained more and more attention as their performance and capitalization improvement. In this work, we use market data from Vietnam's two stock market to develop an incorporated model that combines Sequence to Sequence with Long-Short Term Memory model of deep learning and structural models time series. We choose 21 most traded stocks with over 500 trading days from VN-Index of Ho Chi Minh Stock Exchange and HNX-Index of Hanoi Stock Exchange (Vietnam) to perform the proposed model and compare their performance with pure structural models and Sequence to Sequence. For back testing, we use our model to decide long or short position to trade VN30F1M (VN30 Index Futures contract settle within one month) that are traded on HNX exchange. Results suggest that the Sequence to Sequence with LSTM model of deep learning and structural models time series achieve higher performance with lower prediction errors in terms of mean absolute error than existing models for stock price prediction and positive profit for derivative trading. This work significantly contribute to literature of time series prediction as our approach can relax heavy assumptions of existing methodologies such as Auto-regressive–moving-average model, Generalized Auto-regressive Conditional Heteroskedasticity. In practical, investors from Vietnam stock market can use the proposed model to develop trading strategies.

Key words: LSTM, Seq2Seq, Structural models, Hybrid model

[1]Quantitative and Computational Finance, John von Neumann Institute, Ho Chi Minh City, Vietnam

[2]Economic Mathematics, University of Economics and Law, VNU-HCM, Ho Chi Minh City, Vietnam

Correspondence

Quoc Luu, Quantitative and Computational Finance, John von Neumann Institute, Ho Chi Minh City, Vietnam

Email: quoc.luu2015@qcf.jvn.edu.vn

History

- Received: 28/6/2019
- Accepted: 25/9/2019
- Published: 31/3/2020

DOI : 10.32508/stdjelm.v4i1.593

INTRODUCTION

Describing the behavior of the observed time series plays a critical role to understand the past and predict the future in many disciplines. In quantitative finance, time series prediction is a very important task for risk management to measure of the uncertainty of investment return[1], portfolio construction for hedge fund[2], high-frequency trading[3].

However, creating high accuracy prediction of time series with low error is not an easy job, due to high fluctuations of stock market. From this perspective, there have been many methods that were proposed to study historical patterns of time series data to crate high quality of stock price prediction. To measure quality of a prediction, forecast error indicators were compute from actual price and predicted price. Mean Square Error (MSE) or Root-Mean-Square Error are populated indicators for the measurement works.

In terms of time series prediction, there are many approaches to address the matter. However, there are two methods which have been widely adopted. The first one is univariate analysis to capture volatility. They include autoregressive models (AR(p)), moving-average models (MA(q)), combination of autoregressive and moving-average models (ARMA(p, q)) for linear processes, and generalized autoregressive conditional heteroscedastic (GARCH(p, q)) for nonlinear processes to model return of stocks[4]. By differencing, a transformation from price to return of a stock poses a problem. Unobserved components (e.g. seasonal component, trend) of raw series were eliminated. Furthermore, differencing is hard to interpret and select adequate model. Hence, the second approach have been proposed to fill the gap[5]. It is called Structural Time Series Models which comprises trend component, seasonal component, and a random irregular component to model a time series without differentiation.

With revolution of computational power, beside statistical models, machine learning, and deep learning models have been widely adopted to solve many problems from academic to industry. In financial time series prediction, we can leverage these models to achieve higher accuracy. In deep learning, models are considered black-box with billions of parameters. However, feature engineering is still an important work to improve model accuracy[6]. Furthermore, neural network like Long-Short Term Memory of deep learning can link current event to previous events while Structural Time Series Models only depends on previous event. Hence, it is critical to combine these two approaches to address the limitations. In this work, we step-by-step describe procedures to perform fitting data with Structural Time Series Models, Sequence to Sequence model, and the combination of these two models. We report the results of the fitting process to 21 stocks listed on Ho Chi Minh Stock Exchange, then we compare the prediction results. Furthermore, we use proposed model to automatically trade VN30F1M futures contract on Ha Noi Stock Exchange for back testing.

LITERATURE REVIEW

Trend Models of Structural Time Series Models

Decomposition of time series is an important procedure. Traditionally, regarded as a functional depended on time and deterministic, non-stationary time series are often detrended by applying difference to construct models from the processed data. It is suggested that this procedure may lead to misleading results if trend is not deterministic[7]. A structural time series models are a decomposable time series in terms of three components of trend, seasonality and cycle[8,9]. It is defined as following equation:

$$y(t) = g(t) + s(t) + h(t) + \varepsilon_t \qquad (1)$$

where $t = 1,\ldots,T,$, and $g(t)$ is stochastic and non-periodic changes trend, $s(t)$ is a seasonal stationary linear process with periodic changes (e.g. quarterly, yearly seasonality), and $h(t)$ is a cyclical frequency of time occurring on potentially irregular schedules over one or more days[10].

Many researches strongly support the model in practice have been carried out. For instance, Harvey shown that class of structural models have several advantages over the seasonal ARIMA models adopted and are applicable to model cycles in macroeconomic time series[5,11]. Kitagawa, Gersch decomposed time series into trend, seasonal, globally stationary autoregressive and observation error components with state space Kalman filter and used Akaike minimum AIC procedure to select the best of the alternative state space models[12] Taylor, Letham use structural models for forecasting of business time series[10].

The local linear trend is a process can be regarded as a local approximation to a linear trend. The stochastic linear process can be described as:

$$\begin{aligned} y(t) &= g(t) + \varepsilon_t \\ g(t) &= g(t-1) + \beta(t-1) + \eta_t \qquad (2) \\ \beta(t) &= \beta(t-1) + \zeta_{t'} \end{aligned}$$

where the $\varepsilon_t \sim NID(0,\sigma_\varepsilon^2)$, $t = 1,\ldots,T$, $\eta_t \sim NID(0,\sigma_\eta^2)$, *and* $\zeta_t \sim NID(0,\sigma_\zeta^2)$ are distributed independent of one another and white noise disturbance terms with mean zero and variances σ_ε^2, σ_η^2 *and* σ_ζ^2 respectively[13]. Koopman and Ooms[14] proposed trend with stationary drift process to extend local linear trend process by adding a stationary stochastic drift component:

$$\begin{aligned} g(t) &= g(t-1) + \beta(t-1) + \eta_t \qquad (3) \\ \beta_t &= (1-\varphi_\beta)\,\bar{\beta} + \varphi_\beta\,\beta_t + \zeta_t \end{aligned}$$

with autoregressive coefficient $0 < \varphi_\beta \leq 1$. However, there is a drawback with this approach that make such drift processes are difficult to identified. It requires very large data samples.

Taylor and Letham[10] developed new type of trend models. Accordingly, they suggested that there are two types of trend models: a saturating growth model, and a piecewise linear model (see **Figure 1**). Saturating growth model is characterized by growth rate and limitation of population growth. By applying nonlinear logistic function:

$$g(t) = \frac{C}{1+e^{-k(t-m)}} \qquad (4)$$

with e is the natural logarithm base, m is the value of sigmoid middle point, C is the maximum capacity value, k is growth rate of the curve. From that point of view, it cannot be captured movement in dynamic world due to nonconstant growth of maximum capacity value and rate of the curve. Hence, to overcome the issues, Taylor and Letham defined a time-varying of maximum capacity C and growth rate k. Suppose that we explicitly define S changepoints at times s_j, $j = 1,\ldots,S,$, and a vector of rate adjustments $\delta \in R^S$ with δ_j is the change in rate that occurs at time s_j[10]. The saturating growth model is defined as:

$$g(t) = \frac{C(t)}{1+e^{-(k+a(t)^\intercal\,\delta)(t-(m+a(t)^\intercal\,\gamma)}} \qquad (5)$$

where

$$\gamma_j = (s_j - m - \textstyle\sum_{i<j}\gamma_l\,)(1 - (\frac{k+\sum\limits_{l<j}\delta_l}{\sum_{l\leq j}\delta_l}\,)$$

$$a_j\,(t) = \{ {}^{1,if\ t\geq s}_{0,otherwise}$$

Maximum capacity $C(t)$ is adopted from external data source.

From saturating growth model, we can define piecewise linear model without exhibit saturating growth:

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma) \qquad (6)$$

like saturating growth model, k is the growth rate, δ has the rate adjustments, m is offset parameter, and γ_j is set to $-s_j\delta_j$ to make the function continuous.

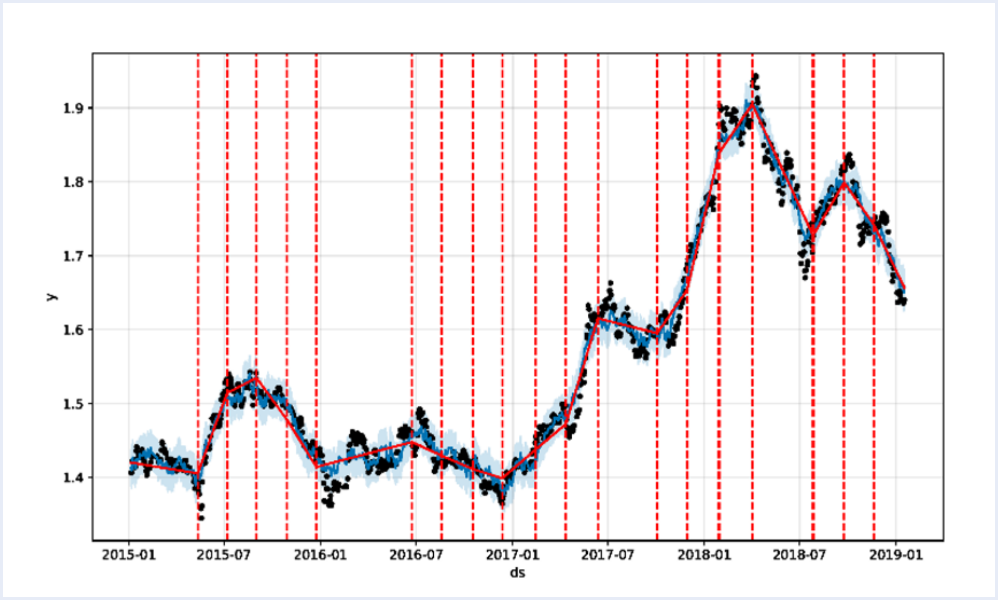


Figure 1: **Fitted daily stock price of Ho Chi Minh City Securities (HCM) stock with piecewise linear model from January 3rd, 2017 to February 26th, 2019 in log-scale.**

Deep Neural Network

Recurrent Neural Network

Despite powerfulness of deep neural networks, traditional neural networks have two drawbacks[15] . Firstly, main assumption of standard neural networks is independence among the samples (data points). On the other words, traditional neural networks cannot link current event to previous events to inform later ones due to it stateless preservation. In time series analysis, it is widely accepted that current value depends on past values[4]. It is unacceptable because the independence assumption fails. Secondly, traditional neural networks require fixed-length vector of each sample. Hence, it is critical to develop a new generation of deep neural networks. Rumelhart, Hinton, Williams *(p.533)* introduced a new learning procedure for neuron networks with backpropagation which can capture internal hidden state to "represent important features of the task domain"[16]. Furthermore, with current development, recurrent neural network can model sequential data with varying length and time dependences. A simple feed forward recurrent neural network is defined[17]:

$$h^{(t)} = \sigma(W^{hx}x^{(t)} + W^{hh}h^{t-1} + b_h) \quad (7)$$

$$\widehat{y}^{(t)} = softmax(W^{yh}h^{t} + b_y) \quad (8)$$

where $h^{(t)}$ is hidden state of input data point at time t. Clearly, $h^{(t)}$ is influenced by $h^{(t-1)}$ in the networks previous state. The output $\widehat{y}^{(t)}$ at each time t is calculated given the hidden node values $h^{(t)}$ at time t. W^{yh} is weight matrix of input-hidden layer and W^{hh} is the matrix of hidden-to-hidden transition. In most context, $h^{(0)}$ is initialized to zero. Haykin, Principe, Sejnowski, Mcwhirter suggested that RNN can achieve stability and higher performance by nonzero initialization[18]. By comparison to traditional fully connected feedforward network, a recurrent neural network takes advantage of sharing parameters across the model that helps it learns without separately at each position of sentence or series[19]. Earlier, Jordan proposed an almost like[17]. However, context nodes are fed from the output layer instead of from hidden layers[20]. It means that Jordans neural network can take previous predicted output into account to predict current output.

$$h^{(t)} = \sigma(W^{hx}x^{(t)} + W^{hh}\,\widehat{y}^{(t-1)} + b_h) \quad (9)$$

$$\widehat{y}^{(t)} = softmax(W^{yh}h^{t} + b^{y})$$

In term of training, there are two steps to train a recurrent neural network. First, the forward propagation creates $\widehat{y}$ outputs. After that, loss function value $L(\widehat{y}^{k}, y_k)$ of the network of each output node k are compute in backpropagation stage. There are many types of loss function to measure distance between the output and the actual value of classification problems. To minimize the distance, we need to update each of the weights iteratively by applying backpropagation algorithm[16].

The algorithm applies derivative chain rule to calculate the derivative of the loss function L for each parameter in the network. In addition, weights of neural network are updated by gradient descent algorithm[15]. Hence, gradient of error of a neuron is calculated as:

$$\delta_k = \frac{\partial L(\widehat{y}^k, y_k)}{(\partial \widehat{y}_k)} g'_k(a_k) \qquad (10)$$

where $a_k = w\widetilde{a}_k + b$ is input to node k and $\widetilde{a}_k$ is incoming activation output of $a_k, g'_k(a_k)$, is activation function for node k. The first term $\frac{\partial L(\widehat{y}_k, y_k)}{(\partial \widehat{y}_k)}$ expresses how fast the cost is changing as a function of estimated output. The second term $g'_k(a_k)$ suggests rate of change of g_k activation function at a_k . In vectorized form, we generalize equation (10) for any layerl^{th} :

$$\delta^l = \nabla_{\widehat{y}\, C \odot g'(a^l)} \qquad (11)$$

In addition, from the δ^l , we can compute the error of the next layer δ^{l+1} as:

$$\delta^l = ((w^{l+1})^T \ \delta^{t+1}) \odot g'(a^l) \qquad (12)$$

and error with respect to any weight, bias in the neural network:

$$\frac{\partial C}{\partial w^l} = \widehat{y}^{l-1} \delta^l \qquad (13)$$
$$\frac{\partial C}{\partial b^l} = \delta^l$$

From the final layer to first hidden layer, for each layer of the neural network, we can apply the back-propagation and compute the error vector δ^l with the chain rule repeatedly to update weight and bias vectors. In term of local minimum optimization, gradient descent is utilized for finding the minimum of cost function by updating weight and bias vectors. It is computed as:

$$w^l \rightarrow w^l - \frac{\eta}{m} \sum x \ \delta^{x,l} \ (\widehat{y}^{x,l-1}) \qquad (14)$$
$$b^l \rightarrow b^l - \frac{\eta}{m} \sum \delta^{x,l}$$

where m is number of training examples in a given mini-batch with each training example x, η is a step size. In practical, there are many optimizers developed to improve mini-batch gradient descent limitations[21]. For instance, Qian[22] and Yu[23] accelerated gradient was developed to relax navigating ravines problem of stochastic gradient decent. Recurrent neural network is a breakthrough in temporal sequence by adding internal state (memory) in each cell to process sequences of inputs. In term of training, recurrent neural network parameters can be computed and optimized by feed forward propagation and back-propagation. For shallow network with a few hidden layers, the algorithm can be trained effectively. However, with many hidden layers, it is hard to train the network due to vanishing and exploding gradient problem as derivatives become too small (e.g. 0 to 1 for sigmoid activation function) or too large. It only allows the network to learn in short-range dependencies and prevents from learning long-range dependencies. As a result, long-short term memory network architecture[24], rectified linear units activation function[25], residual learning framework He, Zhang, Ren, Sun were introduced to overcome the limitation[26].

Long-Short Term Memory Network

Formally identified by Hochreiter in both theoretical and experimental approaches[27], with involvement of long-term dependencies data, back propagation algorithm of recurrent neural network is showed that it suffers from insufficient that tends to explode or vanish through time may lead to oscillating weights or unusable model. Not just recurrent neural network, Bengio, Simard, Frasconi[28] also pointed out that any deep feed-forward neural network with shared weights may have vanishing gradient problem. Hochreiter, Schmidhuber (p.6) developed a new approach called Long Short-Term Memory (LSTM) to fill these gaps by introducing "input gate unit", "output gate unit", and "memory cell"[24]. Accordingly, the purpose of multiplicative input gate unit is to protect memory contents from irrelevant inputs, and multiplicative output gate unit is to protect other units from perturbation by currently irrelevant stored memory contents. On the other words, with the new LSTM architecture (see **Figure 2**), each cell can maintain its state over time, and adjust input or output information. Hence, the new type of neural network architecture is able to capture very long-term temporal dependencies effectively, handle noise and continuous values with unlimited state numbers in principle.

Since introduction, with revolution of computational power, LSTM has been widely adopted and applied for many difficult problems of many fields in practice and academic. This includes language modeling[28], text classification[30], language translation[30], speech recognition[31]. From original LSTM proposed by Hochreiter, Schmidhuber[24], a significant improvement had been developed by introducing forget gates to reset out-of-dated contents of LSTM memory cells[32]. In addition, to achieve higher capability of learning timings, peephole connections that allows gates to look at cell state were added to LSTM neural network. A forward pass LSTM architecture with forget gate and peephole connections is described as[33]:

$$\bar{z}^t = W_\zeta \, x^t + R_\zeta \, y^{t-1} + b_\zeta \qquad (15)$$
$$\bar{z}^t = W_\zeta \, x^t + R_\zeta \, y^{t-1} + b_\zeta$$
$$z^t = g(\bar{z}^t)$$
$$i^{-t} = W_i x^t + R_i y^{t-1} + p_i \odot c^{t-1} + b_i$$
$$i^t = \sigma(i^{-t})$$
$$\bar{f}^t = W_f \, x^t + R_f \, y^{t-1} + p_f \odot c^{t-1} + b_f$$
$$f^t = \sigma(\bar{f}^t)$$
$$c^t = z^t \odot i^{-t} + c^{t-1} \odot f^t$$
$$o^{-t} = W_o x^t + R_o y^{t-1} + p_o \odot c^{t-1} + b_o$$
$$o^t = \sigma(o^{-t})$$
$$y^t = h(c^t) \odot o^t$$

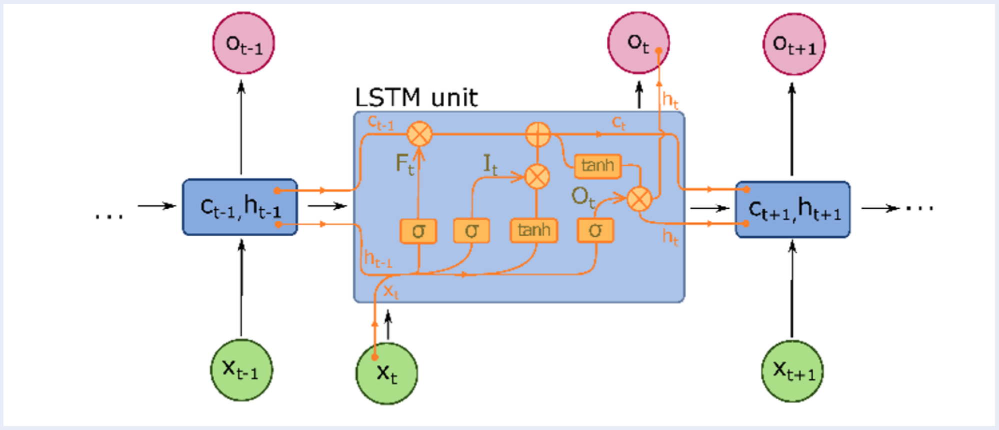


Figure 2: **Long-Short term memory network architecture. Adopted from Harmon, Klabjan[29].**

where z^t is block input, i^t is input gate, f^t is forget gate, c^t is memory cell, o^t is output gate, y^t is block output. $W_z,\ W_i,\ W_f,\ W_o\ \in R^{N\times M}$ are input weights; $R_z,\ R_i,\ R_f,\ R_o\ \in R^{N\times M}$ are recurrent weights; $p_i,\ p_f,\ p_o \in R^N$ are peephole weights; $b_z,\ b_i,\ b_f,\ b_o$ are bias weights; $g,\ \sigma,\ h$ are activation functions.

Like RNN, LSTM is trained with gradient descent as it is a differentiable function estimator[34]. Backpropagation equations of LSTM are detailed:

$\delta y^t = \Delta^t + R_\zeta \tau \delta z^{t+1} + R_i^T \delta i^{t+1} + R_f^T \delta f^{t+1} + R_o^T \delta o^{t+1}$ (16)

$\delta o^{-t} = \delta y^t \odot h(c^t) \odot \sigma' o^{-t})$

$\delta c^t = \delta y^t \odot o^t \odot h'\ (c^t) + p_o \odot \delta \overline{o}^{\,t} + p^i \odot \delta \overline{i}^{\,t+1} + p^f \odot \delta \overline{f}^{\,t+1} + \delta c^{t+1} + f^{t+1}$

$\delta \overline{f}^{\,t} = \delta c^t \odot c^{t-1} \odot \sigma'(\overline{f}^{\,t})$

$\delta \overline{i}^{\,t} = \delta c^t \odot z^t \odot \sigma'^{(\overline{i}^{\,t})}$

$\delta \overline{z}^{\,t} = \delta c^t \odot i^t \odot g'^{(\overline{z}^{\,t})}$

$\delta x^t = W_z^T t \delta \tilde{z}^t + W_i^T \delta i\ t + W_f^T\ \delta f\ t + W_o^T\ \delta o\ t$

$\delta W_* = \sum_{t=0}^{T} \langle \delta *^t, X^t \rangle$

$\delta R_* = \sum_{t=0}^{T-1} \langle \delta *^{t+1}, X^t \rangle$

$\delta b_* = \sum_{t=0}^{T} \langle \delta *^t \rangle$

$\delta p_i = \sum_{t=0}^{T-1} c^t \odot \delta \overline{i}^{\,t+1}$

$\delta p_f = \sum_{t=0}^{T-1} c^t \odot \delta \overline{f}^{\,t+1}$

$\delta p_o = \sum_{t=0}^{T-1} c^t \odot \delta \overline{o}^{\,t+1}$

Where * can be one of $\overline{z},\ \overline{i},\ \overline{f},\ \overline{o}$ and $\langle *1, *2 \rangle$ is outer product of two vectors.

It is worth to note that peephole is not always implemented as forget gate because it simplifies LSTM and reduce computational cost without significantly scarifying performance. For instance, Keras[35] does not support peephole, but CNTK, TensorFlow does support[35,36]. There have been many variant versions of vanilla LSTM architecture with minor changes. Greff *et al.* found that vanilla LSTM (with forget gate and peephole) achieve reasonably performance on various datesets[33]. Despite effectiveness of LSTM, there are many efforts to simplify the architecture as LSTM requires huge computational power of hardware. Gated Recurrent Unit (GRU), a variant of LSTM with fewer parameters than LSTM by simplifying forget gate, which introduced by Cho *et al.*[37] has reasonable accuracy. However, Britz *et al.* shows that LSTM still significantly outperforms GRU[38]. Hence, Van der Westhuizen *et al.* is another attempt to save computational costs and maintain performance of models by developing a forget-gate-only version of the LSTM with chrono-initialized biases that achieves lightly higher accuracy[39].

Sequence to sequence model

Sequence to Sequence is a learning model that maps an input sequence from a fixed-sized vector using a LSTM to another LSTM to extract an output sequence. Sequence to Sequence has been widely applied in machine translation[40], video captioning[41], time series classification for human activity recognition[42]. Bahdanau *et al.* used RNN Encoder-Decoder that contains two recurrent neural networks (or long short-term memory) to represent an input sequence into another sequence of symbols[43]. One the other words, encoder-decoder architecture is used to encode a sequence, decode the encoded sequence, and recreate the sequence. The approach aims to maximize the conditional probability of output sequence given an input sequence.

Encoder neural network transforms an input sequence of variable length $X = x_1, x_2, \ldots, x_T$ into a

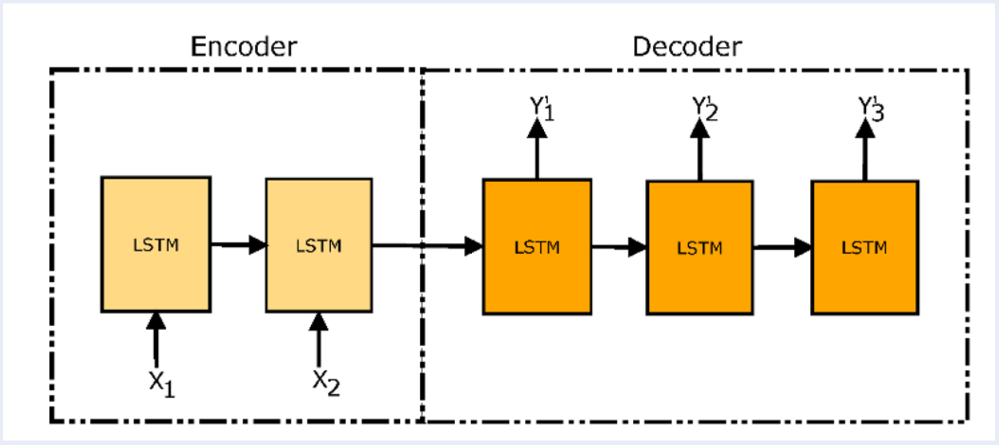


Figure 3: **Encoder-Decoder architecture.**

fixed-length context variable with information of the sequence (see **Figure 3**). RNN is mostly used as an encoder neural network. However, Sutskever *et al.*[40] found that LSTM significantly outperformed shallow LSTMs and RNN. As mentioned, RNN and LSTM use previous hidden states $h_1, h_2, \ldots, h_{t-2}, h_{t-1}$ to create current hidden stateh_t. Hence, hidden state of an input sequence is defined as:

$$h_t = f(x_t,\ h_{t-1}) \qquad (17)$$

$$c = k(h_1, h_2, \ldots, h_T)$$

where is hidden state at time t, c is summary hidden state of the whole input sequence, function f() can be RNN, LSTM, GRU network, or an activation function. With summary hidden state c, given a target output $Y = y_1,\ y_2, \ldots,\ y_{T'}$, instead of computing $P(Y|X)$ directly, decoder neural network computes conditional probability of using previous information and summary hidden state. It is formally described as:

$$P(y_1, \ldots,\ y_{T'}|x_1, \ldots,\ y_T)$$

$$= \prod_{t'=1}^{T'} P(y_t, |c, y_1, \ldots, y_{t'-1}) \qquad (18)$$

The trained sequence to sequence model can be used to generate a sequence give an input sequence. In machine translation, reverse the order of the words of the input sequence is necessary because it is easy for optimizer (e.g. stochastic gradient decent) to "establish communication between the input and the output"[40] (p.3). For the sake of nature, time series prediction problems always have desired order as input and output is straightforward sequence.

EMPIRICAL RESULTS

Data

In this study, for liquidity and fairness of trading, we use daily price data of 21 most traded stocks that is listed on from VN-Index of Ho Chi Minh Stock Exchange and HNX-Index of Hanoi Stock Exchange (Vietnam) from 05 January 2015 to 19 January 2019. It is 1010 data points in total. We use first 965 data points for training and the last 45 data points for testing. It is 9-type of window size for out-of-sample prediction. It varies from 5 to 45 with 5-step ahead. Furthermore, we use daily price of VN30F1M contract that are traded on Ha Noi Stock Exchange from 1 September 2017 to from 13 November 2018 for training, and from 14 November 2018 to 15 May 2019 for performance validation (120 trading days).

Data Pre-processing

Beyond algorithm improvement and parameter tuning, an approach to improve the accuracy of machine learning model is apply data pre-processing techniques. For instances, these techniques are imputed missing values, encode categorical data, detect outliers, transform data, and scaling data. In this work, we perform logarithm and Box-Cox transform to transform the input dataset. Rationally, the idea behind the logarithm transformation is to turn probabilistic distribution of raw input data from skewed data into approximately normal. Hence, prediction performance is improved dramatically[44]. However, in some circumstances, the logarithm technique does not generate new data with less variable or more normal. In contrast, it may lead to be more variable and more skewed[45]. Thus, it is recommended that transformation techniques must be applied very cautiously. Output data of the transform stage is passed to data scaler to be normalized. There are many types of scaling method (e.g. maximum absolute value, given range of feature). We use min-max scaler by scaling the input feature to a range of [0,1]. It ensures the large input value do not overwhelm smaller value inputs, then helps to stabilize neural networks[46].

Detail Results

Structural Time Series

The aim of this step is to create baseline models for evaluating prediction quality of structural time series and sequence to sequence models with our proposed model. Mean square error was calculated to measure performance of each out-of-sample forecast.

We develop structural time series models as a baseline model. For this task, we choose Prophet package which is developed by Facebook for Python programming language[10]. In this model, data input is a transformed price of stocks in logarithm. In terms of parameter tuning, we almost use default settings except adding monthly, quarterly, and yearly with Fourier orders. We initialize 20, 30, 30 for Fourier orders of monthly, quarterly, and yearly respectively. As it is required future data would have to be known to perform prediction if we use Box-Cox transformation as an extra regressor in structural time series models, we omit the transformation procedure[47]. Without extra regressor, the model can generate prediction of 21 selected tickers from 5 to 45 with 5-step incrementation window size.

As mentioned, Prophet model is structural time series models that combines trend g(t), seasonality s(t), and irregular events. **Figure 4** describes our attempt to generate out-of-sample prediction for model quality evaluation and trend g(t) of series as a feature input of Sequence to Sequence using Prophet model from transformed logarithmic form and Box-Cox form of stock price series.

In detail, for every stock v in selected list of stocks, we transform the price to log-scale LP and Box-Cox series BC to use as an input for Prophet model P. We set no out-of-sample prediction (W=0) to extract trend series T from in-sample data generated by P as a feature of Sequence to Sequence model. For performance comparation, we set w to every 9-type W of window size for out-of-sample prediction.

Sequence to sequence Model

Regarding to baseline models, we also develop a Sequence to Sequence with LSTM architecture. We use Keras with Tensorflow backend to create Encoder-Decoder model to solve the sequence to sequence problem[35,36]. To benefit from the efficiency of parallel computation for training deep learning neural network, we train the model on virtual machine with GPU on Google Cloud Platform.

Sequence to sequence model use states of encoder neural network to generate prediction from decoder neural networks. Hence, we feed normalized stock price series to the model and generate prediction. In **Figure 5**, we describe approach that we use to develop baseline prediction with sequence to sequence model. Like vanilla LSTM model for supervised learning, we train input data with many iterations. However, we discard output of encoder and use state and as input for decoder. Furthermore, to create prediction for the proposed model, we add trend series (extracted from **Figure 4**) as another input feature.

The implementation is straightforward. First, like **Figure 4**, we use scaled data of Box-Cox BC and logarithm transforms LP as input data. However, we scale every BC and LP to range from 0 to 1 to create x for every scaled list of stocks price X^*. Furthermore, we use logarithm transformed series as target data. We create and extract hidden states of encoder model En with LSTM architecture and initialize decoder model DE with these hidden states. A main advantage of Sequence to Sequence with LSTM over structural time series models is that it can dynamically perform prediction multiple time steps without requiring extra data. In terms of accuracy, we found that result of deeper LSTM model does not outperform shallow one. Hence, we used LSTM with single hidden layer, with 64 cells and rectified linear unit activation function. To prevent over-fitting, we apply both L2 regularization and dropout. We use 0.0001 for regularization parameter lambda, rate of dropout is 0.001 as recommended[48].

Sequence to Sequence with Structural Time Series Models

In this step, we combine both sequence to sequence model and structural time series models. Specifically, we use output dataset D (with W = 0) from **Figure 5** as train data for **Figure 6**. On the other words, we combine trend component of structural time series models with price of stock in Box-Cox and logarithm forms. Parameters of these models are defined exactly same as aforementioned baseline models. We found that results are improved dramatically.

Results Analysis

Structural time series models was used to generate a set of out-of-sample forecast in multiple window time steps in log-scale (see **Table 1**). In terms of prediction error, the result show that MSE = 0.087787 (CTG at 45 time steps ahead) is highest, MSE = 0.003501 (SSI at 5 time steps ahead) is lowest. Likewise, results from Sequence to Sequence model (see **Table 2**) and Sequence

```
Input  : List of price of $V$ stocks in logarithmic form
         $LP$, Box-Cox form $BC$, steps ahead $W$
Output: Trend prediction $T$, log-price prediction $\hat{Y}$,
         combination result dataset $D$ of stocks
for $v$ in $V$ do
    if Length of $LP > 500$ then
        Initialize Prophet model $P$;
        if $W = 0$ then
            $T \leftarrow P(LP, BC)$;
            $D \leftarrow [LP, BC, T]$;
            return $D$;
        else
            for $w$ in $W$ do
                // out-sample prediction w > 0;
                $\hat{Y} \leftarrow P(LP, BC)$;
                return $\hat{Y}$;
            end
        end
    end
end
```

Figure 4: **Algorithm structural time series analysis with Facebook Prophet library.**

```
Input  : Arrays of train data $X^*$, steps ahead $W$
Output: Log-price prediction $O$
for $x$ in $X^*$ do
    Pre-train encoder prediction network $En$ and
      decoder prediction network $De$ with $x$;
    Last element $k$ of array $x$;
    States $H, C \leftarrow En(k)$;
    for $w$ in $W$ do
        Output $O \leftarrow De([H, c])$
    end
end
```

Figure 5: **Algorithm of Sequence to Sequence**.

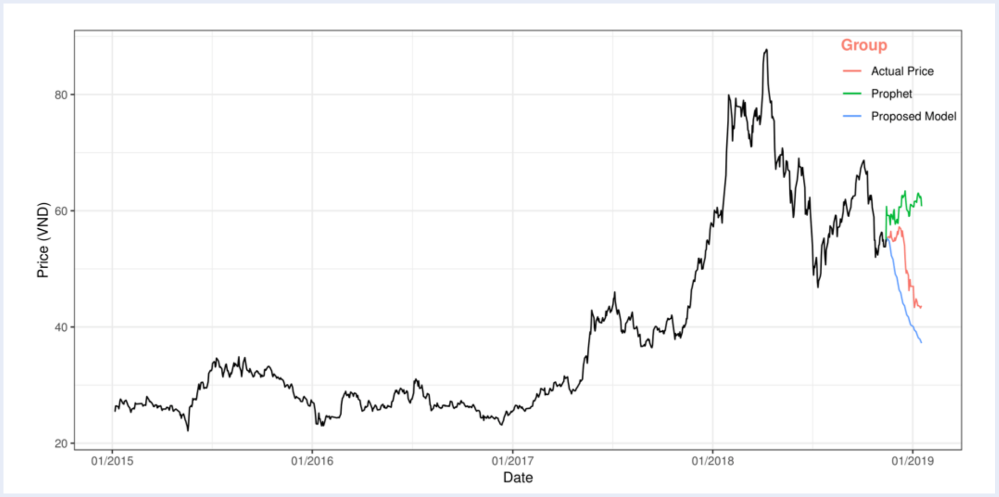


Figure 6: 45-day forecast of HCM.

to Sequence with Structural Time Series Models (see **Table 3**) show that MSE = 0.231800 (PNJ at 45 time steps ahead) and MSE = 0.046146 (ACB at 45 time steps ahead) are highest, MSE = 0.000068 (CII at 5 time steps ahead) and MSE = 0.000006 (CII at 5 time steps ahead) are lowest. **Figure 6** plots prediction output of models with actual data of HCM stock.

In term of back testing, by applying **Figure 6**, we found that the proposed model can create positive profit (see **Figure 7**). For simplicity, we do not consider tax and transaction fee. From initial invested money $1000, we get $1159.2 at the end of the test. Specifically, we develop trading environment from real market data with return *TRR* (index point) to measure reward of the test. Agent is developed from proposed model. For every day of 120 trading days *TD*, it uses predicted return *PR* to choose positions. If predicted return *PR* on the next two days (*W=2)* is positive, we choose *Long* position. If it is negative, we choose *Short* position. If is around zero, we hold position. Position is closed when profit *PFT* is bigger than a point or the position is held more than a day (*T=2*).

From univariate time series analysis perspective, we found that structural time series models of Facebook Prophet generate stable and high quality out-of-sample prediction without requiring advanced techniques or data assumptions. In addition, we also found that it even achieves higher accuracy in-sample fitted data when we add an extra regressor to structural time series models. Unfortunately, we cannot create out-sample prediction with extra regressor. In contrast to structural time series models, Sequence to Sequence model with LSTM neural network cannot create stable out-of-sample prediction. As **Figure 8** point out, in some cases, Sequence to Sequence model captures movement of stocks to generate high accuracy prediction with lower error than structural time series models. However, the model cannot constantly capture movement of stocks in some other cases. In terms of computational performance, Sequence to Sequence model also takes more time for training and predicting than structural time series models. It leads to a gap to leveraging the state-of-the-art technique for time series prediction. Fortunately, results from **Table 3** suggest that we can fill gaps of structural time series models and Sequence to Sequence model by adding output from structural time series models to Sequence to Sequence model. **Figure 8** show that the model is stable and prediction error of proposed model is almost always lowest among in three models. In terms of benchmark limitation, there are some drawbacks in this benchmark. On the one hand, it is lack of residual analysis for each prediction. We only compute Mean Square Error (MSE) for performance comparison. The evaluated results are not consistent enough to be fully accurate as some outlier points as **Figure 9** point out. On the other hand, although the results are clear and useful when we use MSE as an indicator for forecasting accuracy evaluation, these forecasting evaluation criteria cannot be discriminated between forecasting models when errors of the forecast data are very close to each other. Thus, the Chong and Hendry encompassing test for nested models[49] should be carried out to evaluate the statistical significance of the forecasting models.

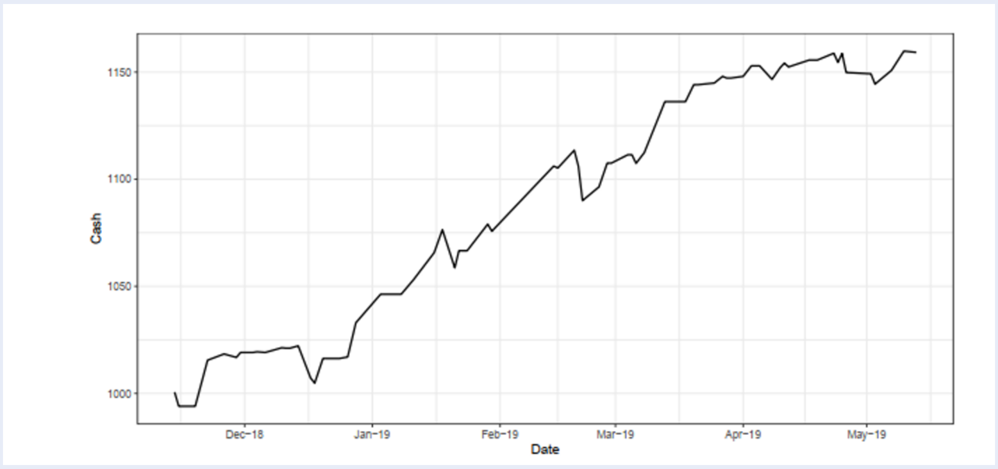


Figure 7: Uncompounded daily cumulative profit of VN30F1M trading.

Table 1: Mean squared error of structural time series model forecast from 5 to 45 window time steps ahead in log-scale

	5	10	15	20	25	30	35	40	45
ACB	0.014721	0.014147	0.015933	0.025314	0.028773	0.029434	0.034342	0.048010	0.058335
BID	0.024936	0.023140	0.014819	0.012375	0.014865	0.013050	0.014286	0.021603	0.023775
BVH	0.016722	0.022044	0.037414	0.046589	0.047791	0.047427	0.048169	0.052609	0.053467
CII	0.015892	0.010909	0.017354	0.025683	0.038861	0.049060	0.050928	0.056750	0.058097
CTD	0.020805	0.038501	0.043919	0.048860	0.056231	0.062125	0.070704	0.081036	0.082875
CTG	0.008746	0.030208	0.054718	0.065659	0.067614	0.068867	0.068824	0.084200	0.087787
DHG	0.024497	0.018136	0.037780	0.043572	0.043265	0.042221	0.044850	0.047424	0.051320
EIB	0.027591	0.027747	0.023507	0.023673	0.029305	0.030964	0.029256	0.021934	0.018138
FPT	0.012170	0.009933	0.009485	0.009722	0.010321	0.011376	0.011212	0.010736	0.010837
GAS	0.012885	0.016930	0.035988	0.047679	0.052466	0.057323	0.065196	0.078481	0.080781
HCM	0.047436	0.057021	0.064199	0.071993	0.069983	0.069544	0.070894	0.080751	0.081232
HPG	0.031256	0.030599	0.031586	0.044547	0.047608	0.050945	0.055629	0.067218	0.076217
MBB	0.015255	0.010532	0.027337	0.032361	0.031718	0.031117	0.032678	0.046915	0.052013
MSN	0.029451	0.022273	0.014707	0.011364	0.012848	0.012974	0.014851	0.022448	0.026552
PNJ	0.006993	0.007920	0.013121	0.016369	0.022160	0.028659	0.045313	0.065240	0.077773
PPC	0.030586	0.030464	0.024987	0.022773	0.029191	0.037727	0.045555	0.054872	0.056164
REE	0.019733	0.016948	0.015939	0.014662	0.012662	0.013016	0.011561	0.011009	0.013720
SBT	0.009171	0.009984	0.020123	0.023605	0.026334	0.026340	0.024738	0.013508	0.021442
SSI	0.003501	0.010974	0.025759	0.033560	0.035462	0.038449	0.042812	0.058004	0.059730
VCB	0.043887	0.052832	0.048772	0.038443	0.031976	0.029193	0.024631	0.011838	0.017969
VNM	0.049289	0.055314	0.049015	0.043668	0.047589	0.050864	0.054577	0.048458	0.042112

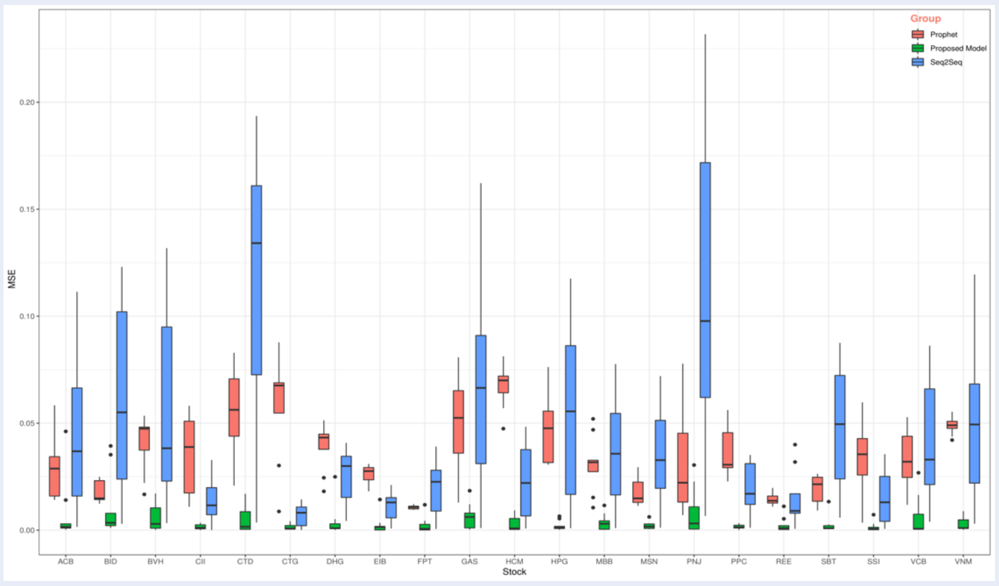
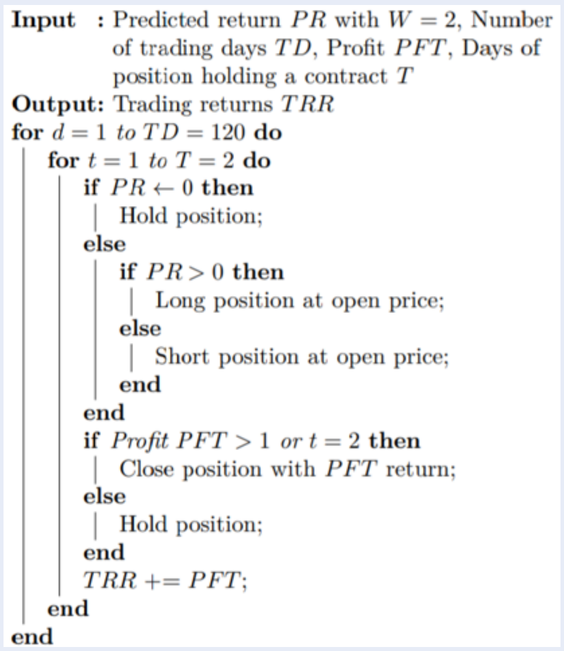


Figure 8: **Error of out-of-sample forecasts in log-scale.**



```
Input  : Predicted return $PR$ with $W = 2$, Number
         of trading days $TD$, Profit $PFT$, Days of
         position holding a contract $T$
Output: Trading returns $TRR$
for $d = 1$ to $TD = 120$ do
    for $t = 1$ to $T = 2$ do
        if $PR \leftarrow 0$ then
            Hold position;
        else
            if $PR > 0$ then
                Long position at open price;
            else
                Short position at open price;
            end
        end
        if Profit $PFT > 1$ or $t = 2$ then
            Close position with $PFT$ return;
        else
            Hold position;
        end
        $TRR$ += $PFT$;
    end
end
```

Figure 9: **Backtest trading strategy.**

Table 2: Mean squared error of sequence to sequence model forecast from 5 to 45 window time steps ahead in log-scale

	5	10	15	20	25	30	35	40	45
ACB	0.001549	0.010649	0.015936	0.036856	0.066454	0.088031	0.065661	0.023986	0.111370
BID	0.002975	0.023858	0.022093	0.055056	0.045212	0.055175	0.123015	0.111880	0.102072
BVH	0.003339	0.007065	0.022930	0.037389	0.038270	0.048385	0.094943	0.131791	0.118588
CII	0.000068	0.000949	0.007197	0.011593	0.008407	0.015061	0.032799	0.024295	0.019871
CTD	0.003564	0.031081	0.093107	0.072614	0.134139	0.160979	0.193552	0.157608	0.180879
CTG	0.000067	0.001404	0.002087	0.008134	0.003509	0.010062	0.014351	0.011942	0.010831
DHG	0.004307	0.015290	0.027839	0.005491	0.031615	0.040781	0.029965	0.034531	0.036360
EIB	0.000734	0.004151	0.005739	0.015192	0.009920	0.018238	0.014715	0.021097	0.012947
FPT	0.000538	0.004052	0.008909	0.011313	0.022583	0.027503	0.039039	0.028022	0.028253
GAS	0.001004	0.021844	0.031094	0.073882	0.090982	0.066480	0.057572	0.106243	0.162174
HCM	0.000771	0.005351	0.012451	0.006651	0.022035	0.030805	0.048325	0.047193	0.037641
HPG	0.000912	0.016714	0.055040	0.013062	0.073548	0.086221	0.099066	0.055516	0.117556
MBB	0.000955	0.008257	0.016460	0.021182	0.035717	0.054560	0.038501	0.064305	0.077589
MSN	0.001165	0.008558	0.019539	0.023428	0.032733	0.051309	0.039536	0.061719	0.071979
PNJ	0.009910	0.006657	0.061987	0.097721	0.094561	0.124379	0.171758	0.173592	0.231800
PPC	0.001150	0.002104	0.014166	0.012028	0.017004	0.027315	0.031092	0.031779	0.035133
REE	0.000612	0.005939	0.008998	0.008640	0.012904	0.017005	0.007903	0.031858	0.039960
SBT	0.005828	0.017245	0.031245	0.023963	0.067473	0.087509	0.049534	0.072302	0.079048
SSI	0.000589	0.004103	0.003022	0.011251	0.014304	0.013024	0.026790	0.025110	0.035506
VCB	0.003926	0.009320	0.032977	0.021303	0.023619	0.066008	0.039739	0.086166	0.084254
VNM	0.003010	0.021958	0.021868	0.043893	0.065320	0.049399	0.068308	0.087976	0.119454

However, there is no package in Python supporting the test at this time, the test was not carried out to conduct appropriated benchmark in terms of statistics. In addition, Diebold-Mariano (DM) test for comparing predictive accuracy (not for comparing models) cannot be applied as it only works for non-nested models[50,51]. Hence, we develop a back testing for the best model as our benchmark suggest (i.e. the proposed model) with real market data in different asset class to relax this limitation.

Overall, in same window size, the combination of structural time series models and Sequence to Sequence model are always achieve high performance than pure structural time series models and Sequence to Sequence model. However, in some cases, the hybrid model cannot capture movement of stock when market is highly volatile.

CONCLUSION AND DISCUSSION

In this work, we generally discussed a set of procedures to model and predict price of stocks in Vietnam stock market with structural time series models and Sequence to Sequence model and the combination of these models. Specifically, we fit stock prices data with structural time series models then use fitted data as input feature of Sequence to Sequence model and generate out-sample prediction. We used output of models to compare accuracy performance of each model. We found that our proposed model can overcome limitations of each model and generate forecast with higher accuracy. The proposed model also achieves positive results for derivatives trading with real market data. Hence, the combination of Long Short-term memory and structural time series model is applicable to Vietnam stock markets.

Furthermore, deep learning is a powerful approach to address time series problems. However, without fea-

Table 3: Mean squared error of proposed model forecast from 5 to 45 window time steps ahead in log-scale

	5	10	15	20	25	30	35	40	45
ACB	0.000243	0.001528	0.000953	0.002946	0.000294	0.001339	0.002150	0.014070	0.046146
BID	0.002119	0.001047	0.003420	0.002281	0.007414	0.007904	0.039367	0.035262	0.002000
BVH	0.000145	0.007243	0.000876	0.000427	0.011959	0.002539	0.010426	0.017116	0.002917
CII	0.000006	0.002513	0.000787	0.001118	0.001933	0.000599	0.000063	0.002505	0.003436
CTD	0.001580	0.000395	0.000173	0.000329	0.008648	0.000776	0.016954	0.015527	0.002938
CTG	0.001237	0.000776	0.000641	0.002322	0.000283	0.000287	0.000458	0.002171	0.004216
DHG	0.000081	0.002963	0.000692	0.000894	0.005131	0.002980	0.000729	0.000448	0.024931
EIB	0.000022	0.001645	0.000096	0.001451	0.000103	0.001309	0.003419	0.001766	0.014310
FPT	0.000018	0.000009	0.002727	0.000098	0.000629	0.000087	0.004376	0.011860	0.001626
GAS	0.001049	0.000242	0.006162	0.001701	0.007973	0.000451	0.012106	0.007322	0.018421
HCM	0.000120	0.000814	0.000365	0.006616	0.000347	0.000154	0.005450	0.009231	0.003637
HPG	0.001480	0.000214	0.000307	0.000747	0.000886	0.001769	0.005409	0.006507	0.001292
MBB	0.000151	0.000234	0.002842	0.000414	0.003575	0.004369	0.011552	0.002987	0.007829
MSN	0.002896	0.001696	0.000507	0.001222	0.003149	0.006190	0.000697	0.002198	0.000785
PNJ	0.000267	0.003331	0.000503	0.000599	0.010909	0.003138	0.030432	0.022743	0.000275
PPC	0.000106	0.001729	0.001398	0.002341	0.001320	0.000454	0.003241	0.002314	0.000987
REE	0.000349	0.001104	0.000185	0.000304	0.000936	0.000123	0.002099	0.005302	0.011126
SBT	0.000863	0.002671	0.001414	0.000694	0.000526	0.000598	0.000768	0.013324	0.002088
SSI	0.000214	0.000316	0.000056	0.000131	0.001353	0.002919	0.000708	0.000588	0.007224
VCB	0.000439	0.000612	0.000192	0.000652	0.016464	0.007455	0.000140	0.002109	0.026819
VNM	0.000237	0.000130	0.008860	0.005173	0.002549	0.000879	0.000835	0.004840	0.000963

ture engineering, deep learning generates prediction lower accuracy than structural time series models. In future work, we will improve that model to achieve real-time prediction to apply for quantitative trading. In addition, we believe that Generative Adversarial Networks (GAN) is a promising approach to apply.

ACKNOWLEDGEMENT

This paper is written under Grant number CS/2019-04 funding from the University of Economics and Law, VNU-HCM. Besides, we also would like to thank the John Von Neumann Institute for their support throughout the project.

ABBREVIATIONS

ARMA: Auto-regressive–moving-average
GARCH: Generalized Auto-regressive Conditional Heteroskedasticity
RNN: Recurrent Neural Network
LSTM: Long Short-term Memory
Seq2Seq: Sequence to sequence
GAN: Generative Adversarial Networks
MSE: Mean Squared Error
HOSE: Ho Chi Minh Stock Exchange
HNX: Hanoi Stock Exchange

COMPETING INTERESTS

The authors declare that they have no conflict of interest.

AUTHORS' CONTRIBUTIONS

Quoc Luu and Uyen Pham initiate the idea, study relevant models and seek for the data. Quoc Luu and Son Nguyen build the main programs for numerical simulations. All authors check the simulation and contribute for the interpretation of the results as well as edit, revise the text and approve the article.

REFERENCES

1. Zhang Y, Tan X, Xi H, Zhao X. Real-time risk management based on time series analysis. In2008 7th World Congress on Intelligent Control and Automation Jun 25. IEEE. 2008;p. 2518–2523.
2. Giamouridis D, Vrontos I. Hedge fund portfolio construction: A comparison of static and dynamic approaches. Journal of Banking & Finance. 2007;31(1):199–217. Available from: https://doi.org/10.1016/j.jbankfin.2006.01.002.
3. Zhou X, Pan Z, Hu G, Tang S, Zhao C. Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering. 2018;Available from: https://doi.org/10.1155/2018/4907423.
4. Box G, Jenkins G, Reinsel G, Ljung G. Time series analysis: forecasting and control. John Wiley & Sons. 2015;.
5. Harvey A, Todd P. Forecasting economic time series with structural and Box-Jenkins models: A case study. Journal of Business & Economic Statistics. 1983;1(4):299–307. Available from: https://doi.org/10.1080/07350015.1983.10509355.
6. Géron A. Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems. O'Reilly Media, Inc. 2017;.
7. Nelson C, Kang H. Pitfalls in the Use of Time as an Explanatory Variable in Regression. Journal of Business & Economic Statistics. 1984;2(1):73–82. Available from: https://doi.org/10.1080/07350015.1984.10509371.
8. Harvey A, Peters S. Estimation procedures for structural time series models. Journal of Forecasting. 1990;9(2):89–108. Available from: https://doi.org/10.1002/for.3980090203.
9. Jalles J. Structural time series models and the Kalman Filter: a concise review. 2009;Available from: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1496864.
10. Taylor S, and BL. Forecasting at scale. The American Statistician. 2018;72(1):37–45. Available from: https://doi.org/10.1080/00031305.2017.1380080.
11. Harvey A. Trends and cycles in macroeconomic time series. Journal of Business & Economic Statistics. 1985;3(3):216–227. Available from: https://doi.org/10.1080/07350015.1985.10509453.
12. Kitagawa G, Gersch W. A smoothness priors-state space modeling of time series with trend and seasonality. Journal of the American Statistical Association. 1984;79(386):378–389. Available from: https://doi.org/10.1080/01621459.1984.10478060.
13. Harrison P, Stevens C. Bayesian forecasting. Journal of the Royal Statistical Society: Series B (Methodological). 1976;38(3):205–228. Available from: https://doi.org/10.1111/j.2517-6161.1976.tb01586.x.
14. Koopman S, Ooms M. Forecasting economic time series using unobserved components time series models. 2011;p. 129–162. Available from: https://doi.org/10.1093/oxfordhb/9780195398649.013.0006.
15. Lipton Z, Berkowitz J, Elkan C. A critical review of recurrent neural networks for sequence learning. arXiv preprint arXiv:150600019. 2015;.
16. Rumelhart D, Hinton G, Williams R. Learning representations by back-propagating errors. Cognitive modeling. 1988;5(3):1.
17. Elman J. Finding structure in time. Cognitive science. 1990;14(2):179–211. Available from: https://doi.org/10.1207/s15516709cog1402_1.
18. Haykin S, Principe J, Sejnowski T, Mcwhirter J. Modeling Large Dynamical Systems with Dynamical Consistent Neural Networks;.
19. Goodfellow I, Bengio Y, Courville A. Deep learning. MIT press. 2016;.
20. Jordan MI. Serial order: a parallel distributed processing approach. Technical report, June 1985-March 1986. California Univ., San Diego, La Jolla (USA). Inst for Cognitive Science. 1986;.
21. Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:160904747. 2016;.
22. Qian N. On the momentum term in gradient descent learning algorithms. Neural networks. 1999;12(1):145–151. Available from: https://doi.org/10.1016/S0893-6080(98)00116-6.
23. Yu N. Introductory lectures on convex optimization: a basic course. Springer US. 2004;.
24. Hochreiter S, Schmidhuber J. Long short-term memory. Neural computation. 1997;9(8):1735–1780. PMID: 9377276. Available from: https://doi.org/10.1162/neco.1997.9.8.1735.
25. Nair V, Hinton G. Rectified linear units improve restricted boltzmann machines. InProceedings of the 27th international conference on machine learning (ICML-10). 2010;p. 807–814.
26. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition. 2016;p. 770–778. Available from: https://doi.org/10.1109/CVPR.2016.90PMid:26180094.
27. Hochreiter S. Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München. 1991;91(1).
28. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks. 1994;5(2):157–166. PMID: 18267787. Available from: https://doi.org/10.1109/72.279181.
29. Harmon M, Klabjan D. Dynamic prediction length for time series with sequence to sequence networks. arXiv preprint arXiv:180700425. 2018;.
30. Zhou C, Sun C, Liu Z, Lau F. A C-LSTM neural network for text classification. arXiv preprint arXiv:151108630. 2015;.
31. Graves A, Jaitly N, Mohamed A. Hybrid speech recognition with deep bidirectional LSTM. In 2013 IEEE workshop on automatic speech recognition and understanding. IEEE. 2013;p. 273–278. Available from: https://doi.org/10.1109/ASRU.2013.6707742.
32. Gers F, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM;.
33. Greff K, Srivastava R, Koutník J, Steunebrink B, Schmidhuber J. LSTM: A search space odyssey. IEEE transactions on neural networks and learning systems. 2016;28(10):2222–32. PMID: 27411231. Available from: https://doi.org/10.1109/TNNLS.2016.2582924.
34. Graves A. Supervised sequence labelling. InSupervised sequence labelling with recurrent neural networks. Springer, Berlin, Heidelberg. 2012;p. 5–13. Available from: https://doi.org/10.1007/978-3-642-24797-2_2.
35. Chollet FK. Deep learning library for theano and tensorflow. 2015;7(8). Available from: https://keras.io/k.
36. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. In12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16). 2016;p. 265–283.
37. Cho K, Merriënboer BV, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078. 2014;Available from: https://doi.org/10.3115/v1/D14-1179.
38. Britz D, Goldie A, Luong M, Le Q. Massive exploration of neural machine translation architectures. arXiv preprint arXiv:170303906. 2017;Available from: https://doi.org/10.18653/v1/D17-1151.
39. van der Westhuizen J, Lasenby J. The unreasonable effectiveness of the forget gate. arXiv preprint arXiv:180404849. 2018;.
40. Sutskever I, Vinyals O, Le Q. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. 2014;p. 3104–3112.
41. Venugopalan S, Rohrbach M, Donahue J, Mooney R, Darrell T, Saenko K. Sequence to sequence-video to text. InProceedings of the IEEE international conference on computer vision;p. 4534–4542. Available from: https://doi.org/10.1109/ICCV.2015.515.
42. Tang Y, Xu J, Matsumoto K, Ono C. Sequence-to-sequence model with attention for time series classification. In2016 IEEE

16th International Conference on Data Mining Workshops (ICDMW) . IEEE. 2016;p. 503–510. Available from: https://doi.org/10.1109/ICDMW.2016.0078.

43. Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:14090473. 2014;.
44. Yang Y, Linkens D, Mahfouf M. Nonlinear Data Transfonnation to Improve Flow Stress Prediction Using Neural Networks. IFAC Proceedings Volumes. 2004;37(15):371–376. Available from: https://doi.org/10.1016/S1474-6670(17)31052-2.
45. Changyong F, Hongyue W, Naiji L, Tian C, Hua H, Ying L. Log-transformation and its implications for data analysis. . Shanghai archives of psychiatry. 2014;26(2):105.
46. Kim K. Financial time series forecasting using support vector machines. Neurocomputing. 2003;55(1-2):307–319. Available from: https://doi.org/10.1016/S0925-2312(03)00372-2.
47. Facebook. Prophet Notebook;.
48. Hinton G. Nitish Srivastava Alex Krizhevsky Ilya Sutskever and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. CoRR. 2012;.
49. Chong Y, Hendry D. Econometric evaluation of linear macro-economic models. The Review of Economic Studies. 1986;1;53(4):671–690. Available from: https://doi.org/10.2307/2297611.
50. Vavra M. On a Bootstrap Test for Forecast Evaluations. Research Department, National Bank of Slovakia. 2015;.
51. Diebold F. Comparing predictive accuracy, twenty years later: A personal perspective on the use and abuse of Diebold-Mariano tests. Journal of Business & Economic Statistics. 2015;33(1):1. Available from: https://doi.org/10.1080/07350015.2014.983236.

Tạp chí Phát triển Khoa học và Công nghệ – Kinh tế-Luật và Quản lý, 4(1):500-515

Open Access Full Text Article

Bài Nghiên cứu

Dự báo chuỗi thời gian: sự kết hợp giữa mô hình Long Short-term Memory và mô hình cấu trúc chuỗi thời gian

Lưu Hoài Thương Quốc[1,*], Nguyễn Phúc Sơn[2], Phạm Hoàng Uyên[1,2]

TÓM TẮT

Thị trường chứng khoán là một kênh huy động vốn quan trọng cho nền kinh tế. Tuy nhiên, thị trường có một sự mất mát tiềm tàng do sự biến động của giá cổ phiếu để phản ánh các sự kiện không chắc chắn như tin tức chính trị, nguồn cung và nhu cầu của khối lượng giao dịch hàng ngày. Có nhiều cách khác nhau để giảm rủi ro như xây dựng và tối ưu hóa danh mục đầu tư, phát triển chiến lược phòng ngừa rủi ro. Vì thế kỹ thuật dự báo chuỗi thời gian có thể rất hữu ích nhằm giúp cải thiện hiệu suất lợi nhuận cao hơn trên thị trường chứng khoán. Gần đây, thị trường chứng khoán Việt Nam ngày càng được chú ý bởi hiệu suất đầu tư và vốn hóa đang được cải thiện. Trong nghiên cứu này, chúng tôi đề xuất mô hình kết hợp giữa mô hình Sequence to Sequence với kiến trúc mạng bộ nhớ dài-ngắn (Long Short-Term Memory) của học sâu và mô hình cấu trúc chuỗi thời gian. Chúng tôi dùng dữ liệu giá của 21 cổ phiếu được niêm yết có giao dịch nhiều nhất trên sàn giao dịch chứng khoán Hồ Chí Minh (HOSE) và sàn giao dịch chứng khoán Hà Nội (HNX) của thị trường chứng khoán Việt Nam để đánh giá độ chính xác của mô hình đề xuất với mô hình Sequence to Sequence và mô hình cấu trúc chuỗi thời gian thuần. Mặt khác, để kiểm tra lại tính ứng dụng của mô hình trong môi trường đầu tư thực tế, chúng tôi dùng mô hình đề xuất cho quyết định mua (Long) hay bán (Short) hợp đồng tương lai VN30F1M (hợp đồng tương lai chỉ số VN30 kỳ hạn một tháng) được niêm yết trên sàn HNX. Kết quả cho thấy mô hình đề xuất kết hợp giữa Sequence to Sequence với kiến trúc mạng bộ nhớ dài-ngắn và mô hình cấu trúc chuỗi thời gian đạt hiệu quả cao hơn với sai số nhỏ hơn các mô hình thuần trong việc dự báo giá chứng khoán và có lời đối với giao dịch hợp đồng tương lai. Nghiên cứu này có ý nghĩa tích cực trong việc đóng góp vào cơ sở lý luận của dự báo chuỗi thời gian bởi phương pháp được đề xuất trong nghiên cứu này giúp bỏ qua những giả định khó thoản mãn trong môi trường tài chính thực tế của các phương pháp hiện tại như Auto-regressive–moving-average model, Generalized Auto-regressive Conditional Heteroskedasticity. Về mặt ứng dụng, các nhà đầu tư có thể sử dụng mô mình để phát triển các chiến thuật để giao dịch trên thị trường chứng khoán Việt Nam.

Từ khoá: LSTM, Seq2Seq, Mô hình cấu trúc, mô hình kết hợp

[1]*Tài Chính Tính Toán Định Lượng, Viện John von Neumann, Thành phố Hồ Chí Minh, Việt Nam*

[2]*Toán Kinh Tế, Trường Đại học Kinh tế Luật, Thành phố Hồ Chí Minh, Việt Nam*

Liên hệ

Lưu Hoài Thương Quốc, Tài Chính Tính Toán Định Lượng, Viện John von Neumann, Thành phố Hồ Chí Minh, Việt Nam

Email: quoc.luu2015@qcf.jvn.edu.vn

Lịch sử

- Ngày nhận: 28/6/2019
- Ngày chấp nhận: 25/9/2019
- Ngày đăng: 31/3/2020

DOI : 10.32508/stdjelm.v4i1.593