May 2020

An Efficient Hybrid Mechanism with LSTM Neural Networks in Application to Stock Price Forecasting

Ngoc-An NGUYEN-PHAM [a,b,c,1], and Trung T. NGUYEN [b,c]
[a] *AISIA Research Lab, Ho Chi Minh City, Vietnam*
[b] *University of Science, Ho Chi Minh City, Vietnam*
[c] *Vietnam National University in Ho Chi Minh City, Vietnam*

Abstract. Recurrent neural networks (RNN) and long short-term memory (LSTM) neural networks have shown some success with many practical applications in recent years such as machine translation, speech recognition, image processing and financial market forecasting. In recent years, a dual-stage attention-based recurrent neural network (DA-RNN) have shown some promising results on stock price prediction. We propose dual attention-dilated long short-term memory (DAD-LSTM) models combining DA-RNN and dilated recurrent neural networks (DRNN) to select the most relevant input features and capture the long-term temporal dependencies of a time series more efficiently. Numerical results from experiments on the NASDAQ 100, S&P 500, HSI and DJIA datasets show that DAD-LSTM models outperform the state-of-the-art and most recent approaches.

Keywords. stock price forecasting, RNN, LSTM, DA-RNN, DRNN, DAD-LSTM, attention mechanism, skip connections, time series, deep learning.

1. Introduction

Forecasting stock market [1] is still an challenging problem due to the extreme complex and nonlinear volatility. The nature of this problem requires a highly precise predictor to solve for the non-stationary, nonlinear and high-noise multivariate time series. Although traditional statistical models such as the well-known autoregressive moving average (ARMA) model [2] and its variants ARIMA [3] shown their effectiveness for various real world applications, they are not able to capture long-range dependencies and model nonlinear spatial-temporal relationships among various time series. A dramatic increase of deep learning research [4–6] based on long short-term memory (LSTM) neural networks [7] in recent years has led to a new class of predictors that may overcome those drawbacks of traditional statistical models. The notable results of an LSTM neural network are the ability to capture long-term dependencies and resist the problem of vanishing gradients during the back-propagation process. Therefore, it may be well suited for multivariate time series forecasting. In particular, LSTM neural networks can be more efficient in multi-step forecasting.

[1]Corresponding Author; E-mail: 1611009@student.hcmus.edu.vn

A dual-stage attention-based recurrent neural network (DA-RNN) [4] was proposed to adaptively select the most relevant input features and capture the long-term temporal dependencies of a time series appropriately. An RNN unit can be a traditional RNN [8–10], LSTM [7] or gated recurrent unit (GRU) [11]. In this article, we only focus on the LSTM unit to capture long-term dependencies. The architecture of a DA-RNN neural network consists of two stages which are spatial and temporal attentions mechanisms. In the first stage, a spatial attention mechanism is introduced to adaptively extract relevant driving series at each time step by referring to the previous encoder hidden state. In the second stage, we use a temporal attention mechanism to select relevant encoder hidden states across all time steps. The output the last decoder LSTM unit is the predicted result.

Inspired by the DA-RNN neural network, we propose a new dual attention-dilation LSTM (DAD-LSTM) model that are still based upon the two-stage attention mechanism while the encoder and/or decoder may use the dilated recurrent neural network (DRNN) [12] or a single layer of LSTM units depending on the characteristic of each dataset. The DRNN neural network stacks multiple dilated recurrent layers with hierarchical dilations to learn temporal dependencies of different scales at different layers and use dilated recurrent skip connections to alleviate gradient problems, extend the range of temporal dependencies and significantly enhance computational efficiency.

In order to justify the reliability and effectiveness of the DAD-LSTM, we compare it with state-of-the-art (SOTA) DA-RNN and most recent approaches such as DSTP-RNN [13] and WLSTM+Attention [1] using four stock datasets including National Association of Securities Dealers Automated Quotations System (NASDAQ 100) [14], Standard & Poor (S&P 500) [15], Hang Seng index (HSI) [16], and Dow Jones Industrial Average (DJIA) [17]. Extensive experiments not only demonstrate the reliability and effectiveness of the proposed model, but also show that the DAD-LSTM neural networks outperform all proposed methods in [1, 4, 13].

The outline of the paper is as follow. Section 2 introduces some notions of RNN, LSTM, DRNN neural networks and the architecture of DAD-LSTM models. Section 3 discusses experiments with the description of datasets, training procedure, parameter setting, evaluation metrics and numerical results. Section 4 concludes our work and present some perspectives in future research.

2. Model

2.1. RNN and LSTM Neural Network

The recurrent neural network (RNN), *a.k.a* Auto Associative or Feedback Network [8–10, 18], belongs to a class of artificial neural networks that can remember the past information to feed as input to the current step. The update of an RNN unit has the form

$$\mathbf{h}_t = \mathscr{G}\left(\mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hx}\mathbf{x}_t + \mathbf{b}_h\right) \tag{1}$$

where $\mathbf{h}_t$ and $\mathbf{x}_t$ in $\mathbb{R}^m$ are hidden state and input at time t, $\mathbb{R}$ is real number set and m is a positive integer. The activation function $\mathscr{G}$ can be a logistic sigmoid, tanh or ReLU function. The weights $\mathbf{W}_{hh} \in \mathbb{R}^{m\times m}, \mathbf{W}_{hx} \in \mathbb{R}^{m\times n}$ and bias $\mathbf{b}_h \in \mathbb{R}^m$ are parameters to learn. Although RNNs have shown their success in time series forecasting [19, 20], they still suffer from the problem of vanishing gradients [21] and thus have difficulty in capturing long-term dependencies. In order to overcome this drawback, we can change the architecture of an RNN cell to an LSTM one.

The LSTM cell $\mathscr{F}(\cdot)$ filters information through the gate structure to update the state of memory cells. Its door structure including input $\mathbf{i}_t$, forgotten $\mathbf{f}_t$, and output gates $\mathbf{o}_t$ are vector in $\mathbb{R}^m$ where the positive integer m is the size of hidden states. Each memory cell has three sigmoid and one tanh layers. The update of an LSTM cell $\mathbf{h}_t = \mathscr{F}(\mathbf{h}_{t-1}; \mathbf{x}_t)$ is given by

$$\mathbf{f}_t = \sigma\left(\mathbf{W}_f\left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}_f\right) \tag{2a}$$

$$\mathbf{i}_t = \sigma\left(\mathbf{W}_i\left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}_i\right) \tag{2b}$$

$$\mathbf{o}_t = \sigma\left(\mathbf{W}_o\left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}_o\right) \tag{2c}$$

$$\mathbf{s}_t = \mathbf{f}_t \odot \mathbf{s}_{t-1} + \mathbf{i}_t \odot \tanh\left(\mathbf{W}_s\left[\mathbf{h}_{t-1}; \mathbf{x}_t\right] + \mathbf{b}_s\right) \tag{2d}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh\left(\mathbf{s}_t\right) \tag{2e}$$

where σ stands for logistic sigmoid function and $\odot$ is elementwise multiplication. The three sigmoid gates ($\mathbf{f}_t$, $\mathbf{i}_t$ and $\mathbf{o}_t$) control the access to the memory with the cell state $\mathbf{s}_t \in \mathbb{R}^m$ and the hidden state $\mathbf{h}_t \in \mathbb{R}^m$. The term $[\mathbf{h}_{t-1}; \mathbf{x}_t] \in \mathbb{R}^{m+n}$ stands for a concatenation of the previous hidden state $\mathbf{h}_{t-1} \in \mathbb{R}^m$ and the current input $\mathbf{x}_t \in \mathbb{R}^n$ where n is positive integer. Parameters to learn $\mathbf{W}_f, \mathbf{W}_i, \mathbf{W}_o, \mathbf{W}_s \in \mathbb{R}^{m\times(m+n)}$, and $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_o, \mathbf{b}_s \in \mathbb{R}^m$ stand for weights and biases corresponding to each gate or state. The sums activities over time of the cell state in the LSTM unit is the key idea to overcome the problem of vanishing gradients and better capture long-term dependencies of time series [7]. Based on each LSTM unit, the architecture of an unroll LSTM single layer [22] is given in Figure 1.

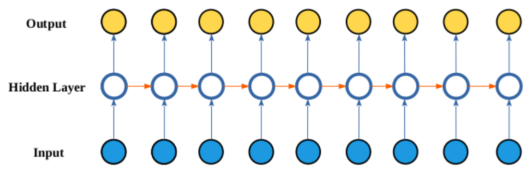


Figure 1. LSTM single layer: each white cell in hidden layer is an LSTM unit.

2.2. Dilated Recurrent Neural Network (DRNN)

In the architecture of DRNN [12], we utilize dilated recurrent skip connections then stack multiple dilated recurrent layers with hierarchical dilations to learn temporal dependencies of different scales at different layers. The DRNN architecture is similar to DenseNet [23] with cross connections however information travel along fewer edges. The dilated recurrent skip connection helps to alleviate gradient problems and extend the range of temporal dependencies but require fewer parameters versus conventional recurrent skip connections. An update of DRNN neural network can be represented as

$$\mathbf{h}_t^{(l)} = \mathscr{F}\left(\mathbf{x}_t^{(l)}, \mathbf{h}_{t-s^{(l)}}^{(l)}\right) \tag{3}$$

where the positive integer $s^{(l)}$ stands for the dilation of layer l, $\mathbf{h}_t^{(l)}$ and $\mathbf{x}_t^{(l)}$ are the hidden state and input of layer l at time t, respectively, and $\mathscr{F}(\cdot)$ is an LSTM cell (2). The architecture of DRNNs is given in Figure 2 consisting of three layers. The main difference between the architecture of DRNN in [12] and this paper is that we apply an unroll LSTM layer with the dilation one. The two other layers remain unroll RNNs. For the rest of the paper, we use the term DLSTM to mention our modified DRNN.

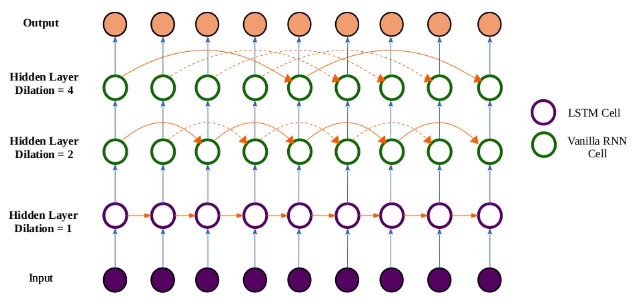

Output
Hidden Layer Dilation = 4
Hidden Layer Dilation = 2
Hidden Layer Dilation = 1
Input
LSTM Cell
Vanilla RNN Cell

Figure 2. DLSTM with three hidden layers: the first layer is an LSTM neural network with Dilation=1, the second and third hidden layers are RNN with Dilation=2 and Dilation=4, respectively.

2.3. *DAD-LSTM Neural Network*

We address in this section the full architecture of our proposed dual attention-dilation long short-term memory (DAD-LSTM) neural network. Given n driving series $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n)^\top = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_T) \in \mathbb{R}^{n\times T}$ where T is the length of the time window and $\top$ stands for the transpose of the given matrix, we denote $\mathbf{x}^k = (x_1^k, x_2^k, \ldots, x_T^k)^\top \in \mathbb{R}^T$ the k-th driving series of length T, and $\mathbf{x}_t = (x_t^1, x_t^2, \ldots, x_t^n)^\top \in \mathbb{R}^n$ the vector of n driving input series at time t. The architecture of DAD-LSTM consisting of two stages: the encoder with spatial attention Figure 3 and the decoder with temporal attention Figure 4.

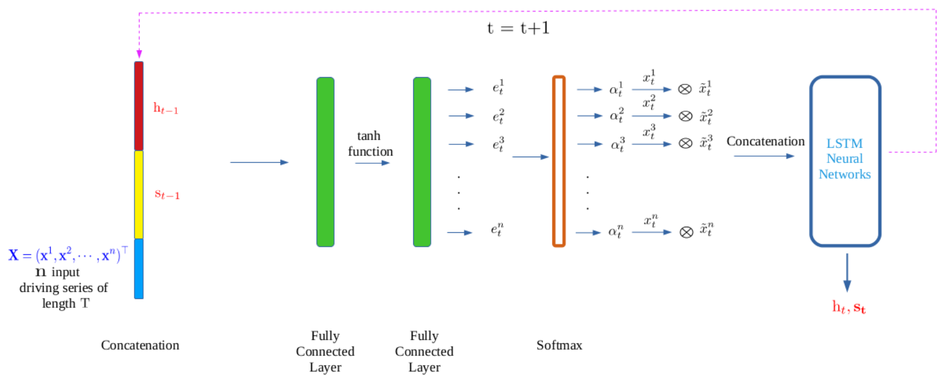

t = t+1
h_{t-1}
s_{t-1}
$\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^n)^\top$
n input driving series of length T
tanh function
e_t^1 e_t^2 e_t^3 e_t^n
α_t^1 α_t^2 α_t^3 α_t^n
x_t^1 x_t^2 x_t^3 x_t^n
$\otimes\ \tilde{x}_t^1$ $\otimes\ \tilde{x}_t^2$ $\otimes\ \tilde{x}_t^3$ $\otimes\ \tilde{x}_t^n$
Concatenation
LSTM Neural Networks
$\mathrm{h}_t, \mathbf{s}_t$
Concatenation
Fully Connected Layer
Fully Connected Layer
Softmax

Figure 3. Encoder with spatial attention

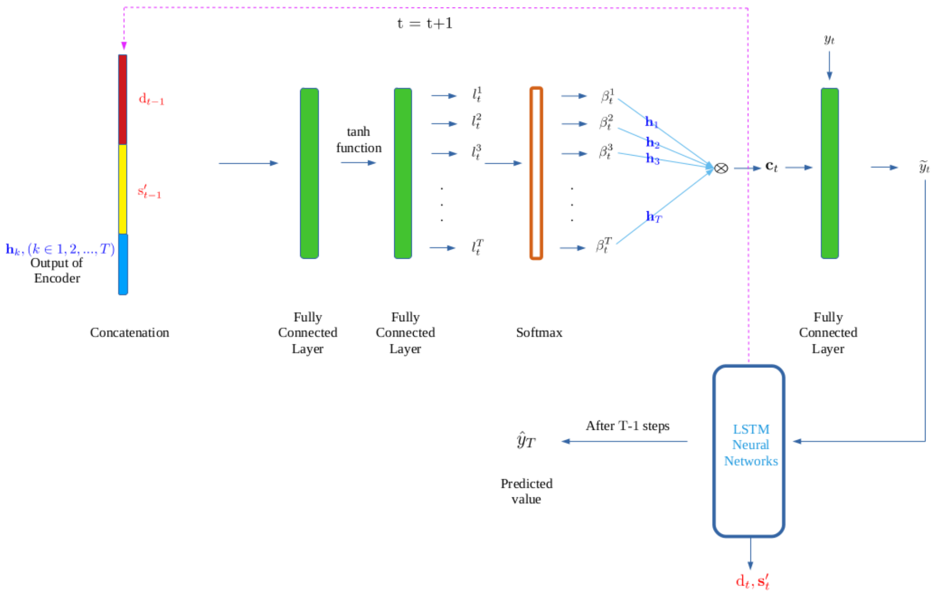


Figure 4. Decoder with temporal attention

In the first stage, we apply the encoder with spatial attention to learn a nonlinear mapping $\mathbf{h}_t = \mathscr{F}_{\text{enc}}(\mathbf{h}_{t-1}, \mathbf{x}_t)$ where $\mathscr{F}_{\text{enc}}$ stands for a single LSTM layer (Figure 1) or DLSTM (Figure 2) of the encoder and the positive integer m is the size of the encoder hidden state. We construct the spatial attention mechanism via a multilayer perceptron (*a.k.a* feed forward network)

$$e_t^k = \mathbf{v}_e^\top \tanh\left(\mathbf{W}_e[\mathbf{h}_{t-1}; \mathbf{s}_{t-1}] + \mathbf{U}_e\mathbf{x}^k + \mathbf{b}_e\right) \tag{4a}$$

$$\alpha_t^k = \frac{\exp\left(e_t^k\right)}{\sum_{i=1}^n \exp\left(e_t^i\right)} \tag{4b}$$

where weights $\mathbf{v}_e \in \mathbb{R}^T, \mathbf{W}_e \in \mathbb{R}^{T\times 2m}$, $\mathbf{U}_e \in \mathbb{R}^{T\times T}$ and bias $\mathbf{b}_e \in \mathbb{R}^T$ are parameters to learn. The attention weight e_t^k measure the importance of the k-th driving series at time t and α_t^k is the encoder attention weights to ensure the unit sum of all attention weights. We train the spatial attention mechanism with other components of each LSTM cell to adaptively extract the driving series with softmax encoder attention weights

$$\tilde{\mathbf{x}}_t = \left(\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \cdots, \alpha_t^n x_t^n\right)^\top \tag{5}$$

Finally, an update of the encoder hidden state at each $t \in \{1,2,\ldots,T\}$ is given by

$$\mathbf{h}_t = \mathscr{F}_{\text{enc}}(\mathbf{h}_{t-1}, \tilde{\mathbf{x}}_t). \tag{6}$$

In the second stage, the decoder with temporal attention is used to adaptively select relevant encoder hidden states across all time steps. Let us denote positive integer p

the dimensions of decoder hidden state. The decoder attention weight β_t^i at time t of each encoder hidden state $\mathbf{h}_i$ is calculated based upon the previous decoder hidden state $\mathbf{d}_{t-1} \in \mathbb{R}^p$ and the cell state of the LSTM unit $\mathbf{s}'_{t-1} \in \mathbb{R}^p$

$$l_t^i = \mathbf{v}_d^\top \tanh\left(\mathbf{W}_d \left[\mathbf{d}_{t-1}; \mathbf{s}'_{t-1}\right] + \mathbf{U}_d \mathbf{h}_i + \mathbf{b}_d\right), \quad 1 \le i \le T \tag{7a}$$

$$\beta_t^i = \frac{\exp\left(l_t^i\right)}{\sum_{j=1}^T \exp\left(l_t^j\right)} \tag{7b}$$

where weights $\mathbf{v}_d \in \mathbb{R}^m$, $\mathbf{W}_d \in \mathbb{R}^{m \times 2p}$, $\mathbf{U}_d \in \mathbb{R}^{m \times m}$ and bias $\mathbf{b}_d \in \mathbb{R}^m$ are parameters to learn. The context vector $\mathbf{c}_t$, for each $t \in \{1, 2, \dots, T-1\}$, is a weighted sum of all the encoder hidden states $\{\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_T\}$

$$\mathbf{c}_t = \sum_{i=1}^T \beta_t^i \mathbf{h}_i. \tag{8}$$

An update of the decoder hidden state at time t is given by

$$\mathbf{d}_t = \mathscr{F}_{\text{dec}}\left(\mathbf{d}_{t-1}, \tilde{y}_t\right) \tag{9}$$

where $\mathscr{F}_{\text{dec}}$ stands for a single LSTM layer (Figure 1) or DLSTM (Figure 2) of the decoder, for each t in $\{1, 2, \dots, T-1\}$, and the decoder input $\tilde{y}_t$ at time t is the combination of the weighted summed context vectors $\mathbf{c}_t$ and the corresponding value of the given target series $(y_1, y_2, \cdots, y_{T-1})$

$$\tilde{y}_t = \tilde{\mathbf{w}}^\top \left[y_t; \mathbf{c}_t\right] + \tilde{b}, \quad 1 \le t \le T-1. \tag{10}$$

The parameters to learn $\tilde{\mathbf{w}} \in \mathbb{R}^{m+1}$ and $\tilde{b} \in \mathbb{R}$ map the concatenation $[y_t; \mathbf{c}_t]$ to the unit size of the decoder input $\tilde{y}_t$.

In the architecture of DAD-LSTM model, we propose to select either a single LSTM layer or DLSTM neural network for the encoder and/or decoder. This choice strongly depends on the characteristic of data and problem. In the context of stock price forecasting, it is more reasonable to use the DLSTM neural network in the encoder and a single LSTM layer in the decoder since we need to extract relevant information from driving series at each time step. Specifically speaking, we expect to know more precisely which stocks have the impact on the target one. We remark that if a DLSTM neural network is used in the decoder, we take the last hidden states to feed into a multilayer perceptron.

Finally, the prediction function is given by a feed forward network

$$\hat{y}_T = \mathbf{v}_y^\top \left(\mathbf{W}_y \left[\mathbf{d}_{T-1}; \mathbf{c}_{T-1}\right] + \mathbf{b}_y\right) + b \tag{11}$$

where weights $\mathbf{W}_y \in \mathbb{R}^{p \times (p+m)}$, $\mathbf{b}_y, \mathbf{v}_y \in \mathbb{R}^p$ and bias $b \in \mathbb{R}$ are parameters to learn.

3. Experiments

3.1. Description of Datasets

We describe in this section four datasets of stock prices including NASDAQ 100 [14], S&P 500 [15], HSI [16] and DJIA [17]. The statistics of these four datasets are shown in Table 1. Each dataset is split exactly in the same way of [1, 4, 13] to compare our methods and the others. We remark that S&P 500, HSI and DJIA datasets in [1] are divided without validation sets. The NASDAQ 100 are split in two ways corresponding to [4] and [13].

Reference	Dataset	Driving Series	size		
			train	valid	test
[4]	NASDAQ 100 (1)	81	35,100	2,730	2,730
[13]	NASDAQ100 (2)	81	32000	0	8560
[1]	S&P500	5	4873	0	32
[1]	DJIA	5	4873	0	31
[1]	HSI	5	4277	0	31

Table 1. The statistics of stock price datasets

NASDAQ 100 [14]: The collection of NASDAQ 100 comprises of 81 time series of 81 corresponding major corporations. Each series contains the index values collected minute-by-minute in total 105 days, from July 26, 2016 to December 22, 2016. Each day contains 390 data points from the opening to closing of the market except that there are 210 data points on November 25 and 180 data points on December 22. In order to compare with DA-RNN method [4], we separate the first 35,100 data points as the training set, the following 2,730 data points as the validation set and the last 2,730 data points are used as the test set. On the other hand, in comparison with DSTP-RNN [13] we use the first 32000 data points as the training set and 8560 data points for test set.

S&P 500 [15]: This dataset is collected from January 3, 2000, to July,1 2019 with 4904 days in total. Data data from January 3, 2000, to May 16,2019 are used for the training set and remaining days are dedicated for the test set. On this dataset, we compare the DAD-LSTM with WLSTM+Attention model [1].

HSI [16]: The HSI data consists of 4308 days in total, from January 2, 2002 to July 1, 2019. The training set is taken from January 2, 2002, to May 16, 2019 and the test set includes data from May 17, 2019, to July 1, 2019. On HSI data, we conduct the comparison between DAD-LSTM and WLSTM+Attention models [1].

DJIA [17]: The DJIA data are collected in the same period with the S&P 500 data, from January 3, 2000, to July 1, 2019. There are 4904 days in total. Data from January 3, 2000, to May 16,2019 are used for the training set and remaining days are dedicated for the test set. We compare the DAD-LSTM model with WLSTM+Attention model [1] on this dataset.

3.2. Training Procedure, Parameter Settings and Evaluation Metrics

In the training procedure, we use stochastic gradient descent (SGD) together with the Adam optimizer model [24, 25]. The batch size is set to 128, learning rate is 0.001, and output size is 1, The size of time window T is set to 10. The size of hidden states for the encoder and decoder is $m = p = 64$. In order to use DLSTM neural network, we add the second and third layers with the dilation is set to 2 and 4, respectively. The loss function is provided using the mean squared error (MSE)

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(y_t^i - \hat{y}_t^i\right)^2. \tag{12}$$

We train each method 10 times over each dataset with the maximum of 101 epochs. After each time of training, the best model can be selected based on the MSE (12). The metrics measurements are computed as the mean and 95% confidence interval of 10 times execution. We implement the proposed DAD-LSTM models in PyTorch framework on a machine with CPU Intel Core i9 - 7900X (10 Cores, 20 Threads), GPU GIGABYTE GeForce GTX 1080 (11GB DDR5) and 64GB RAM DDR4.

Let us consider three common evaluation metrics (13) for time series prediction: root mean squared error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE) to compare and measure the effectiveness of various methods

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_t^i - \hat{y}_t^i\right)^2}, \tag{13a}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_t^i - \hat{y}_t^i\right|, \tag{13b}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{y_t^2 - \hat{y}_t^2}{y_t^i}\right| \times 100\%, \tag{13c}$$

where y_t and $\hat{y}_t$ are the target and predicted value at time t, respectively. For the sake of ease, we show results of MAPE without multiplying with 100%.

3.3. Results

For the sake of simplicity, we define DAD-LSTM-I, DAD-LSTM-II and DAD-LSTM-III corresponding to the use of DLSTM in the encoder, decoder and both, respectively. The best performance is displayed in boldface. We show a hyphen (-) corresponding to the metric which authors does not provide the error response.

Table 2. Compare DAD-LSTM with DSTP-RNN [13] on the NASDAQ 100 data.

Models	NASDAQ 100		
	RMSE	MAE	MAPE
DSTP-RNN ($\tau = 120$) [13]	0.0895	0.0661	0.0613
DSTP-RNN-II ($\tau = 120$) [13]	0.1214	0.0739	0.0731
DSTP-RNN ($\tau = 30$) [13]	0.0987	0.0761	0.1047
DSTP-RNN-II ($\tau = 30$) [13]	0.0661	0.0507	0.0551
DAD-LSTM-I	**0.0303 $\pm$ 0.004**	**0.0209 $\pm$ 0.0034**	**0.0281 $\pm$ 0.0038**
DAD-LSTM-II	0.0785 $\pm$ 0.0211	0.0552 $\pm$ 0.0121	0.0487 $\pm$ 0.0069
DAD-LSTM-III	0.0578 $\pm$ 0.0169	0.0401 $\pm$ 0.0094	0.0373 $\pm$ 0.0050

Table 3. Compare DAD-LSTM with DA-RNN [4] on the NASDAQ 100 data.

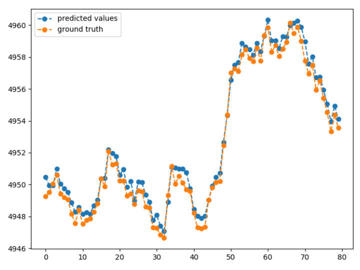
Models	NASDAQ 100		
	RMSE	MAE	MAPE
DA-RNN(64) [4]	0.31 $\pm$ 0.003	0.21 $\pm$ 0.002	0.43 $\pm$ 0.005
DA-RNN(128) [4]	0.33 $\pm$ 0.003	0.22 $\pm$ 0.002	0.45 $\pm$ 0.005
DAD-LSTM-I	**0.0327 $\pm$ 0.0023**	**0.0242 $\pm$ 0.0026**	**0.0122 $\pm$ 0.0013**
DAD-LSTM-II	0.0810 $\pm$ 0.0227	0.0655 $\pm$ 0.0234	0.0360 $\pm$ 0.0107
DAD-LSTM-III	0.0771$\pm$ 0.0236	0.0681 $\pm$ 0.0228	0.0330 $\pm$ 0.0109

Table 4. Compare DAD-LSTM with WLSTM+Attention [1] on the HSI data.

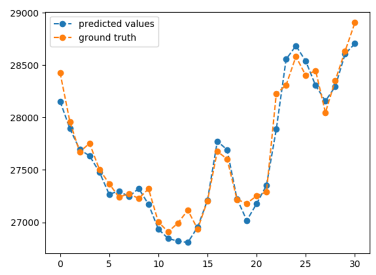
Models	HSI		
	RMSE	MAE	MAPE
WLSTM+Attention [1]	0.3429	0.2433	-
DAD-LSTM-I	**0.0394 $\pm$ 0.0015**	**0.0294 $\pm$ 0.0010**	**0.0215 $\pm$ 0.0008**
DAD-LSTM-II	0.0416 $\pm$ 0.0015	0.0316 $\pm$ 0.0015	0.0231 $\pm$ 0.0011
DAD-LSTM-III	0.0437 $\pm$ 0.0026	0.0340 $\pm$ 0.0019	0.0249 $\pm$ 0.0013

Table 5. Compare DAD-LSTM with WLSTM+Attention [1] on DJIA data.

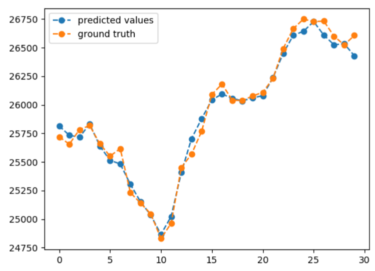
Models	DJIA		
	RMSE	MAE	MAPE
WLSTM+Attention [1]	0.1971	0.1569	-
DAD-LSTM-I	**0.0350 $\pm$ 0.0017**	**0.0275 $\pm$ 0.0017**	**0.0108 $\pm$ 0.0006**
DAD-LSTM-II	0.0373 $\pm$ 0.0035	0.0303 $\pm$ 0.0033	0.0119 $\pm$ 0.0013
DAD-LSTM-III	0.0385 $\pm$ 0.003	0.0314 $\pm$ 0.0027	0.0123 $\pm$ 0.0011



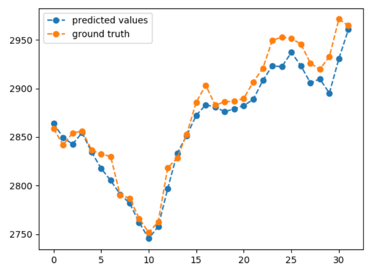
(a) NASDAQ 100



(b) HSI



(c) DJIA



(d) S&P 500

Figure 5. Prediction of DAD-LSTM-I on four datasets: NASDAQ 100, HSI, DJIA and S&P 500. Blue and orange points are ground truth and predicted values, respectively.

Table 6. Compare DAD-LSTM with WLSTM+Attention [1] on the S&P 500 data.

Models	S&P 500		
	RMSE	MAE	MAPE
WLSTM+Attention [1]	0.2337	0.1935	-
DAD-LSTM-I	**0.0445 ± 0.0020**	**0.0341 ± 0.0015**	**0.0139 ± 0.0006**
DAD-LSTM-II	0.0469 ± 0.0028	0.0370 ± 0.0023	0.0149 ± 0.0009
DAD-LSTM-III	0.0509 ± 0.0024	0.0406 ± 0.0022	0.0164 ± 0.0009

Table 7. Computational time results (unit in second, round up to the nearest positive integer) over all the datasets with three methods: DAD-LSTM-I, DAD-LSTM-II and DAD-LSTM-III.

Datasets	DARNN	DAD-LSTM-I	DAD-LSTM-II	DAD-LSTM-III
NASDAQ100(1)	2025	2256	2234	2469
NASDAQ100(2)	2343	2606	2573	2838
S&P500	248	285	281	308
HSI	217	248	252	269
DJIA	249	287	278	308

The evaluation metrics in Table 2-6 show that DAD-LSTM models outperform all of other methods in three common metrics of time series forecasting. Numerical results confirm the fact that the DAD-LSTM architecture with DLSTM in the encoder yields the best accuracy. For visual comparison, we show the prediction result of DAD-LSTM-I model over the four stock datasets in Figure 5. Although the codes have not been fully optimized, the computing time measurements in Table 7 shows that DAD-LSTM models can be applied to predict a large dataset such as NASDAQ 100 in a reasonable time. Moreover, the computational time of DAD-LSTM-I model increases approximately 1.13 times in comparison with DA-RNN, however the accuracy can be gained significantly.

4. Conclusion

In this paper, we proposed an efficient hybrid mechanism with LSTM neural network called DAD-LSTM to predict stock prices. This method uses DLSTM models to combine with the architecture of DA-RNN neural network. Based upon the DLSTM neural network in the encoder, the DAD-LSTM models can adaptively select the most relevant input features and capture the long-term temporal dependencies of a time series more appropriately and efficiently. The numerical results show that DAD-LSTM models outperform the DA-RNN and more recent approaches such as DSTP-RNN and WLSTM+Attention over four well-known stock datasets including NASDAQ 100, HSI, DJIA and S&P 500. In the future research, from the practical point of views we can analyze and apply DAD-LSTM models to other fields such as weather forecasting, energy consumption prediction and heart and brain signal analysis. From the theoretical point of views, a deep analysis of mathematical properties of DAD-LSTM models can be conducted following [26].

References

[1] Y. Qiu, B. Wang, and C. Zhou, "Forecasting stock prices with long-short term memory neural network based on attention mechanism," *PLoS ONE 15(1): e0227222. https://doi.org/10.1371/journal.pone.0227222*, 2020.

[2] P. Whittle, "Hypothesis testing in time series analysis," PhD thesis, 1951.

[3] D. Asteriou and S. G. Hall, "ARIMA models and the box-jenkins methodology," *Applied Econometrics*, vol. 2(2), pp. 265–286, 2011.

[4] Y. Qin, D. Song, H. Cheng, W. Cheng, G. Jiang, and G. W. Cottrell1, "A dual-stage attention-based recurrent neural network for time series prediction," *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, 2017.

[5] Y. Tao, L. Ma, W. Zhang, J. Liu, W. Liu, and Q. Du, "Hierarchical attention-based recurrent highway networks for time series prediction," 2018. arXiv:1806.00685.

[6] H. K. Cao, H. K. Cao, and B. T. Nguyen, "Delafo: An efficient portfolio optimization using deep neural networks," in *Advances in Knowledge Discovery and Data Mining* (H. W. Lauw, R. C.-W. Wong, A. Ntoulas, E.-P. Lim, S.-K. Ng, and S. J. Pan, eds.), pp. 623–635, Springer International Publishing, 2020.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9(8), p. 1735–1780, 1997.

[8] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by backpropagating errors," *Nature*, vol. 323(9), pp. 533–536, 1986.

[9] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78(10), pp. 1550–1560, 1990.

[10] J. L. Elman, “Distributed representations, simple recurrent networks, and grammatical structure,” *Machine learning*, vol. 7(2-3), pp. 195–225, 1991.

[11] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” 2014. arXiv:1412.3555.

[12] S. Chang, Y. Zhang, W. Han, M. Yu, X. Guo, W. Tan, X. Cui, M. Witbrock, M. Hasegawa-Johnson, and T. S. Huang, “Dilated recurrent neural networks,” *The Conference on Neural Information Processing Systems NIPS 2017*, 2017.

[13] Y. Liu, C. Gong, L. Yang, and Y. Chen, “DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction,” *Expert Systems With Applications (2019)*, 2019.

[14] “NASDAQ 100 dataset.” `https://cseweb.ucsd.edu/~yaq007/NASDAQ100_stock_data.html`.

[15] “Yahoo! finance. S&P 500 dataset.” `https://finance.yahoo.com/quote/%5EGSPC/history/`.

[16] “Yahoo! finance. HSI dataset.” `https://finance.yahoo.com/quote/%5EHSI/history/`.

[17] “Yahoo! finance. DJIA dataset.” `https://finance.yahoo.com/quote/%5EDJI/history/`.

[18] T. I. Poznyak and A. S. Poznyak, “Background on dynamic neural networks,” *Ozonation and Biodegradation in Environmental Engineering*, 2019.

[19] Y. Gao and M. J. Er, “Narmax time series model prediction: Feedforward and recurrent fuzzy neural network approaches,” *Fuzzy Sets and Systems*, vol. 150(2, p. 331–350, 2005.

[20] E. Diaconescu, “The use of NARX neural networks to predict chaotic time series,” *WSEA Transactions on Computer Research*, vol. 3(3), 2008.

[21] Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” *IEEE Transactions on Neural Networks*, vol. 5(2), pp. 157–166, 1994.

[22] A. Sherstinsky, “Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network.” `https://www.sciencedirect.com/science/article/abs/pii/S0167278919305974`, 2020.

[23] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” *arXiv:1608.06993*, 2018.

[24] S. Sun, Z. Cao, H. Zhu, and J. Zhao, “A survey of optimization methods from a machine learning perspective,” 2019. arXiv:1906.06821.

[25] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *In International Conference on Learning Representations*, 2014.

[26] J. Miller and M. Hardt, “Stable recurrent models,” 2019. arXiv:1805.10369.