Received November 19, 2019, accepted December 10, 2019, date of publication December 16, 2019, date of current version December 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2959789

Mixture of Activation Functions With Extended Min-Max Normalization for Forex Market Prediction

LKHAGVADORJ MUNKHDALAI[1], (Student Member), TSENDSUREN MUNKHDALAI[2], KWANG HO PARK[1], (Student Member), HEON GYU LEE[3], MEIJING LI[4], AND KEUN HO RYU[5], (Member, IEEE)

[1]Database/Bioinformatics Laboratory, School of Electrical and Computer Engineering, Chungbuk National University, Cheongju 28644, South Korea
[2]Microsoft Research Montreal, Montreal, QC H3A 3H3, Canada
[3]Big Data Research Center, GAION, Seoul 06167, South Korea
[4]College of Information Engineering, Shanghai Maritime University, Shanghai 200136, China
[5]Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City 700000, Vietnam

Corresponding author: Keun Ho Ryu (khryu@tdtu.edu.vn)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning under Grant 2017R1A2B4010826 and Grant 2019K2A9A2A06020672, in part by the Private Intelligence Information Service Expansion under Grant C0511-18-1001, in part by the National IT Industry Promotion Agency (NIPA) in Republic of Korea, and in part by the National Natural Science Foundation of China under Grant 61702324 and Grant 61911540482 in People's Republic of China.

ABSTRACT An accurate exchange rate forecasting and its decision-making to buy or sell are critical issues in the Forex market. Short-term currency rate forecasting is a challenging task due to its inherent characteristics, which include high volatility, trend, noise, and market shocks. We propose a novel deep learning architecture consisting of an adaptive activation function selection mechanism to achieve higher predictive accuracy. The proposed architecture is composed of seven neural networks that have different activation functions as well as softmax layer and multiplication layer with a skip connection, which are used to generate the dynamic importance weights that decide which activation function is preferred. In addition, we introduce an extended Min-Max smoothing technique to further normalize financial time series that have non-stationary properties. In our experimental evaluation, the results showed that our proposed model not only outperforms deep neural network baselines but also other classic machine learning approaches. The extended Min-Max smoothing technique is step towards forecasting non-stationary financial time series with deep neural networks.

INDEX TERMS Neural networks, activation function, value at risk, min-max normalization, forex market.

I. INTRODUCTION

The Forex (FX) market is one of the most popular financial markets for trading currencies because it is the largest financial market in the world in terms of trading volume [1]. The main players in the FX market are central banks, commercial banks, investment firms, hedge firms and individuals who are also known as traders. Their goal of participation in the FX market comes from their financial activities and economic needs. Large participants mostly engage in the FX market to manage their portfolio and avoid the exchange-rate risk. Small participants usually play in the FX market to make a profit from the short-term currency rate changes. In other words, both participants aim to define the profitable trading strategy. In order to fulfil their goals, it is a priority to accurately forecast short-term exchange rate movements [2]. The basic approaches that solve this problem are classified into two categories: technical and fundamental analysis. The fundamental analysis concentrates on long-term FX market prediction based on the multivariate time series analysis using other economic and financial variables. The technical analysis, in contrast, considers short-term FX market prediction and it attempts to predict future exchange rate movements in the FX market by thoroughly examining past market data [3].

The associate editor coordinating the review of this manuscript and approving it for publication was Choon Ki Ahn.

In addition, statistical and econometric methods such as exponential smoothing-ETS, autoregressive integrated moving average – ARIMA have been used to predict financial time series [4]–[6]. The advantage of these models does not require data normalization and can deal with non-stationary time series by differencing transformations. But their prediction performances could be poor depending on financial time series characteristics [7]. Therefore, in recent years, many machine learning models have been commonly used in this field [8], [9]. For instance, support vector machine – SVM [10], random forest – RF [11], [12], artificial neural networks [13]–[20] as well as reinforcement learning has been used for portfolio management [21]–[23]. Among the machine learning algorithms, artificial neural networks are the most widely used as a prediction model for forecasting financial time series. For instance, many studies have proposed the hybrid artificial neural network models such as the combination of ARIMA and neural networks, genetic algorithms with neural networks, etc. [24]–[26]. Moreover, deep learning approaches have become increasingly popular in this field [17]–[20], [26]–[28]. Galeshchuk and Mukherjee Sumitra [7] used deep convolution neural networks to predict the direction of change in forex rates. Qin *et al.* [18] proposed an attention mechanism based deep neural network model to adaptively extract relevant input features at each time step. Although researchers only focused advanced neural network architectures and techniques, they do not pay much attention to the choice of activation function and time series normalization problem. Due to the non-linear activation functions and their characteristics in neural networks, it is essential that the data must be normalized. Since we are not able to know future values of data (test set), data normalization and the prediction model have to be performed based on the past or known values of data (training set). The standard Min-Max normalization technique can be used when the maximum and minimum values of time series are knowable a priori or do not change over time. Unfortunately, in the real world, most financial time series is non-stationary heteroscedastic, therefore the standard Min-Max normalization technique is inappropriate [29]. For instance, Figure 1 shows normalized and actual currency rate British pound (GBP) to US Dollar (USD) from 2000 to 2019. We divided the dataset into training and test sets and normalized them using minimum and maximum values of training set based on standard Min-Max normalization. As it is seen, the minimum value of test set is less than the minimum value of training set. Consequently, after the normalization, some values of test set are less than 0. Assuming that if the outputs of many non-linear activation functions are always greater than or equal to 0, it will make the test set unpredictable. Figure 2 shows the prediction result for test set using ReLU function as an output of neural network. Although one of solutions to this problem is to use a linear function as an output of neural network, it may not be efficient for non-linear time series analysis.

In the literature, the most widely used normalization techniques are logarithmic scale [30]–[33], Min-Max

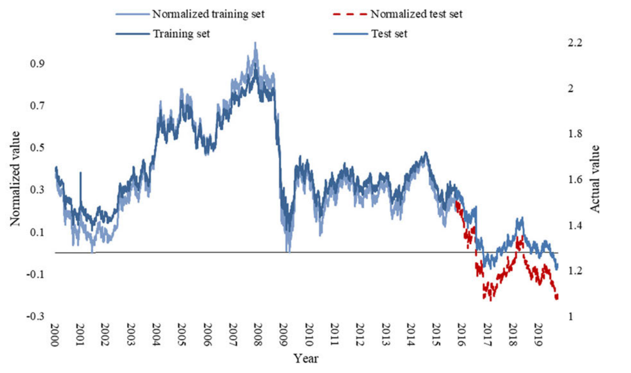


FIGURE 1. Normalized and actual exchange rate British pound (GBP) to US Dollar (USD) from 2000 to 2019. GBP/USD is partitioned into training and test sets. Minimum and maximum values of training set are used to normalize test set (red dashed line).

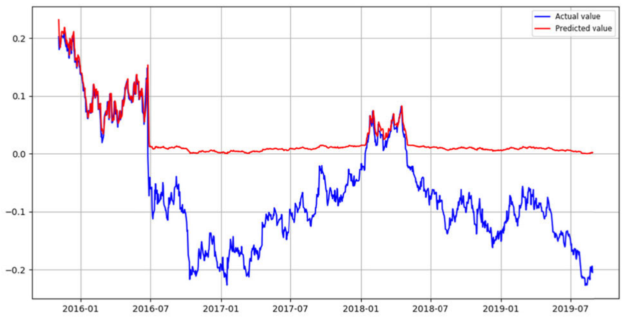


FIGURE 2. The example of ReLU activation function used as the output of neural network. The predicted and actual values for test set.

normalization (scaling of data between ranges of 0 or -1 and 1) [7], [27], [34] and the z-score standardization [35]. When using deep neural network as a prediction model, the logarithmic scale technique is not suitable because normalization can stable each value to reduce the vanishing or exploding gradient problem. However, the z-score technique is useful when the minimum and maximum values of time series are unknown. But this technique can be applied to stationary time series, which is not always possible.

Only a limited number of studies have considered non-stationary financial time series normalization. Ogasawara *et al.* [36] proposed an adaptive normalization method consisting of three stages. The first stage transforms the non-stationary time series into a stationary sequence, the second stage removes outliers and the third stages normalize data by using itself. Zhang *et al.* [37] and Guresen *et al.* [9] substituted $+/-$ 0.9 instead of $+/-$ 1 for Min-Max normalization. But these methods are not commonly used to normalize financial time series data.

Unlike these normalization techniques, we extend Min-Max normalization, which is developed by Value at Risk (VaR) technique, for constructing neural network model with non-linear activation functions. Our extended Min-Max normalization has a beneficial advantage that can measure the potential Forex risk during data normalization. In other words, the calculated minimum and maximum bounds by using VaR can express the highest and lowest possible value of exchange rate, respectively.

After data normalization by using extended Min-Max normalization technique, any activation function can be used for neural network modelling. On the other hands, the choice of activation function in neural network has a significant impact on the training dynamics and task performance [38]. But the procedure to fund suitable activation function is an extremely costly job. In practice, the searching algorithms have mostly been used to perform this procedure. Therefore, in this paper, we propose a novel end-to-end deep learning framework for efficiently finding suitable activation function to the task.

Our proposed deep learning architecture consists of two components: the first component contains seven neural networks consisting of different activation functions such as Rectified Linear Unit (ReLU), linear, sigmoid and Hyperbolic Tangent (tanh), sine, cosine [39] and swish [38], and the second component is the softmax layer that produces the probabilities or importance weights for the previous seven networks. Then the multiplication layer with a skip connection is used to compute an element-wise multiplication of outputs of seven neural networks and the importance weights to prepare final input for the output layer to predict target value. We jointly optimize the parameters of these components in an end-to-end framework.

In the experimental part, Multi-Layer Perceptron (MLP), an Elman Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU) and Long-short Term Memory (LSTM) are trained based on daily FX market dataset. The root mean square error (RMSE) and mean absolute error (MAE) are used to evaluate model performances. The random forest, Ada Boost, XGBoost and Support Vector Machine algorithms are chosen as a machine learning baseline and standard architecture of MLP, RNN, GRU and LSTM are compared to our proposed architecture. The results show that our proposed architecture outperforms baseline machine learning models and standard neural network architectures on over six foreign exchange rates. In addition, it is proven that our extended normalization technique and its similarity to standard Min-Max normalization make it easy to predict financial time series using non-linear activation functions.

This paper is organized as follows. Section 2 describes details of the VaR technique and an extended Min-Max normalization algorithm. Section 3 presents our proposed deep learning framework. Then Section 4 presents data partitioning, backtesting for extended Min-Max normalization technique and the prediction results. Finally, Section 5 summarizes the general findings from this study and discusses possible future research areas.

II. EXTENDED MIN-MAX NORMALIZATION

A. VALUE AT RISK

Value at Risk (VaR) has emerged as a widely used statistical tool for risk management of financial institutions [40]. This method is most commonly used by financial institutions to determine the occurrence probability of thee potential losses in their financial portfolios. VaR technique attempts to estimate the distribution of financial return series and compute VaR from the estimated distribution to calculate the potential loss [41]. We use this basic idea of VaR technique to calculate the future minimum and maximum bounds for financial time series. The JP Morgan-RiskMetrics' standard exponentially weighted moving average (EWMA) technique, which is an extensively used measure of market risk for financial portfolios, is adopted in this paper [42].

Given a dataset $\{y_i\}_{i=1}^{t+k}$, $i = 1, \ldots, t + k$, where y denotes a single currency rate, t and k are the length of training and test sets, respectively. Then the return of the currency rate is defined as:

$$r_t = \log\left(y_t / y_{t-1}\right) \tag{1}$$

Using distribution of return series, r_t, the σ_{t+1}^2 volatility of currency rate on period $t+1$ is estimated by Eq.(2) at the end of period t:

$$\sigma_{t+1}^2 = (1 - \lambda)\, r_t^2 + \lambda \sigma_t^2 \tag{2}$$

where σ_{t+1}^2 is the square of volatility of currency rate as defined by, $\sigma_t^2 = \sum_{i=1}^{t} (r_i - \bar{r})^2 / t$, λ is a constant number between 0 and 1, the RiskMetrics database produced by JP Morgan uses the EWMA with $\lambda = 0.94$ for daily data [42]. Finally, the VaR is modelled as Eq. (3) using the JP Morgan RiskMetrics' standard-EWMA that assumes conditional normality, with volatility σ_{t+1}:

$$VaR_{t+1} = \sigma_{t+1} \Phi^{-1}(\alpha) \tag{3}$$

where α is the significance level of VaR, $\Phi^{-1}(*)$ is an inverse function of cumulative standard normal distribution. In addition, we can easily calculate the VaR for a particular time period using a time variable and the VaR of one time period. This can be done by relying on a classic idea in finance that the standard deviation of asset returns tends to increase with the square root of time. Accordingly, the VaR of one-time period is multiplied by the square root of the required specific time period (the length of test set):

$$VaR_{t+k} = VaR_{t+1} \sqrt{k} \tag{4}$$

where k is the specified time period as well as it can be the length of test set in this paper.

Since we estimated future VaR using the distribution of return series, it is easy to calculate the future minimum and maximum bounds of financial asset price:

$$\begin{aligned} Max_{t+k} &= y_t \left(1 + VaR_{t+k}\right) \\ Min_{t+k} &= y_t \left(1 - VaR_{t+k}\right) \end{aligned} \tag{5}$$

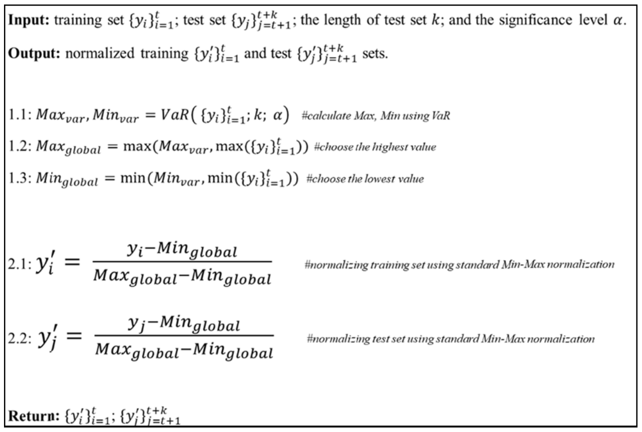
B. EXTENDED MIN-MAX NORMALIZATION

One of the primary objectives of this study is to extend Min-Max normalization for financial time series data. Because standard Min-Max normalization technique makes an impossible assumption that the maximum and minimum values of financial time series do not change over time. Therefore, we extend it based on the VaR technique as described above. This normalization technique is implemented in two main steps according to the pseudocode in Figure 3.

Step 1: Following Eq. (5), we calculate the future maximum and minimum bounds using training set and the length of test set. Then we compare the minimum and maximum values of training set to the bounds calculated by the VaR technique and select the lowest and highest values. The output of the first step can be the global maximum and minimum values.

Step 2: Since the global maximum and minimum values are determined, training and test sets can be normalized by using standard Min-Max normalization. Depending on the given activation function in neural networks, financial time series data can be treated by one of the following two types of normalization:

Scaling between 0 and 1	Scaling between -1 and 1
$\hat{x} = \frac{x - x_{\min\_global}}{x_{\max\_global} - x_{\min\_global}}$	$\hat{x} = 2 \bullet \frac{x - x_{\min\_global}}{x_{\max\_global} - x_{\min\_global}} - 1$



```
Input: training set $\{y_i\}_{i=1}^{t}$; test set $\{y_j\}_{j=t+1}^{t+k}$; the length of test set $k$; and the significance level $\alpha$.
Output: normalized training $\{y_i'\}_{i=1}^{t}$ and test $\{y_j'\}_{j=t+1}^{t+k}$ sets.

1.1: $Max_{var}, Min_{var} = VaR(\{y_i\}_{i=1}^{t}; k; \alpha)$ #calculate Max, Min using VaR
1.2: $Max_{global} = \max(Max_{var}, \max(\{y_i\}_{i=1}^{t}))$ #choose the highest value
1.3: $Min_{global} = \min(Min_{var}, \min(\{y_i\}_{i=1}^{t}))$ #choose the lowest value

2.1: $y_i' = \frac{y_i - Min_{global}}{Max_{global} - Min_{global}}$ #normalizing training set using standard Min-Max normalization

2.2: $y_j' = \frac{y_j - Min_{global}}{Max_{global} - Min_{global}}$ #normalizing test set using standard Min-Max normalization

Return: $\{y_i'\}_{i=1}^{t}; \{y_j'\}_{j=t+1}^{t+k}$
```

FIGURE 3. The pseudocode for extended min-max normalization for time series.

By using the extended Min-Max normalization algorithm described above, the entire dataset can be normalized based on the training set only.

III. THE PROPOSED DEEP LEARNING FRAMEWORK

The overall architectural diagram of our proposed deep learning model for forecasting short-term FX market is shown in Figure 4. The proposed framework consists of seven neural networks that have different activation function. Each neural network takes inputs separately and then all neural networks will be parallelly trained to produce own output. The concatenation layer concatenates their outputs to feed into the softmax layer for producing the probabilities or importance weights for each neural network. Finally, the multiplication layer prepares the weighted inputs using skip connection for output layer to predict the target output.

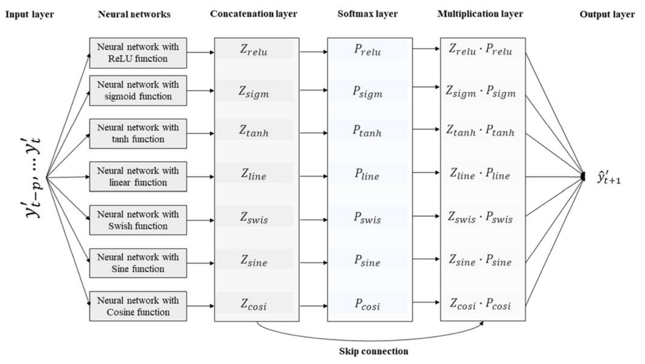

Input layer
Neural networks
Concatenation layer
Softmax layer
Multiplication layer
Output layer
$y'_{t-p} \cdots y'_t$
Neural network with ReLU function
Neural network with sigmoid function
Neural network with tanh function
Neural network with linear function
Neural network with Swish function
Neural network with Sine function
Neural network with Cosine function
Z_{relu}
Z_{sigm}
Z_{tanh}
Z_{line}
Z_{swis}
Z_{sine}
Z_{cosi}
P_{relu}
P_{sigm}
P_{tanh}
P_{line}
P_{swis}
P_{sine}
P_{cosi}
$Z_{relu} \cdot P_{relu}$
$Z_{sigm} \cdot P_{sigm}$
$Z_{tanh} \cdot P_{tanh}$
$Z_{line} \cdot P_{line}$
$Z_{swis} \cdot P_{swis}$
$Z_{sine} \cdot P_{sine}$
$Z_{cosi} \cdot P_{cosi}$
$\hat{y}'_{t+1}$
Skip connection

FIGURE 4. The proposed deep learning framework for forex market prediction, where $y'_{t-p}, \ldots, y'_t$ denote the normalized lagged values, p is the lag or time steps, y'_t denotes the predicted normalized value at $t - th$ period, Z denotes the output of each neural network, P is the output of softmax layer, which is probability or importance weight for the output of neural networks.

A. NEURAL NETWORKS

We apply MLP, RNN, GRU and LSTM architectures as a basic neural network:

- MLP is the most commonly used type of feedforward artificial neural network that has been developed similar with human brain function, the basic concept of a single perceptron was introduced by Rosenblatt [43]. This network consists of three layers with completely different roles called input, hidden and output layers. Each layer contains a given number of neurons with the activation function and neurons in neighbour layers are linked by weight parameters.
- A simple RNN was proposed by Elman [44] RNN networks especially useful for processing sequential, time series data as well as spatial patterns. This RNN network contains a feedback loop, which is the output from time t-1 is fed back to the network to affect the outcome of time t, and so forth for each time step, called the recurrent layer. Only RNNs have feedback connection that gives it the ability to memorize and make use of past information is the major difference between RNNs and MLP.
- GRU was proposed by Cho *et al.* [45] to make each recurrent unit to be able to adaptively capture dependencies of different time scales. GRU has only two gates: a reset gate and update gate. Those gates harmonize the flow of information inside the unit without having a separate cells. The update and reset mechanisms help the GRU to capture long-term dependencies and to use the model capacity efficiently by allowing it to reset whenever the detected information is not necessary anymore, respectively.
- LSTM network is an extension of simple RNNs. It was proposed by Hochreiter and Schmidhuber [46] as a solution to the vanishing gradient problem. LSTM helps to solve long-term dependencies by extending their memory cells and utilizing a gating mechanism to control information flow. The memory cell consists of three gates – input, forget and output gate. Those gates decide whether or not to add new input in (input gate), delete the unnecessary information

(forget gate) or to add it impact the output at the current time step (output gate).

B. ACTIVATION FUNCTIONS

We use seven activation functions for constructing our proposed framework, which are ReLU, sigmoid, Tanh, linear, Swish, Sine and Cosine functions. The linear, ReLU, sigmoid and tanh functions are commonly used activation functions in neural networks [47], [48]. However these functions are non- periodic and monotonic. Sine and Cosine activation functions, in contrast, are periodic and non-monotonic, therefore these periodic functions may learn better than non-periodic functions on seasonal time series data [39]. In addition, we consider Swish activation function, which is a new activation function explored by researchers at Google This new function has outperformed ReLU and other activation functions [38] Table 1 summarized these activation functions. We apply these activation functions to four deep learning architectures, which are the aforementioned architectures, in our proposed framework. In our proposed deep learning architecture, since the softmax layer automatically determines which activation function to select, our architecture could improve the performance when it has as many activation functions as possible.

TABLE 1. The summary of the selected activation functions with their respective gradients.

№	Name	Functions	Gradients
1	Linear	$f(x)=x$	$f'(x)=1$
2	ReLU	$f(x)=\max(0,x)$	$f'(x)=\begin{cases}0 & \text{if } x\le 0\\ 1 & \text{if } x>0\end{cases}$
3	Sigmoid	$f(x)=\frac{1}{1+e^{-x}}$	$f'(x)=\frac{1}{1+e^{-x}}\left(1-\frac{1}{1+e^{-x}}\right)$
4	Tanh	$f(x)=\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	$f'(x)=1-\left(\frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\right)^2$
5	Sine	$f(x)=\sin(x)$	$f'(x)=\cos(x)$
6	Cosine	$f(x)=\cos(x)$	$f'(x)=-\sin(x)$
7	Swish	$f(x)=x\bullet\frac{1}{1+e^{-x}}$	$f'(x)=x\bullet\frac{1}{1+e^{-x}}+\frac{1}{1+e^{-x}}\left(1-x\bullet\frac{1}{1+e^{-x}}\right)$

C. THE CONCATENATION LAYER

Let $Y_t' = (y_1',y_2',\ldots y_t')$ denote the normalized currency rate. Each neural network returns a one-dimensional output Z and our basic neural network is a $F_I(*)$, where I denotes the activation function. The inputs of concatenation layer are as follows:

$$\begin{aligned}
Z_{\text{linear}} &= F_{\text{linear}}(Y_t'; \theta_{\text{linear}})\\
Z_{\text{ReLU}} &= F_{\text{ReLU}}(Y_t'; \theta_{\text{ReLU}})\\
Z_{\text{sigmoid}} &= F_{\text{sigmoid}}(Y_t'; \theta_{sigmoid})\\
Z_{\text{tanh}} &= F_{\text{tanh}}(Y_t'; \theta_{\text{tanh}})\\
Z_{\text{sine}} &= F_{\text{sine}}(Y_t'; \theta_{\text{sine}})\\
Z_{\text{cosine}} &= F_{\text{cosine}}(Y_t'; \theta_{\text{cosine}})
\end{aligned}$$

TABLE 2. The searching space of hyper-parameters.

Models	Parameters	Search Space
Random forest	n_estimators	[500, 3000]
	max_depth	[2, 8]
	min_samples_split	[1, 8]
	min_samples_leaf	[1, 8]
Ada Boost	n_estimators	[500, 3000]
	learning_rate	[0.1, 1]
	algorithm	{'SAMME.R', 'SAMME'}
XGBoost	n_estimators	[10, 100]
	min_child_weight	[1, 10]
	gamma	[0, 1]
	subsample	[0.5, 1]
	colsample_bytree	[0.5, 1]
	max_depth	[2, 8]
	learning_rate	[0.01, 0.5]
SVM	kernel function	{'linear','rbf'}
	gamma	[0.001, 1000]
	cost	[0.001, 1000]
	max_iter	[5000, 10000]

$$Z_{swish} = F_{swish}(Y_t'; \theta_{\text{swish}}) \tag{6}$$

where θs are the weight parameters of neural networks. The concatenation layer takes the outputs of neural networks and concatenates them to return a single array.

$$Z=[Z_{\text{linear}}, Z_{ReLU}, Z_{\text{sigmoid}}, Z_{\text{tanh}}, Z_{\text{sine}}, Z_{\text{cosine}}, Z_{\text{swish}}] \tag{7}$$

D. THE SOFTMAX LAYER

The concatenated output feeds the softmax layer to determine the probabilities or importance weights for each neural network. These dynamic probabilities or importance weights decides which neural network is preferred [49]

$$P_i = \frac{e^{\omega_i^\top Z_i}}{\sum_{i=1}^{7} e^{\omega_i^\top Z_i}} \tag{8}$$

where P_i is $i-th$ network's probability or importance weight, ω_i is $i-th$ network's weight parameters.

Note that the probability or importance weights can be seen as fast-weights, which has successfully been used in the meta-learning context for rapid adaptation [50]–[53].

E. THE MULTIPLICATION LAYER

The multiplication layer takes an element-wise multiplication of the concatenated outputs of neural networks and the importance weights to prepare the input for the output layer using a skip connection [54]

$$X = Z \odot P \tag{9}$$

where X denotes the weighted input for the output layer and $\odot$ denotes an element-wise multiplication.

F. THE OUTPUT LAYER

The output layer consists of a fully connected (FC) layer with linear activation function. It takes the weighted inputs come

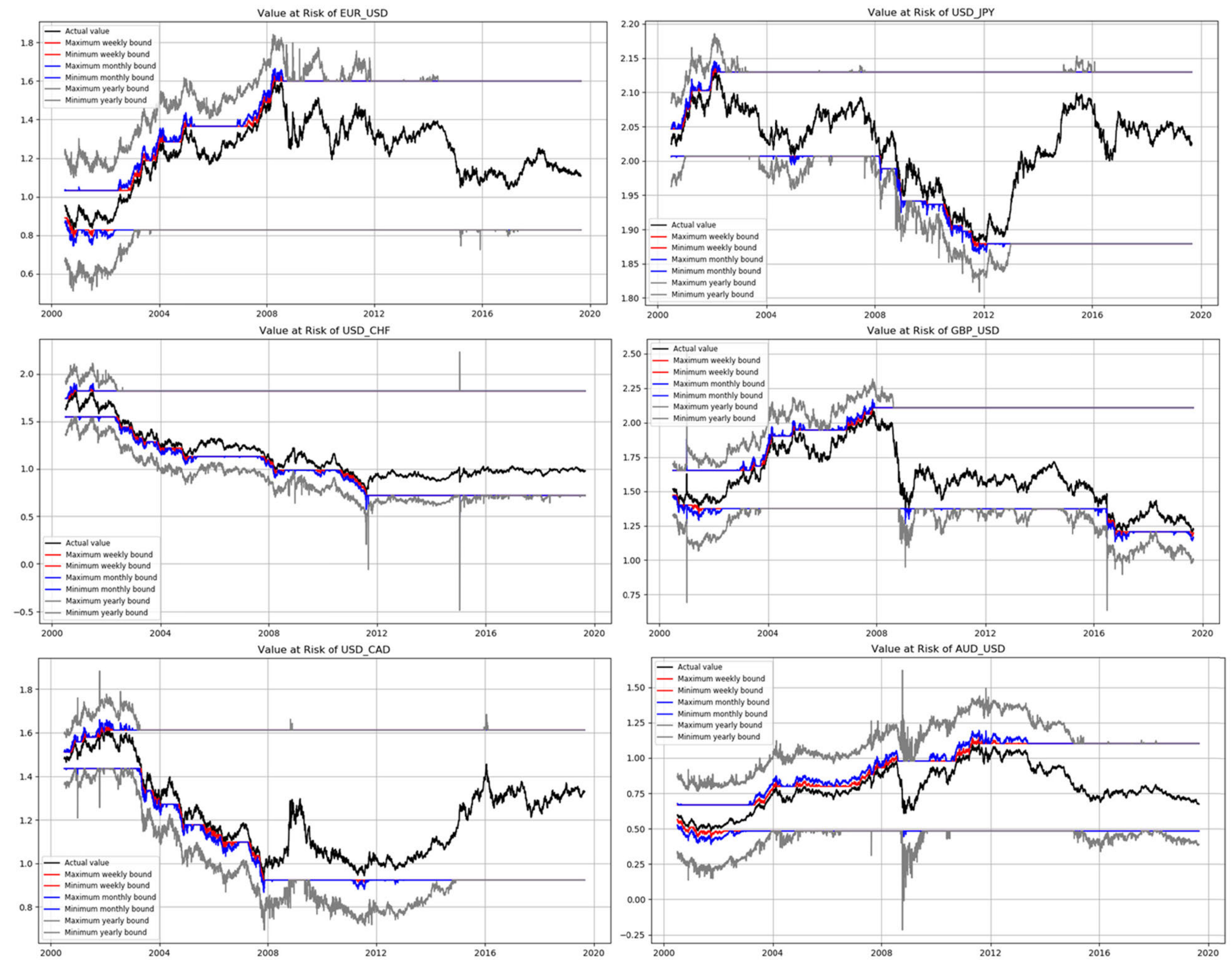


FIGURE 5. **The calculated maximum and minimum bounds by extended Min-Max normalization for weekly, monthly and yearly periods.**

from the multiplication layer to predict the target variable.

$$\hat{y}'_{t+1} = FC(X; \nu) \qquad (10)$$

where ν denotes the weight parameters of output layer.

G. END-TO-END TRAINING

As previously explained, our deep learning framework contains three types of weight parameters: the neural networks' parameters -θ, the softmax layer's parameters -ω, and the output layer's parameters -ν. The prediction target value is as follows:

$$\hat{y}'_{t+1} = FC\left(\left(Z(Y'_t; \theta) \odot P\left(Z(Y'_t; \theta); \omega\right)\right); \nu\right) \qquad (11)$$

The mean squared error (MSE), which is a measurement of the average of the squares of the difference between the predicted and actual values is used as a loss function.

$$J(\theta, \omega, \upsilon) = \frac{1}{t}\sum_{i=1}^{t}\left(y'_i - FC\left(\left(Z(Y'_{i-1}; \theta)\right.\right.\right.$$
$$\left.\left.\left.\odot P\left(Z(Y'_{i-1}; \theta); \omega\right)\right); \nu\right)\right)^2 \qquad (12)$$

where y'_i is $i-th$ actual value.

We can now jointly optimize our deep learning framework that consists of the seven neural networks based on the above loss function.

IV. EXPERIMENTAL RESULTS

In this section, the proposed framework is implemented using seven activation functions and four neural network architectures on over six foreign exchange rates. We used daily forex prices such as EUR/USD, USD/JPY, USD/CHF, GBP/USD, USD/CAD and AUD/USD. Data partitioning and experimental set-up are presented in Section 4.1. Thereafter, the backtesting of extended Min-Max normalization technique is displayed in Section 4.2. Finally, the prediction performances are demonstrated in Section 4.3 and the learning combinations of activation functions are explained in Section 4.4.

A. DATA PARTITIONING AND EXPERIMENTAL SET-UP

We now describe the data partitioning and model experimental set-up. The dataset used in this study is retrieved from www.global-view.com. To evaluate the models built by containing different activation functions and architectures,

TABLE 3. RMSE of our proposed model and the baseline models on six currency rates.

Models	Activation function	EUR/USD	USD/JPY	USD/CHF	GBP/USD	USD/CAD	AUD/USD
RNN	Tanh	0.0127	1.0838	0.0133	0.0378	0.0138	0.0121
	Sine	0.0119	1.0127	0.0236	0.0333	0.0141	0.0100
	Cosine	0.0123	1.6536	0.0125	0.0330	0.0136	0.0179
	ReLU	0.0062	0.8931	0.0075	0.0155	0.0069	0.0054
	Linear	0.0061	0.6548	0.0060	0.0090	0.0062	0.0047
	Swish	0.0059	0.6435	0.0062	0.0099	0.0085	0.0054
	Sigmoid	0.0079	6.0677	0.0097	0.0188	0.0075	0.0090
	Ours	**0.0057**	**0.5993**	**0.0059**	0.0098	0.0062	**0.0045**
GRU	Tanh	0.0283	1.7696	0.0243	0.0375	0.0177	0.0235
	Sine	0.0254	1.5267	0.0264	0.0390	0.0257	0.0158
	Cosine	0.0129	2.6072	0.0168	0.0548	0.0133	0.0187
	ReLU	0.0073	1.0381	0.0071	0.1957	0.0116	0.0050
	Linear	0.0058	0.6054	0.0061	0.0104	0.0066	0.0052
	Swish	0.0060	0.8547	0.0063	0.0095	0.0061	0.0052
	Sigmoid	0.0086	0.7807	0.0118	0.0181	0.0110	0.0087
	Ours	0.0059	0.6871	0.0060	**0.0083**	**0.0060**	0.0082
LSTM	Tanh	0.0121	1.0786	0.0147	0.0336	0.0149	0.0135
	Sine	0.0128	0.9842	0.0242	0.0305	0.0127	0.0094
	Cosine	0.0118	1.0697	0.0214	0.0321	0.0130	0.0098
	ReLU	0.0069	1.6890	0.0079	0.0155	0.0074	0.0058
	Linear	0.0062	0.8575	0.0075	0.0105	0.0072	0.0053
	Swish	0.0061	0.6890	0.0072	0.0088	0.0081	0.0069
	Sigmoid	0.0071	0.9775	0.0162	0.0171	0.0075	0.0067
	Ours	0.0062	0.6757	0.0087	0.0092	0.0078	0.0055
MLP	Tanh	0.0150	1.2248	0.0138	0.0372	0.0211	0.0129
	Sine	0.0156	1.0796	0.0122	0.0300	0.0172	0.0191
	Cosine	0.0137	1.7440	0.0167	0.0984	0.0144	0.0153
	ReLU	0.0064	0.9711	0.0081	0.0217	0.0066	0.0048
	Linear	0.0059	1.4826	0.0060	0.0098	0.0064	0.0047
	Swish	0.0058	0.9360	0.0061	0.0097	0.0070	0.0054
	Sigmoid	0.0062	1.1372	0.0180	0.0181	0.0078	0.0080
	Ours	0.0061	0.7722	0.0082	0.0088	0.0064	0.0053
Random Forest		0.0068	0.6781	0.0081	0.0244	0.0075	0.0056
AdaBoost		0.0076	0.8198	0.0136	0.0238	0.0080	0.0082
XGBoost		0.0076	0.6554	0.0071	0.0241	0.0085	0.0058
SVM		0.0332	0.6387	0.0394	0.0329	0.0125	0.0200

the datasets are partitioned into three parts; i.e., training (80%), validation (20%) and test (last 365 days for one fold) sets and 5-fold time series cross-validation method is used.

In neural networks, the hyper-parameters: learning rate, batch size, and epoch number must be pre-defined before training the model. We set the default learning rate to 0.001 using Adam optimizer [55], and use a mini-batch with 64 instances at each iteration. We set the number of maximum epoch to 3000 for the first fold only, and then we use the first pre-trained model for the next folds with 300 epochs. An Early Stopping algorithm is employed for finding the optimal epoch number based on given other hyper-parameters.

Regarding neural network architectures, the number of input layer neurons is equal to 5 (output of neural network depends on the prices of last 5 days) and one hidden layer with 16 neurons builds the neural networks.

All experiments for this study are performed using Python programming language, 3.5 version, on a PC with 3.4 GHz, Intel CORE i7, and 32 GB RAM, using Microsoft Windows 10 operating system.

We also compare our proposed model to the baseline machine learning algorithms, which are the most widely used models in this field such as Random Forest, AdaBoost, XGBoost and Support vector machine. The hyper-parameters of these baseline models are optimized by random search with 5 cross-validation methods over parameter settings as shown in Table 2.

B. BACKTESTING RESULT OF EXTENDED MIN-MAX NORMALIZATION

Backtesting is a technique for simulating VaR model on the historical data to measure its accuracy and effectiveness. Financially, the backtesting in VaR is used to compare the predicted losses by using the VaR technique to actual losses realized at the end of the specified time period. For extended Min-Max normalization, we evaluate whether the minimum and maximum values calculated by using VaR technique exceed actual values or not. When the confidence level (α) is equal to 0.01, the specified time periods are equal to 7, 30 and 365 days, no exceptions occurred. Figure 5 compares actual values to bounds for different time periods.

C. PREDICTION RESULTS

To prove the effectiveness and robustness of the proposed deep learning framework, we apply it to six currency rates as described above. We compare and evaluate the performance

TABLE 4. MAE of our proposed model and the baseline models on six currency rates.

Models	Activation function	EUR/USD	USD/JPY	USD/CHF	GBP/USD	USD/CAD	AUD/USD
	Tanh	0.0097	0.8396	0.0090	0.0258	0.0106	0.0095
	Sine	0.0090	0.7438	0.0203	0.0208	0.0110	0.0080
	Cosine	0.0095	1.3976	0.0082	0.0210	0.0106	0.0142
RNN	ReLU	0.0047	0.7115	0.0055	0.0103	0.0052	0.0042
	Linear	0.0046	0.4965	0.0038	0.0068	0.0047	0.0036
	Swish	0.0044	0.4925	0.0042	0.0074	0.0068	0.0043
	Sigmoid	0.0060	5.4000	0.0081	0.0126	0.0060	0.0079
	Ours	**0.0043**	0.4479	**0.0037**	0.0083	0.0047	**0.0034**
	Tanh	0.0248	1.4848	0.0208	0.0243	0.0146	0.0212
	Sine	0.0224	1.2658	0.0231	0.0273	0.0223	0.0136
	Cosine	0.0098	2.0191	0.0130	0.0396	0.0106	0.0162
GRU	ReLU	0.0057	0.8538	0.0048	0.1858	0.0095	0.0040
	Linear	0.0044	**0.4475**	0.0041	0.0083	0.0052	0.0041
	Swish	0.0046	0.6951	0.0042	0.0072	0.0047	0.0041
	Sigmoid	0.0071	0.6052	0.0100	0.0119	0.0096	0.0073
	Ours	0.0044	0.5327	0.0039	**0.0062**	**0.0046**	0.0071
	Tanh	0.0092	0.8205	0.0109	0.0219	0.0119	0.0110
	Sine	0.0096	0.7275	0.0206	0.0194	0.0099	0.0074
	Cosine	0.0088	0.8194	0.0166	0.0211	0.0101	0.0077
LSTM	ReLU	0.0053	1.4920	0.0054	0.0101	0.0058	0.0046
	Linear	0.0047	0.6666	0.0050	0.0082	0.0056	0.0042
	Swish	0.0045	0.5243	0.0049	0.0065	0.0063	0.0054
	Sigmoid	0.0056	0.8295	0.0146	0.0112	0.0058	0.0055
	Ours	0.0046	0.5200	0.0060	0.0069	0.0061	0.0044
	Tanh	0.0120	0.9754	0.0090	0.0246	0.0176	0.0100
	Sine	0.0120	0.8220	0.0076	0.0195	0.0132	0.0166
	Cosine	0.0102	1.3631	0.0130	0.0781	0.0114	0.0123
MLP	ReLU	0.0049	0.7499	0.0058	0.0150	0.0052	0.0038
	Linear	0.0044	1.2326	0.0039	0.0077	0.0050	0.0036
	Swish	0.0043	0.7351	0.0039	0.0073	0.0055	0.0042
	Sigmoid	0.0046	0.9013	0.0156	0.0120	0.0063	0.0069
	Ours	0.0047	0.6114	0.0059	0.0066	0.0049	0.0041
Random Forest		0.0053	0.5209	0.0061	0.0156	0.0059	0.0044
AdaBoost		0.0059	0.6440	0.0103	0.0158	0.0063	0.0066
XGBoost		0.0059	0.4958	0.0048	0.0156	0.0064	0.0045
SVM		0.0304	0.4718	0.0376	0.0294	0.0099	0.0176

of our model and the baseline models using two criteria: RMSE and MAE.

Table 3 and Table 4 show the prediction performances of our proposed deep learning model and the baseline models on six currency rates. The prediction results show that our proposed model with RNN architecture outperforms other neural network models including machine learning baselines on four currency rates in terms of RMSE. For other two currency rates, our proposed deep learning model with GRU architecture achieves the best performance in terms of both RMSE and MAE. The GRU model with a linear activation function achieves better performance than others in terms of MAE for USD/JPY only.

We observe that the baseline machine learning models underperform our proposed deep learning model, but they achieve the comparable performance with standard neural network architectures. Among machine learning baselines, while random forest regression performs the best on the EUR/USD, GBP/USD, USD/CAD and AUD/USD, SVM and XGBoost perform the best on USD/JPY and USD/CHF, respectively.

From Figure 6, Linear and Swish functions are learned better than other activation functions for the recurrent neural network architectures. In contrast, concerning MLP network, ReLU and Swish functions achieves better performances than others. However, our proposed deep learning framework works well for most neural network architectures.

From Table 3 - 4 and Figure 6, a conclusion can be drawn that our proposed deep learning model achieved significantly better performances than other baselines in most of the currency rates.

D. MIXTURE OF ACTIVATION FUCNTIONS

Our proposed deep learning framework builds a prediction model based on the combinations of activation functions. This section shows how the activation functions learn from data during the prediction. The softmax layer of our proposed deep learning model gives the probabilities or importance weights for the neural networks consisting of different activation function. Figure 7 shows the dynamic importance weights for seven neural networks in our proposed deep learning model. We can observe that the importance weights can change depending on time series characteristics. For EUR/USD, the neural networks, which are consisting of Tanh and Cosine activation functions, are more important than other neural

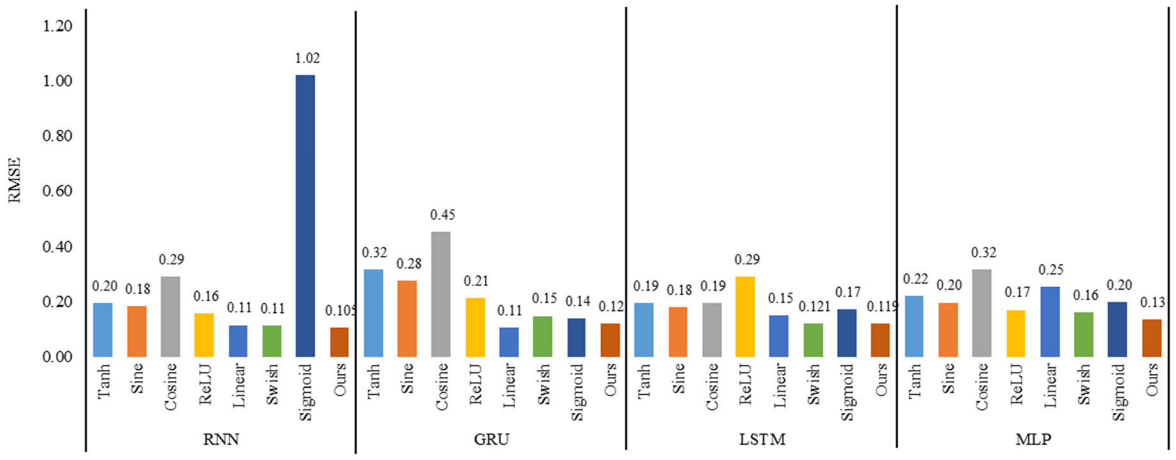


FIGURE 6. The comparison between activation functions and our model for four neural network architectures. The y-axis shows the averaged RMSE for six currency rates.

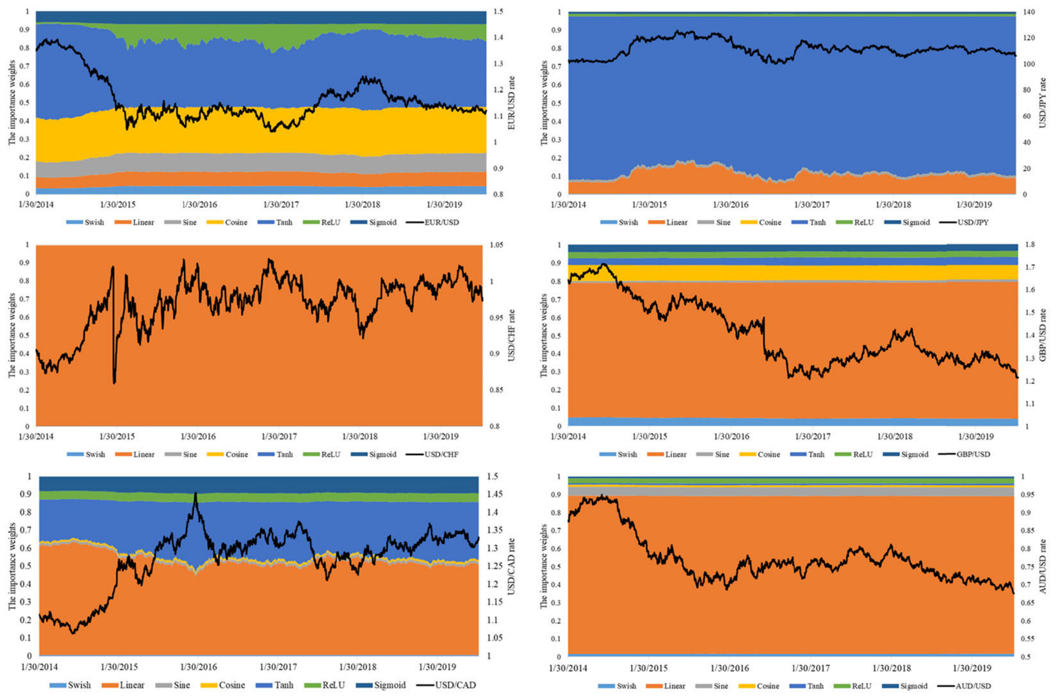


FIGURE 7. The dynamic importance weights for the activation functions on test set.

networks. In contrast, the neural network that has a linear activation function dominates other neural networks for the USD/CHF. Likewise, we can make detailed descriptions as to the importance of the neural networks for other currency rates.

In addition, in order to establish why a certain activation function is more suitable for a particular time series, we measure strength of trend and seasonality for each currency rate [56]. Figure 8 shows the average percentage of activation functions and the strength of trend and seasonality for each currency rate. We now can see that the importance of linear activation function is higher than others for a particular time series, which has high strength of trend and low strength of seasonality. On the contrary, as the strength of seasonality increases, the importance of other non-linear functions boosts. For a particular time series such as EUR/USD, its strength of trend and seasonality is not high. In this case, the importance of activation functions is close to each other.

In the end, the standard practice in training deep neural network treats the choice of an activation function as a hyper-parameter. Therefore, once the model is trained, the activation function remains fixed. This can be non-optimal, especially if the data has a distributional shift and/or a non-stationary property. Allowing model to select and mix a set of activation functions on the fly even during test time addresses this issue

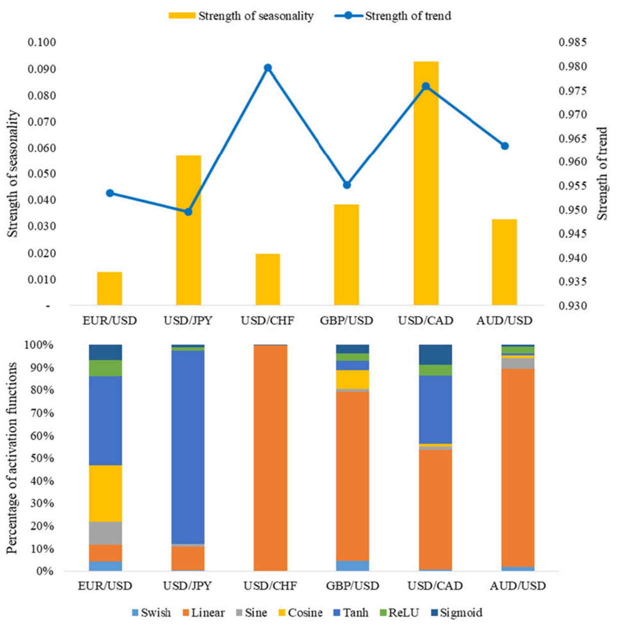


FIGURE 8. The relationship between average percentage activation functions and strength of trend and seasonality for each currency rate.

and thus improving the generalization capacity of the deep neural networks.

V. CONCLUSION

In this paper, we propose a novel deep learning framework equipped with an adaptive activation function selection mechanism for forecasting foreign currency rate. The proposed framework consists of two major components: the first component contains seven neural networks consisting of different activation functions such as ReLU, linear, sigmoid and tanh, sine, cosine and swish and; the second component is the softmax layer that produces the probabilities or importance weights for seven networks. Those dynamic importance weights decide which neural network is more important than others.

In the experimental study, the results showed that our proposed deep learning model achieved better performances compared with other baseline models on six foreign exchange rates. In addition, we extended Min-Max normalization method for non-stationary financial time series using VaR technique. Our extended Min-Max normalization method can be used with the neural networks with non-linear activation functions for forecasting non-stationary financial time series.

Finally, the authors anticipate potential future work in this area that includes developing Meta and Adaptive learning models [52], [57] for financial time series.

REFERENCES

[1] Y. W. Cheung and M. D. Chinn, "Currency traders and exchange rate dynamics: A survey of the US market," *J. Int. Money Finance*, vol. 20, no. 4, pp. 71–439, Aug. 2001.
[2] M. P. Taylor and H. Allen, "The use of technical analysis in the foreign exchange market," *J. Int. Money Finance*, vol. 11, no. 3, pp. 304–314, 1992.
[3] J. Yao and C. L. Tan, "A case study on using neural networks to perform technical forecasting of forex," *Neurocomputing*, vol. 34, nos. 1–4, pp. 79–98, 2000.
[4] J. W. Taylor, "Volatility forecasting with smooth transition exponential smoothing," *Int. J. Forecast*, vol. 20, no. 2, pp. 273–286, 2004.
[5] F.-M. Tseng, G.-H. Tzeng, H.-C. Yu, and B. J. C. Yuan, "Fuzzy ARIMA model for forecasting the foreign exchange market," *Fuzzy Sets Syst.*, vol. 118, no. 1, pp. 9–19, 2001.
[6] E. Yang, H. W. Park, Y. H. Choi, J. Kim, L. Munkhdalai, I. Musa, and K. H. Ryu, "A simulation-based study on the comparison of statistical and time series forecasting methods for early detection of infectious disease outbreaks," *Int. J. Environ. Res. Public Health*, vol. 15, no. 5, p. 966, 2018.
[7] S. Galeshchuk and S. Mukherjee, "Deep networks for predicting direction of change in foreign exchange rates," *Intell. Syst. Account Finance Manage.*, vol. 24, no. 4, pp. 100–110, 2017.
[8] G. S. Atsalakis and K. P. Valavanis, "Surveying stock market forecasting techniques-Part II: Soft computing methods," *Expert Syst. Appl.*, vol. 36, no. 3, pp. 5932–5941, 2009.
[9] E. Guresen, G. Kayakutlu, and T. U. Daim, "Using artificial neural network models in stock market index prediction," *Expert Syst. Appl.*, vol. 38, no. 8, pp. 10389–10397, 2011.
[10] J. Kamruzzaman, R. A. Sarker, and I. Ahmad, "SVM based models for predicting foreign currency exchange rates," in *Proc. ICDM*, Melbourne, FL, USA, 2003, pp. 557–560.
[11] C. Ullrich, D. Seese, and S. Chalup, "Foreign exchange trading with support vector machines," in *Advances in Data Analysis*. Berlin, Germany: Springer, 2007, pp. 539–546.
[12] D. Pradeepkumar and V. Ravi, "FOREX rate prediction using chaos and quantile regression random forest," in *Proc. RAIT*, Dhanbad, India, Mar. 2016, pp. 517–522.
[13] S. McNally, J. Roche, and S. Caton, "Predicting the price of bitcoin using machine learning," in *Proc. PDP*, Cambridge, U.K., 2018, pp. 339–343.
[14] D. Brownstone, "Using percentage accuracy to measure neural network predictions in stock market movements," *Neurocomputing.*, vol. 10, no. 3, pp. 237–250, 1996.
[15] S. Galeshchuk, "Neural networks performance in exchange rate prediction," *Neurocomputing*, vol. 172, pp. 446–452, Jan. 2016.
[16] M. Jaruszewicz and J. Ma dziuk, "One day prediction of NIKKEI index considering information from other stock markets," in *Proc. Int. Conf. Artif. Intell. Soft Comput.* Berlin, Germany: Springer, 2004, pp. 1130–1135.
[17] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, no. 7, 2017, Art. no. e0180944.
[18] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," 2017, *arXiv:1704.02971*. [Online]. Available: https://arxiv.org/abs/1704.02971
[19] M. Abe and H. Nakayama, "Deep learning for forecasting stock returns in the cross-section," in *Proc. PAKDD*, Melbourne, VIC, Australia, 2018, pp. 273–284.
[20] T. Fischer and C. Krauss, "Deep learning with long short-term memory networks for financial market predictions," *Eur. J. Oper. Res.*, vol. 270, no. 2, pp. 654–669, 2018.
[21] J. W. Lee, "Stock price prediction using reinforcement learning," in *Proc. ISIE*, Busan, South Korea, vol. 1, 2001, pp. 690–695.
[22] Z. Jiang, D. Xu, and J. Liang, "A deep reinforcement learning framework for the financial portfolio management problem," 2017, *arXiv:1706.10059*. [Online]. Available: https://arxiv.org/abs/1706.10059
[23] S. Almahdi and S. Y. Yang, "An adaptive portfolio trading system: A risk-return portfolio optimization using recurrent reinforcement learning with expected maximum drawdown," *Expert Syst. Appl.*, vol. 87, pp. 267–279, Nov. 2017.
[24] H. J. Kim and K. S. Shin, "A hybrid approach based on neural networks and genetic algorithms for detecting temporal patterns in stock markets," *Appl. Soft Comput.*, vol. 7, no. 2, pp. 569–576, 2007.
[25] R. K. T. Rathnayaka, D. M. K. N. Seneviratna, W. Jianguo, and H. I. Arumawadu, "A hybrid statistical approach for stock market forecasting based on artificial neural network and ARIMA time series models," in *Proc. BESC*, Durham, NC, USA, Oct./Nov. 2015, pp. 54–60.
[26] O. Castillo and P. Melin, "Hybrid intelligent systems for time series prediction using neural networks, fuzzy logic, and fractal theory," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1395–1408, Nov. 2002.

[27] J. Korczak and M. Hemes, "Deep learning for financial time series forecasting in A-trader system," in *Proc. FedCSIS*, Prague, Czech Republic, 2017, pp. 905–912.
[28] L. Ni, Y. Li, X. Wang, J. Zhang, J. Yu, and C. Qi, "Forecasting of forex time series data based on deep learning," *Procedia Comput. Sci.*, vol. 147, pp. 647–652, Jan. 2019.
[29] R. Salles, K. Belloze, F. Porto, P. H. Gonzalez, and E. Ogasawara, "Nonstationary time series transformation methods: An experimental review.," *Knowl Based Syst.*, vol. 164, pp. 274–291, 2018.
[30] G. Armano, M. Marchesi, and A. Murru, "A hybrid genetic-neural architecture for stock indexes forecasting," *Inf. Sci.*, vol. 170, no. 1, pp. 3–33, 2005.
[31] E. Constantinou, R. Georgiades, A. Kazandjian, and G. P. Kouretas, "Regime switching and artificial neural network forecasting of the cyprus stock exchange daily returns," *Int. J. Finance Econ.*, vol. 11, no. 4, pp. 371–383, 2006.
[32] J. V. Pérez-Rodríguez, S. Torra, and J. Andrada-Félix, "STAR and ANN models: Forecasting performance on the Spanish 'Ibex-35' stock index," *J. Empirical Finance*, vol. 12, no. 3, pp. 490–509, 2005.
[33] C. H. Jin, G. Pok, Y. Lee, H. W. Park, K. D. Kim, U. Yun, and K. H. Ryu, "A SOM clustering pattern sequence-based next symbol prediction method for day-ahead direct electricity load and price forecasting," *Energy Convers. Manage.*, vol. 90, pp. 84–92, Jan. 2015.
[34] S. C. Hui, M. T. Yap, and P. Prakash, "A hybrid time lagged network for predicting stock prices," *Int. J. Comput. Internet Manage.*, vol. 8, no. 3, pp. 26–40, 2000.
[35] W. Leigh, M. Paz, and R. Purvis, "An analysis of a hybrid neural network and pattern recognition technique for predicting short-term increases in the NYSE composite index," *Omega*, vol. 30, no. 2, pp. 69–76, 2002.
[36] E. Ogasawara, L. C. Martinez, D. De Oliveira, G. O. Zimbr, G. L. Pappa, and M. Mattoso, "Adaptive normalization: A novel data normalization approach for non-stationary time series," in *Proc. IJCNN*, Barcelona, Spain, 2010, pp. 1–8.
[37] D. Zhang, Q. Jiang, and X. Li, "Application of neural networks in financial data mining," *Int. J. Comput. Inf. Eng.*, vol. 1, no. 1, pp. 225–228, 2007.
[38] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*. [Online]. Available: https://arxiv.org/abs/1710.05941
[39] G. Parascandolo, H. Huttunen, and T. Virtanen. (2016). *Taming the Waves: Sine as Activation Function in Deep Neural Networks*. [Online]. Available: https://openreview.net/pdf?id=Sks3zF9eg
[40] M. Y. Liu, K. Wu, and H. F. Lee. (2004). *VaR Estimation With Power EWMA Model-Conservativeness, Accuracy and Efficiency*. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=494285
[41] A. Feuerverger and A. C. Wong, "Computation of value-at-risk for nonlinear portfolios," *J. Risk*, vol. 3, pp. 37–56, Oct. 2000.
[42] M. G. Greg, C. F. Christopher, and B. Mickey. (Apr. 1997). Creditmetrics-technical document. JP Morgan. New York, NY, USA. Accessed: Nov. 19, 2019. [Online] Available: http://homepages.rpi.edu/~guptaa/MGMT4370.10/Data/CreditMetrics.pdf/
[43] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychol. Rev.*, vol. 65, no. 6, p. 386, 1958.
[44] J. L. Elman, "Finding structure in time," *Cognit. Sci.*, vol. 14, no. 2, pp. 179–211, Mar. 1990.
[45] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," 2014, *arXiv:1409.1259*. [Online]. Available: https://arxiv.org/abs/1409.1259
[46] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[47] F. Agostinelli, M. Hoffman, P. Sadowski, and P. Baldi, "Learning activation functions to improve deep neural networks," 2014, *arXiv:1412.6830*. [Online]. Available: https://arxiv.org/abs/1412.6830
[48] E. Batbaatar, H. W. Park, D. Li, M. Li, and K. H. Ryu, "DeepEnergy: Prediction of appliances energy with long-short term memory recurrent neural network," in *Proc. ACIIDS*, 2018, pp. 224–234.
[49] L. Munkhdalai, T. Munkhdalai, K. H. Park, T. Amarbayasgalan, B. Erdenebaatar, H. W. Park, and K. H. Ryu, "An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series," *IEEE Access*, vol. 7, pp. 99099–99114, 2019.
[50] G. E. Hinton and D. C. Plaut, "Using fast weights to deblur old memories," in *Proc. CogSci*, 1987, pp. 177–186.
[51] J. Schmidhuber, "Learning to control fast-weight memories: An alternative to dynamic recurrent networks," *Neural Comput.*, vol. 4, no. 1, pp. 131–139, 1992.
[52] T. Munkhdalai and H. Yu, "Meta networks," in *Proc. ICML*, Jul. 2017, pp. 2554–2563.
[53] T. Munkhdalai, X. Yuan, S. Mehri, and A. Trischler, "Rapid adaptation with conditionally shifted neurons," 2017, *arXiv:1712.09926*. [Online]. Available: https://arxiv.org/abs/1712.09926
[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Jun. 2016, pp. 770–778.
[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980
[56] X. Wang, K. Smith, and R. Hyndman, "Characteristic-based clustering for time series data," *Data Mining Knowl. Discovery*, vol. 13, no. 3, pp. 64–335, 2006.
[57] T. Munkhdalai, A. Sordoni, T. Wang, and A. Trischler, "Metalearned neural memory," 2019, *arXiv:1907.09720*. [Online]. Available: https://arxiv.org/abs/1907.09720

LKHAGVADORJ MUNKHDALAI received the M.Sc. degree from the Database and Bioinformatics Laboratory, Department of Computer Science, Chungbuk National University, Cheongju, South Korea, in 2018, where he is currently pursuing the Ph.D. degree. His research interests are econometric, data mining, and artificial neural nets for financial application, such as credit scoring, financial time series forecasting, and portfolio management.

TSENDSUREN MUNKHDALAI received the bachelor's degree from the National University of Mongolia, Ulaanbaatar, Mongolia, and the M.Sc. and Ph.D. degrees from the Department of Computer Science, Chungbuk National University, South Korea. He is currently a Researcher with Microsoft Research Montreal. He spent two years at the University of Massachusetts as a Postdoc sitting right next to the Neurology Department. His research interests include meta-learning, memory and attention systems, rapid and temporal adaptations, and feedbacks in artificial neural nets for language and text understanding

KWANG HO PARK received the B.S. degree in biochemistry from Chungbuk National University, Cheongju, South Korea, in 2015, and the M.Sc. degree from the Database and Bioinformatics Laboratory, Chungbuk National University, in 2017, where he is currently pursuing the Ph.D. degree. His main research interests are data mining, bioinformatics, and medical informatics.

HEON GYU LEE received the M.S. and Ph.D. degrees in computer science from Chungbuk National University, Cheongju, South Korea, in 2005 and 2009, respectively. In 2016, he joined GAION, Seoul, South Korea, where he is currently a Senior Vice President with the Big Data Research Laboratory. His research interests include data mining, DB, bioinformatics, machine learning, and big data analysis.

MEIJING LI received the B.S. degree in computer science from Dalian University, Dalian, China, in 2007, and the M.S. degree in bioinformation technology and the Ph.D. degree in computer science from Chungbuk National University, Cheongju, South Korea, in 2010 and 2015, respectively. She held a postdoctoral position at Chungbuk National University. She is currently an Assistant Professor with the College of Information Engineering, Shanghai Maritime University, Shanghai, China. Her research interests include data mining, information retrieval, database systems, bioinformatics, and biomedicine.

KEUN HO RYU (M'82) received the Ph.D. degree in computer science and engineering from Yonsei University, South Korea, in 1988, and the Honorary Doctorate degree from the National University of Mongolia. He has served at the Reserve Officers' Training Corps (ROTC) of the Korean Army. He has been the Leader of the Database and Bioinformatics Laboratory, South Korea, since 1986. He has worked at The University of Arizona, Tucson, AZ, USA, as a Postdoctoral and a Research Scientist, and the Electronics and Telecommunications Research Institute, South Korea, as a Senior Researcher. He is the former Vice-President of the Personalized Tumor Engineering Research Center. He is currently a Professor with Chungbuk National University, South Korea, and the Faculty of Information Technology, Ton Duc Thang University, Vietnam. He has published or presented over 1000 refereed technical articles in various journals and international conferences, in addition to authoring a number of books. His research interests include temporal databases, spatiotemporal databases, temporal GIS, stream data processing, knowledge-based information retrieval, data mining, biomedical informatics, and bioinformatics. He has been a member of ACM, since 1983. He has served on numerous program committees, including roles as the Demonstration Co-Chair of the VLDB, the Panel and Tutorial Co-Chair of the APWeb, and the FITAT General Co-Chair.