IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 April 2023, accepted 8 May 2023, date of publication 15 May 2023, date of current version 26 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3275964

RESEARCH ARTICLE

Developing Novel Robust Loss Functions-Based Classification Layers for DLLSTM Neural Networks

MOHAMAD ABOU HOURAN[1], (Senior Member, IEEE), MOHAMED H. ESSAI ALI[2], ADEL B. ABDEL-RAMAN[3,4], EMAN A. BADRY[5], ALAAELDIEN HASSAN[1,3], AND HANY A. ATALLAH[3]

[1]School of Electrical Engineering, Xi'an Jiaotong University, Xi'an 710049, China
[2]Department of Electrical Engineering, Faculty of Engineering, Al-Azhar University, Qena 11765, Egypt
[3]Electrical Engineering Department, Faculty of Engineering, South Valley University, Qena 83523, Egypt
[4]Department of Electronics and Communications Engineering (ECE), Egypt-Japan University of Science and Technology, Alexandria 21934, Egypt
[5]Department of Electrical and Communication Engineering, Higher Institute of Engineering and Technology at Al-Tod, Luxor 85842, Egypt

Corresponding author: Mohamed H. Essai Ali (mhessai@azhar.edu.eg)

This work was supported by the National Natural Science Foundation of China under Grant 52150410418.

ABSTRACT In this paper, we suggest improving the performance of developed activation function-based Deep Learning Long Short-Term Memory (DLLSTM) structures by employing robust loss functions like Mean Absolute Error (*MAE*) and Sum Squared Error (*SSE*) to create new classification layers. The classification layer is the last layer in any DLLSTM neural network structure where the loss function resides. The LSTM is an improved recurrent neural network that fixes the problem of the vanishing gradient that goes away and other issues. Fast convergence and optimum performance depend on the loss function. Three loss functions (default(*Crossentropyex*), (*MAE*) and (*SSE*)) that compute the error between the actual and desired output for two distinct applications were used to examine the effectiveness of the suggested DLLSTM classifier. The results show that one of the suggested classifiers' specific loss functions(*SSE*)) works better than other loss functions and does a great job. The suggested functions *Softsign*, Modified-Elliott, Root-sig, Bi-tanh1, Bi-tanh2, *Sech* and *wave* are more accurate than the tanh function.

INDEX TERMS DNN, DLLSTM, loss function, mean absolute error, sum squared error.

I. INTRODUCTION

In recent years, the machine learning (ML) community has come to regard the deep learning (DL) computer paradigm as the Gold Standard. It has also steadily become the most popular computational strategy in the field of machine learning. This is because it does several difficult cognitive tasks as well as or better than humans' performance [1]. A Deep Neural Network (DNN) is a particular type of neural network represented as a multilayer perceptron (MLP), which is trained using algorithms to learn representations from data sets without the need for manually designing feature extractors [2].

As the name DL suggests, it has more or deeper levels of processing than a shallow learning model, which has fewer layers of units [3]. With a deeper knowledge and use of the backpropagation algorithm, self-directed learning was made possible [4]. Deep learning neural networks (DLNNs) are used in a variety of industries for three reasons [5]. First, since DLNN-based classifiers are data-based, they are more resistant to imperfections in real systems. Second, DLNN-based classifiers have a minimum of computational complexity, requiring only a few simple matrix and vector operations at various levels. Thirdly, with the fast development of parallel processing capability in specialized processors like graphic processing units (GPU) [6]. DLNN-based techniques are significantly more efficient because DLNN implementation is simple to parallelize on parallel architectures and easy to implement with low data type accuracy. These benefits helped DLNNs to look the way they did and do well in many fields [7].

The associate editor coordinating the review of this manuscript and approving it for publication was Liang-Bi Chen.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

RNNs are a widely used and well-known algorithm in the field of DL. RNN is primarily utilized in contexts related to speech processing and Natural Language Processing (NLP) [8]. RNN uses sequential data in the network, as opposed to traditional networks. Since the inherent structure in the sequence of the data provides essential information and is necessary for a few different applications, it is important to know the context of the sentence in order to determine the meaning of a specific word. The RNN can therefore be thought of as a short-term memory unit [9].

Hochreiter and Schmidhuber Long Short-Term Memory (LSTM) structure has been proven to be efficient for a variety of learning issues, especially those necessitating large data sets. The LSTM structure is composed of ''blocks,'' which are collections of units that are repeatedly connected to one another. Develop LSTM techniques, structures, and transfer functions to deal with the issue of disappearing or exploding gradients. These make the network more accurate as it trains deeper [10]. One of the best things about DL is how flexible it is when it comes to architectural design. This means that there are many ways to put priors over data into the model and find the best activation functions, learning algorithms, or loss functions [11].

Frank et al. [12] reviewed some of the recent developments and efforts that used ML in science and engineering. They believed that despite its enormous success over recent years, ML is still in its infancy and will play a significant role in scientific research and engineering over the upcoming years. Poulinakis et al. [13] presented a work that shed light on the limitations of various ML and cubic spline methods when data is sparse and noisy. As a result, they discovered the true function hidden under the noise, thus making ML a valuable tool in practical applications. Additionally, they focused on hypothetical generalized functions with and without noise. The conclusions from this study are beneficial in guiding further research regarding the splines and ML modeling.

A critical component of training a deep learning model is the loss function, one of the hyperparameters that can be adjusted. They are employed to determine the difference between the actual and desired outputs' accuracy and loss. As a result of the loss, the DL network changes the weights for the connections between the neurons or classifiers [14]. The performance of the resulting DL model can be affected by the loss function selection. In fact, the recent works on customized loss functions exhibit strong offensive performance on the selected datasets. It should be noted, nonetheless, that a loss function that performs well in one offensive context need not perform similarly in another. In other words, the number of traces, model architecture, and the initialization of the weight are just a few of the many factors that affect both the offensive performance and the loss function [15].

Farzad et al. [16] compared 23 different activation functions in which the three gates (the input, output, and forget gate) changed activation functions while the block input and block output activation functions were held constant with the hyperbolic tangent so that the activation functions of the block could be compared (tanh). The authors have recommended altering the hyperbolic tangent function on the block input and block output as a better alternative for altering the activation functions in the three gates. In addition, they suggested that additional research should be done on other components of an LSTM network. Ali et al. [17] presented qualitative research to improve the performance of LSTM-based classifiers by developing the internal structure of LSTM neural networks using 26 state activation functions as alternatives to the traditional hyperbolic tangent (tanh) activation function and only using default loss functions (Crossentropyex).

In this paper, we expand on our preceding research work [17]. In [17], as an alternative to the conventional (*tanh*) activation function, we have created a conceptual framework for brand-new LSTM-based classifiers that exploit the internal organization of LSTM networks. The findings demonstrate that the suggested LSTM classifiers outperformed the conventional (*tanh*)-based LSTM classifiers and made some progress. In this paper, we present qualitative research to improve the performance of previously developed activation function-based LSTM structures by using robust loss functions like (*MAE*) and (*SSE*) to build new classification layers. The classification layer is the last in any LSTM neural network structure where the loss function resides and use the best suggested DLLSTM-based classifiers from the preceding research work [17].

More precisely, we systematically compare commonly used loss functions (Cross-entropy) and proposed loss functions(*MAE*), (*SSE*) in the DLLSTM base-classifiers. We evaluate the attack performance (*Crossentropyex*), and the number of trainable parameters. Different loss functions on two available datasets are evaluated. The proposed DLLSTM classifiers will be trained using an adaptive moment estimation (*adam*) optimizer and different loss functions to get the most reliable and accurate performance under the conditions of the classifiers. To the best of our knowledge, this is the first time the DLLSTM neural network has been used to build classifiers for different loss functions. These loss functions are critical in style transfer because they determine how much the accuracy has been altered. We will discuss the limitations of the loss functions already used and propose different combinations of loss functions for better accuracy.

The contributions of this study are as follows.

1) We started by compiling a multitude of functions that can be utilized in DLLSTM networks in place of the conventional (*tanh*) function.

2) Examining how various loss functions, which include (*Crossentropyex*), (*MAE*) and (*SSE*) affect the way suggested DLLSTM networks train.

3) Developing ''(*hard* − *sigmoid*)'' gate function-based DLLSTM classifiers and comparing their performance with the commonly used (*sigmoid*) gate function-based DLLSTM

classifiers in the presence of the suggested state functions and (*adam*) optimizer.

4) Employing the recently developed DLLSTM networks to solve a wide range of real-world classification tasks, including vowel classification and image classification.

This Paper is organized as follows. The DLLSTM structure and activation functions are provided in Section II. Providing the methods is Section III. Section IV presents the simulation results of the proposed approach. The conclusion is presented in Section V.

II. DLLSTM STRUCTURE AND ACTIVATION FUNCTIONS (AF)

The parts that follow will provide a quick explanation of the DLLSTM structure and the activation functions used on the network.

A. DLLSTM STRUCTURE

The LSTM network is a recurrent neural network with the ability to detect long-term relationships between the time steps of sequence data [18]. Numerous LSTM-based methods have been created to fix problems, including handwriting recognition [19], audio recognition [9], and online translation using tools like Google neural machine translation [20] and the Facebook translation system [21]. The simplest DLLSTM with a single hidden neuron, batch normalization, and output units is used to classify the data. The DLLSTM structure, which consists of the input, single hidden neurons, and output units, is shown in Figure 1. The elements in each cell are identified using (1) through (6).

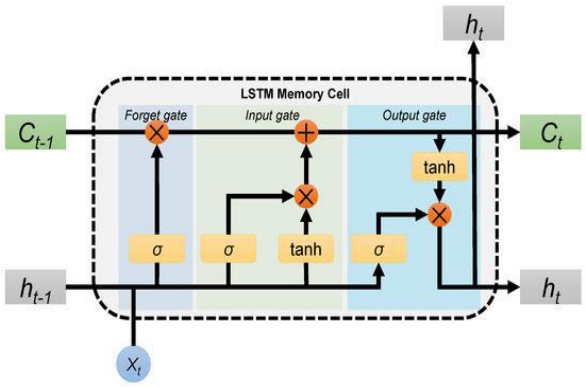

FIGURE 1. LSTM memory cell [22].

Gate function (*sigmoid*) and state function (*tanh*), both found in DLLSTM memory cells as Figure 1, are the two most prevalent activation mechanisms for the neurons in memory blocks and the state activation function (*tanh*) [19].

The variables of the DLLSTM memory cell are specified by equations (1) to (6).

$$f_t = \sigma(W_f \chi_t + U_f h_{t-1} + b_f) \tag{1}$$
$$i_t = \sigma(W_i \chi_t + U_i h_{t-1} + b_i) \tag{2}$$
$$O_t = \sigma(W_O \chi_t + U_o h_{t-1} + b_o) \tag{3}$$
$$C'_t = tanh\,(W_c \chi_t + U_c h_{t-1} + b_c) \tag{4}$$
$$C_t = f_t \odot_{t-1} + i_t \odot C'_t \tag{5}$$
$$h_t = O_t \odot tanh(C_t) \tag{6}$$

Equations (1) to (3) describe the forget, input, and output gates for each DLLSTM cell, where i_t refers to the input, O_t denotes the output, and f_t is the forget gates. C'_t in (4), the block input specifies the volume of data that should be saved in the cell at computing time. C_t an update of the state of time t. Lastly, h_t is the output blocks at the appropriate time [23].

B. ACTIVATION FUNCTIONS (AF)

To support the detection of complicated datasets and to allow the insertion of non-linearity into network without any of the requirement for coding, AF is introduced to an ANN. The AF determines what information should be provided to the following neuron at the end of the process in the cell model of human brains. Using this cell, the output is collected from the cell before and changed it into a format that may be utilized as an input for the cell after [2].

A bad choice of functions can cause the NN's gradients to vanish or explode, as well as cause input data to be lost. The training process, the AF used in NN, and the network structure between cells are the three main factors that affect how well networks operate. The effectiveness of the network is significantly impacted by each of these factors [24]. The relevance of the learning algorithm has dominated research on NNs, whereas the activation functions that are used in these networks have been ignored [25].

In this study, the DLLSTM network is reconstructed by substituting one of the functions indicated in Table 1 for each of the (*tanh*) activation functions found in Equations (4), (5), and (6). Furthermore, we evaluate the effects of employing 17 various functions in (*tanh*) gates of a fundamental DLLSTM cell on network performance. The *tanh* formula is given in (7).

$$tanh\,(x) = \frac{sinh(x)}{cosh\,(x)} \tag{7}$$

The formula of the *sigmoid* function is given below [26].

$$\sigma\,(x) = \frac{1}{e^{-x} - 1} \tag{8}$$

We have compiled a comprehensive list of 17 functions, as shown in Table 1. We experimentally observed that adding a value of 0.5 to some functions makes them applicable as activation functions in the network. Changing the range of the activation functions has been previously observed in other studies [27].

In Table 1, first column to the left, reported activation functions are given as follows. Wave function [28], Softsign function [29]. Then, Bi-sig1, Bi-sig2, Bi-tanh-1, and Bi-tanh-2 functions are suggested by Sodhi and Chandra [30], Cloglog and Cloglog-m [31], Elliott, Gaussian, Logsigm and complementary loglog functions [32]. Next, Logsigm and Log-sigmoid, followed by the Modified-Elliott function [17].

After that comes sigmoid functions with roots, also called Rootsig [33]. Then, And the hyperbolic secant (Sech) [34], and last function is Gaussian Error Linear Unit (GELU) [35].

TABLE 1. Identification and differentiation of functions.

Label	Activation function	Derivative function
Wave	$f(x) = (1 - x^2)e^{-x^2}$	$f'(x) = 2x(x^2 - 2)e^{-x^2}$
Softsign	$f(x) = \frac{x}{1+\|x\|} + 0.5$	$f'(x) = \frac{1}{(1+\|x\|)^2}$
Bi-sig1	$f(x) = \frac{1}{2}(\frac{1}{1+e^{-x+1}} + \frac{1}{1+e^{-x-1}})$	$f'(x) = \frac{e^{1-x}}{(e^{1-x}+1)^2} + \frac{e^{-x-1}}{(e^{-x-1}+1)^2}}{2}$
Bi-sig2	$f(x) = \frac{1}{2}(\frac{1}{1+e^{-x}} + \frac{1}{1+e^{-x-1}})$	$f'(x) = \frac{\frac{e^{-x}}{(e^{-x}+1)^2} + \frac{e^{-x-1}}{(e^{-x-1}+1)^2}}{2}$
Bi-tanh1	$f(x) = \frac{1}{2}\left[tanh\ tanh\left(\frac{x}{2}\right) + tanh(\frac{x+1}{2})\right] + 0.5$	$f'(x) = \frac{sech^2\left(\frac{x+1}{2}\right) + sech^2\left(\frac{x}{2}\right)}{4}$
Bi-tanh2	$f(x) = \frac{1}{2}\left[tanh\ tanh\left(\frac{x-1}{2}\right) + tanh(\frac{x+1}{2})\right] + 0.5$	$f'(x) = \frac{sech^2\left(\frac{x+1}{2}\right) + sech^2\left(\frac{x-1}{2}\right)}{4}$
Cloglog	$f(x) = 1 - e^{-e^x}$	$f'(x) = e^{x-e^x}$
Cloglogm	$f(x) = 1\text{-}2e^{-0.7e^x} + 0.5$	$f'(x) = 7e^{x-0.7e^x}/5$
Elliott	$f(x) = \frac{0.5x}{1+\|x\|} + 0.5$	$f'(x) = \frac{0.5}{(1+\|x\|)^2}$
Gaussian	$f(x) = e^{-x^2}$	$f'^{(x)} = -2xe^{-x^2}$
Loglog	$f(x) = e^{-e^x} + 0.5$	$f'(x) = e^{-e^x-x}$
Logsigm	$f(x) = (\frac{1}{1+e^{-x}})^2 + 0.5$	$f'(x) = \frac{2e^{-x}}{(e^{-x}+1)^3}$
Log-sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	$f'(x) = \frac{e^{-x}}{(e^{-x}+1)^2}$
Modified-Elliott	$f(x) = \frac{x}{\sqrt{1+x^2}} + 0.5$	$f'(x) = \frac{1}{(x^2+1)^{\frac{3}{2}}}$
Rootsig	$f(x) = \frac{x}{1+\sqrt{1+x^2}} + 0.5$	$f'(x) = \frac{1}{\sqrt{x^2+1}+x^2+1}$
Sech	$f(x) = \frac{2}{e^x + e^{-x}}$	$f'^{(x)} = -\frac{2(e^x - e^{-x})}{(e^x + e^{-x})^2}$
GELU	$f(x) = 0.5x(1 + \tanh(\sqrt{2/\pi}(x + 0.447x^3)))$	$f'(x) = 0.5\tanh(0.0356x^3 + 0.797x) + (0.0535x^3 + 0.398x)sech^2(0.0356x^3 + 0.797x) + 0.5$

III. LOSS FUNCTIONS AND METHODOLOGY

A. LOSS FUNCTIONS

In this paper, supervised learning is implemented using both LSTMs and RNNs. The loss, determined by a loss function, is the difference between the model's predicted label and the actual labels that really correspond to the input. The output of the loss function is used to change the network's weights so that the difference between the expected and actual labels is less [36].

We use three different loss functions in the proposed network, compare how well they work, and look at how each one works to find out which one gives the best results.

1) *Crossentropyex* [37] is commonly used in machine learning as a loss function and is a measurement of the difference between two probability distributions for a given random variable or a set of events. *Crossentropyex* Loss, also known as log loss, measures how well a classification model performs when producing a probability between 0 and 1. (*Crossentropyex*) Loss develops as the predicted probability departs from the label. Targets and outputs are given, and the (*Crossentropyex*) function uses additional parameters and optional performance weights to figure out how well the network is doing.

The *Crossentropy* function has the formula is given by:

$$\text{Crossentropyex} = -\sum_{i=1}^{N}\sum_{j=1}^{c} X_{ij}(k)\log\left(\hat{X}_{ij}(k)\right) \quad (9)$$

2) Sum Squared Error (*SSE*) [38] is a measure of accuracy in which the errors are squared and added. Find the dataset mean by adding up all the values and dividing them by the total number of values to determine the sum of squares for error. The deviation for each value is then determined by subtracting the mean from each value. Square each value's deviance next. A network performance function is SSE. The sum of squared errors is used to measure performance. The regularization of the errors and the normalization of the outputs and targets are controlled by two optional function arguments in the formula perf = *SSE* (net, t, y, ew, Name, Value).

The Sum Squared Error function has the formula is given by:

$$\text{SSE} = \sum_{i=1}^{N}\sum_{j=1}^{c}\left(X_{ij}(k) - \hat{X}_{ij}(k)\right)^2 \quad (10)$$

3) Mean Absolute Error (*MAE*) [39] is calculated by taking the difference between the predictions made by your model and the actual data, adding the absolute value to that difference, and then averaging the result across the entire dataset. *MAE* is used to measure network performance. A loss function expression has the formula is given by:

$$\text{MAE} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{c}\left|X_{ij}(k) - \hat{X}_{ij}(k)\right|}{N} \quad (11)$$

where N is the quantity of observations, c is the number of categories, X_{ij} is the ith categorized data for the jth c amount of categories class and $\hat{X}_{ij}\hat{X}_{ij}$ is the output of sample ii for a category j [40].

Loss functions provide more than just a static illustration of how well your model is doing; they also act as the foundation upon which your algorithms fit data. Most machine learning algorithms have some kind of loss function that is used to find the best parameters (weights) for your data or to optimize them [41]. Importantly, the choice of the loss function is directly related to the activation function used in the output layer of your neural network. These two design elements are connected.

B. METHODOLOGY

We updated the function (*tanh*) that is used to choose the cell input and update the output in order to examine the impacts of various loss functions on the performance of DLLSTM classifiers. The suggested DLLSTM classifiers will initially be trained using the standard function (*sigmoid*) and then they will be trained using a ($hard - sigmoid$) function. In each combination, two identical gates are used, and they are chosen from the AF list in Table 1 for each structure to compare the effects of various loss functions (($Crossentropyex$), (MAE) and (SSE)) on the accuracy of the DLLSTM classifiers.

IV. SIMULATION RESULTS

The proposed DLLSTM classifiers are trained using the BPTT approach [42] and (*adam*) optimizer with a variety of loss functions, such as (*Crossentropyex*), (*MAE*), and (*SSE*). The classifiers are developed using 100 hidden neurons for each trial, and the initial parameters are selected at random. Based on the outcomes of the two datasets, the losses and efficiency of each DLLSTM-based classifier are determined.

The evaluation requirements for the classifiers includes accuracy. Accuracy is what determines how much testing information has been correctly recognized. It matches the definition given below:

$$Accuracy = \frac{number\ of\ true\ classified\ samples}{number\ of\ total\ test\ samples} \times 10 \quad (12)$$

A. FIRST SET OF EXPERIMENTS

We used data sets from the Japanese vowels dataset in this experiment for the initial set of trials. 9 male users speaking 2 consecutive Japanese vowels (ae) in a multivariate time series made up the initial vowel set from the University of Southern California. It was done in a variety of ways: a linear prediction study with a sampling rate of 10 kHz, a frame length of 25.6 ms, and a shift length of 6.4 ms. In other words, each time the speaker speaks, a time series between 7 and 29 is created, with a total of 12 features present at each point in the series (12 coefficients). 640 time series make up the entire collection, which is a round number [43].

Table 2 lists the structure variables, training possibilities, various hidden neuron numbers, and loss functions for the suggested DLLSTM-based classifiers. In order to increase performance, the batch size has been determined based on research. For each design, the outcomes of the two tests are used to report the accuracy and loss. According to Table 3, suggested loss function (*SSE*)-based classification layers outperform conventional loss functions *Crossentropyex* and suggested loss function (*MAE*)-based classification layers at (sigmoid) functions, enabling DLLSTM classifiers to attain the maximum efficiency. In addition to the *tanh* function, which gets an efficiency of 93.5432%, 11 others suggested DLLSTM classifiers provide high accuracy with efficiencies in the ranges of 93-98.2378%. Based on experiments, the *wave*-DLLSTM classifier outperformed *tanh* function with an efficiency of 98.2378%.

TABLE 2. Highlights the architecture of the suggested DLLSTM parameter and training option.

Parameters	Values
Input data size	12 inputs
Measurements of hidden neurons	100 neurons
Size of batches	27
Number of Epoch	100
Size of layer connection	9
initially weighted networks	Random
Optimizers	$(adam)$
functional loss	$(Crossentropyex)$, (MAE), (SSE)

TABLE 3. Comparing the results of various DLLSTM classifiers using (sigmoid) function, and different loss functions for japanese vowels dataset.

Activation Functions		$Cross\ entropyex)$	MAE	SSE	Gate Act. Fun.& optimizer
Default Act. Fun.	$Tanh$	93.5432	93.5135	94.3243	$(sigmoid$ & $(adam)$
Propose Act. Fun.	*Gaussian*	95.5757	95.1351	96.2162	
	Wave	97.5676	97.2973	**98.2378**	
	Softsign	95.6757	94.3243	95.9459	
	GELU	95.4643	32.57	90.9189	
	Cloglog	91.6216	72.2341	92.7027	
	Cloglogm	95.4054	91.8216	96.7568	
	Rootsig	94.5946	87.8378	95.4054	
	Sech	96.2351	94.0541	96.7568	
	Loglog	92.4324	70.2973	92.9730	
	Elliott	88.9189	86.58	91.3514	
	Bisig1	92.1622	73.2584	94.3243	
	Bisig2	90.2703	71.8919	90.5	
	Bitanh1	95.1351	95.1351	97.0270	
	Bitanh2	95.6757	94.8649	96.4865	
	Logsigm	95.1351	79.4595	95.6757	
	Log-sigmoid	92.4324	78.1081	93.2432	
	Modified-Elliott	95.3243	93.5135	95.6757	

The performance curves for the suggested *wave* DLLSTM classifier, which has the best performance, and the conventional (*tanh*) DLLSTM classifier are shown in Figures 2 and Figure 3 at (*sigmoid*) gate function.

Table 4 shows how well each classifier performed when the $hard-sigmoid$ function was used in place of the (*sigmoid*)

TABLE 4. Comparing the results of various suggested DLLSTM classifiers using.

Activation Functions		(*Cross entropyex*)	(*MAE*)	(*SSE*)	Gate Act. Fun. & optimizer
Default Act. fun	*Tanh*	94.3243	94.0541	94.5946	(*hard − sigmoid*) & (*adam*)
Propose Act. Fun.	*Gaussian*	94.4054	94.3243	95.4054	
	Wave	97.0270	93.5135	**98.0162**	
	Softsign	95.9459	95.1351	96.4865	
	GELU	95.541	32.5714	94.3243	
	Cloglog	95.8649	73.584	86.2162	
	Cloglogm	94.3243	92.258	96.7568	
	Rootsig	96.4865	93.2432	96.7568	
	Sech	95.6757	94.3243	96.2162	
	Loglog	94.5946	78.1081	95.6757	
	Elliott	93.2432	90.258	94.9584	
	Bi-sig1	95.9459	79.4595	95.9459	
	Bi-sig2	93.7854	90.258	95.5135	
	Bi-tanh1	95.9457	79.8472	96.5587	
	Bi-tanh2	96.7568	81.9730	96.8865	
	Logsigm	96.0270	80	96	
	Log-sigmoid	94.8514	90.2581	95.33	
	Modified-Elliott	96.7568	86.1622	96.8872	

function using the suggested (*SSE*) classification layer, which outperformed the default function (*Crossentropyex*) and suggested function *MAE*-based classification. In comparison to the (*tanh*), which achieves an accuracy of 94,323%. The other 17 suggested DLLSTM classifiers provide high accuracy with efficiencies in the range of 93- 98.0162%. According to tabulated results, 15 of the 17 DLLSTM-based classifiers that have been proposed exceeding the (*tanh*) function, with the *wave* function having the highest efficiency (98.0162%). The performance curves for the suggested *wave* DLLSTM classifier, which has the best performance, and the conventional *tanh* classifier are displayed in Figure 4 and Figure 5 at (*hard − sigmoid*) function.

In general, the DLLSTM classifiers using *SSE*-based classifications perform better than the loss functions *Crossentropye* and *MAE*-based classification, with a *hard − sigmoid* function outperforming the *sigmoid* function.

Figure 6 and Figure 7 illustrate and analyze more highly AF DLLSTM classifiers, which use the *sigmoid* and *hard − sigmoid* functions respectively and use a variety of loss functions ((*Crossentropyex*),(*MAE*) and (*SSE*)) for training.

The *wave* function outperforms the (*tanh*) function by achieving a high classification rate of 98.2378%, as compared

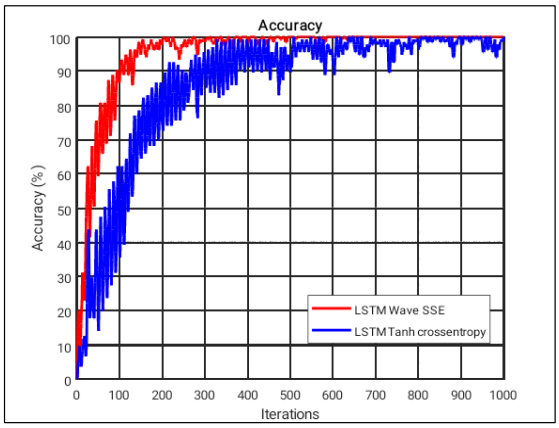

FIGURE 2. The performance curves (accuracy) for the suggested DLLSTM classifier and (*tanh*) using different loss functions, (*adam*) optimizer, and (*sigmoid*) function.

to the latter's 93.4054% when using the (*SSE*) loss function. The *wave* DLLSTM classifier is also the most effective of the suggested classifiers. It appears to work but performs substantially worse when using the suggested (*MAE*) loss function. The *wave* function is the most effective among the suggested classifiers as well. The suggested Modified-Elliott, Rootsig, Sech, Wave, Bi-tanh1, Bi-tanh2, and Softsign based DLLSTM classifiers often perform better than *tanh* function. Additionally, the examined functions that employ the *hard − sigmoid* function outperform the standard function.

B. SECOND SET OF EXPERIMENTS

The second dataset of the experiments will be built upon the Weather Reports Classification System. The dataset illustrates how to use bag-of-words models to develop a simplistic text classifier using word frequency values. By following the guidelines below, word frequency count can be used as a variable in a straightforward classifier. The dataset demonstrates how to develop a straightforward classifier model to identify the category of weather reports using the available text descriptions.

Table 5 lists the structure variables, training possibilities, various hidden neuron numbers, and loss functions for the suggested DLLSTM-based classifiers. In order to increase performance, the batch size has been determined based on research. For each design, the outcomes of the two tests are used to report the accuracy and loss.

Table 6 and Table 7 show the efficiency classifier rates for each DLLSTM classifier used to describe Weather Reports, utilizing optimizer *adam*, *sigmoid*, and *hard − sigmoid* functions. And loss functions (*Crossentropyex*), (*MAE*) and (*SSE*)-based classification layers, respectively at 100 hidden neurons. The classifier receives all the dataset in small batches at each trial, serving as the standard function of the DLLSTM structure. The observed DLLSTM classification results serve as a baseline for comparison.

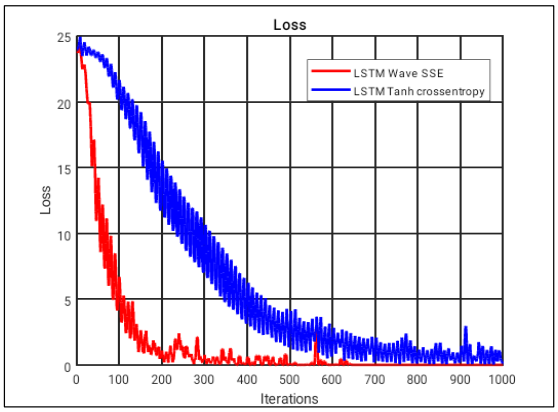

FIGURE 3. The performance curves (loss) for the suggested DLLSTM classifier and (*tanh*) Using different loss functions, (*adam*) optimizer, and (*sigmoid*) function for Japanese vowels dataset.

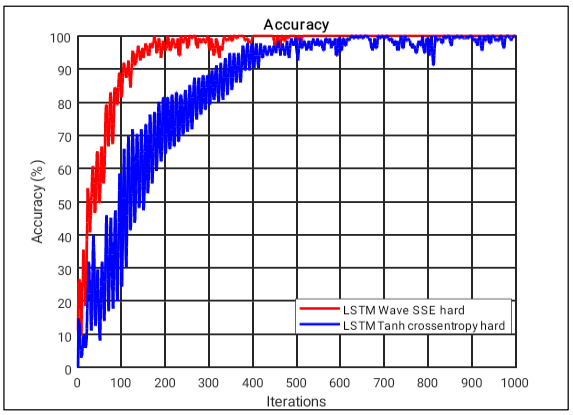

FIGURE 4. The performance curves (accuracy) for the suggested DLLSTM classifier and (*tanh*) using different loss functions and *hard* − *sigmoid* for Japanese vowels dataset.

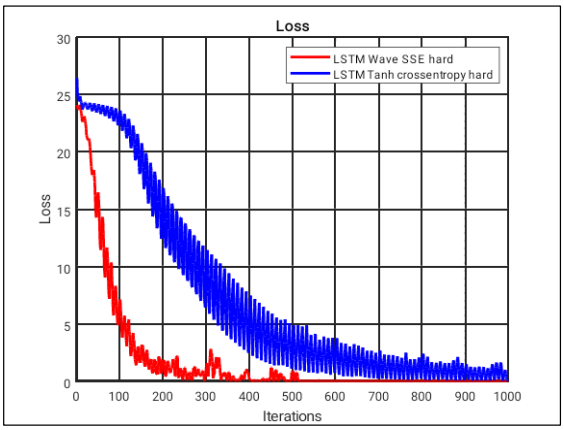

FIGURE 5. The performance curves (loss) for the suggested DLLSTM classifier and (*tanh*) using different loss functions *hard* − *sigmoid* for Japanese vowels dataset.

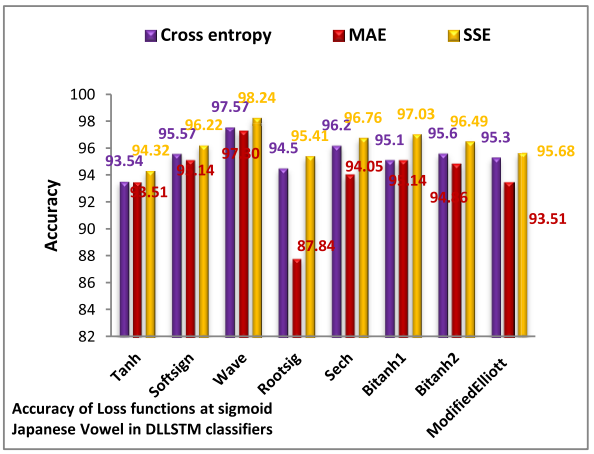

FIGURE 6. Comparing the best result of DLLSTM classifiers using various loss functions ((*Crossentropyex*), (*MAE*) and (*SSE*)) with 100 hidden neurons using (*sigmoid*) gate function.

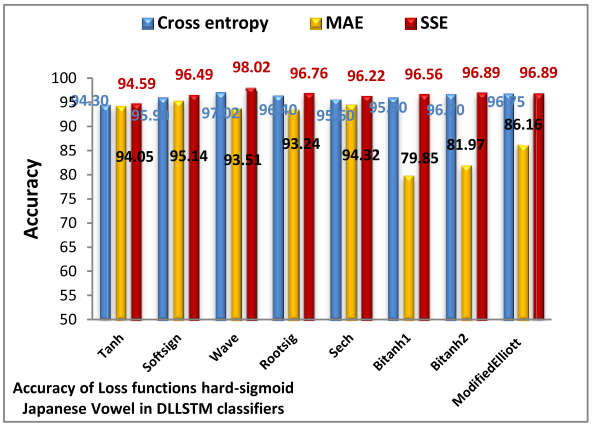

FIGURE 7. Comparing the best result of DLLSTM classifiers using various loss functions (*Crossentropyex*), (*MAE*) and (*SSE*)) with 100 hidden neurons using (hard-sigmoid) gate function.

TABLE 5. Architecture highlights of the suggested DLLSTM parameter and training option.

Parameters	Values
Input data size	1
Measurements of hidden neurons	100 neurons
Size of batches	27
Number of Epoch	10
Threshold of Gradient	1
initially weighted networks	Random
Optimizers	$(adam)$
functional loss	$(Crossentropyex)$, (MAE), (SSE)

According to Table 6, the suggested loss function (*SSE*)-based classification layer outperforms conventional loss functions *Crossentropyex* and suggested loss function (*MAE*)-based classification layers at (*sigmoid*) functions, enabling DLLSTM classifiers to attain the maximum efficiency. In addition to the *tanh* function, which gets an efficiency of 86.1%, 14 others given DLLSTM classifiers provide high accuracy with efficiencies in the ranges of 84-**89.503**%. According to experimental studies, the *Softsign* DLLSTM classifier outperformed the (*tanh*) function with an

TABLE 6. Comparative performances of different proposed activation function-based LSTM classifiers for weather reports dataset, using adam optimizer and (sigmoid) gate activation function.

	activation functions	(*Cross entropyex*	(*MAE*)	(*SSE*) %	Gate Act. Fun. & optimizer
Default act. Fun	*Tanh*	86.19	36.25	86.5	(*sigmoid*) & (*adam*)
Propose Act. Fun.	Gaussian	86.4528	32.7581	86.6	
	Wave	84.3214	32.7581	85.87	
	Softsign	88.0485	32.7581	**89.503**	
	GELU	87.4526	32.7581	32.7581	
	Cloglog	83.0258	32.7581	84.4	
	Cloglogm	84.3625	32.7581	86.64	
	Rootsig	87.5281	32.7581	88.28	
	Sech	86.5241	32.7581	87.35	
	Loglog	82.3258	32.7581	83.28	
	Elliott	85.5225	32.7581	86.56	
	Bi-sig1	85.5238	32.7581	83.8	
	Bi-sig2	84.6814	32.7581	84.9	
	Bitanh1	87.7251	32.7581	87.86	
	Bitanh2	86.5262	32.7581	87.49	
	Logsigm	86.4257	32.7581	86.95	
	Logsigmoid	85.2571	32.7581	85.17	
	Modified-Elliott	*87.985*	32.7581	87.5	

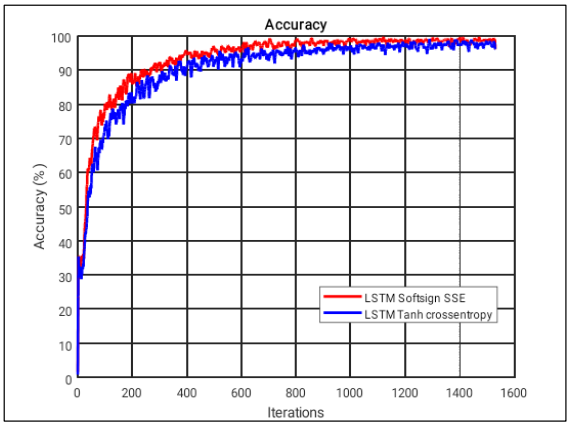

FIGURE 8. The performance curves (accuracy) for the suggested DLLSTM classifier and (*tanh*) using different loss functions, (*adam*) optimizer, and (*sigmoid*) function for weather reports dataset.

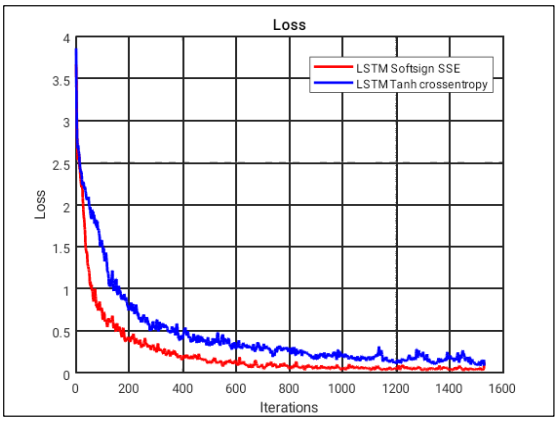

FIGURE 9. The performance curves (loss) for the suggested DLLSTM classifier and (*tanh*) using different loss functions, (*adam*) optimizer, and (*sigmoid*) function for weather reports dataset.

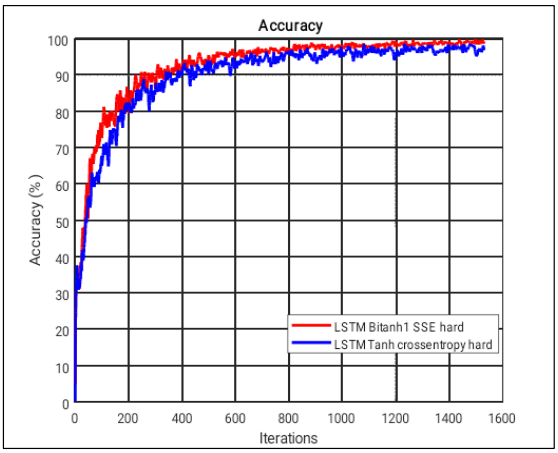

FIGURE 10. The performance curves (accuracy) for the suggested DLLSTM classifier and (*tanh*) Using different loss functions, (*adam*) optimizer, and (*hard − sigmoid*) function for weather reports dataset.

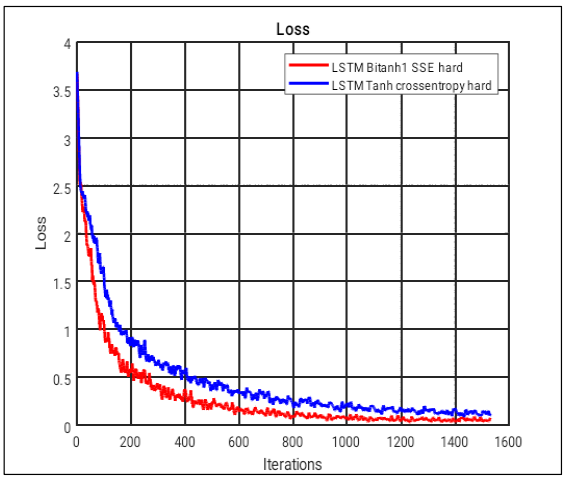

FIGURE 11. The performance curves (loss) for the suggested DLLSTM classifier and (*tanh*) Using different loss functions, (*adam*) optimizer, and (*hard − sigmoid*) function for weather reports dataset.

efficiency of **89.503** %. The performance curves for the given *wave* DLLSTM classifier, which has the best performance, and the conventional (*tanh*) DLLSTM classifier are shown in Figure 8 and Figure 9 at (*sigmoid*) gate function.

Table 7 shows how well each classifier performed when the *hard − sigmoid* function was used in place of the (*sigmoid*) function using the suggested (*SSE*) classification layer, which outperformed the default function (*Crossentropyex*) and suggested function *MAE* based classification. In comparison to the (*tanh*), which achieves an accuracy of 86.3243%,

TABLE 7. Comparing the results of various suggested DLLSTM classifiers using (hard-sigmoid) function and different loss functions for weather reports dataset.

	activation functions	(*Cross entropyex*)	(*MAE*)	(*SSE*)	Gate Act. Fun. & optimizer
Default Act. Fun	*Tanh*	86.3	83.2581	86.5587	(*hard – sigmoid*) & (*adam*)
Propose Act. Fun.	*Gaussian*	87.9521	32.7581	84.9	
	Wave	80.6987	32.7581	85.55	
	Softsign	87.6258	32.7581	88.2515	
	GELU	86.945	32.7581	32.7581	
	Cloglog	82.5135	32.7581	84.58	
	Cloglogm	87.8547	32.7581	86.64	
	Rootsig	87.9521	32.7581	87.74	
	Sech	87.9581	32.7581	87.92	
	Loglog	83.4595	32.7581	85.51	
	Elliott	83.3247	32.7581	85.4	
	Bisig1	85.7382	32.7581	84.3	
	Bisig2	85.1478	32.7581	82.7	
	Bitanh1	87.5527	32.7581	**88.86**	
	Bitanh2	87.4523	32.7581	88.34	
	Logsigm	86.2581	32.7581	86.6	
	Logsigmoid	87.5847	32.7581	85.3	
	Modified-Elliott	84.1471	32.7581	87.5	

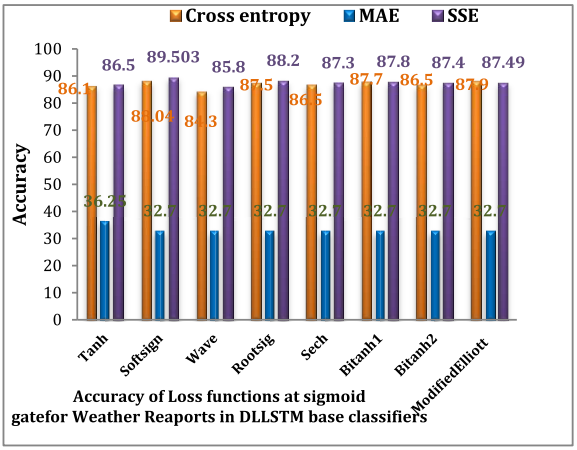

FIGURE 12. Comparing the best result of DLLSTM classifiers using various loss functions (*Crossentropyex*), (*MAE*) and (*SSE*)) with and 100 hidden neurons using (*sigmoid*) gate function.

15 others suggested DLLSTM classifiers provide high accuracy with efficiencies in the range of 84-88.8%. According to tabulated results, 10 of the 17 DLLSTM classifiers that have been suggested exceeding the (*tanh*) function, with the $Bi-tanh1$ function having the highest efficiency (88.8%).

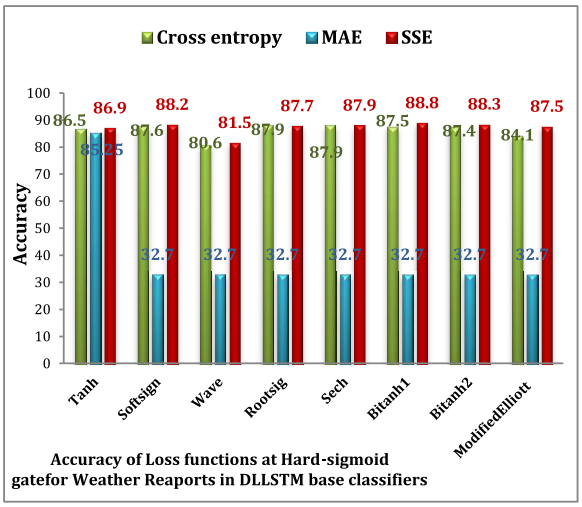

FIGURE 13. Comparing the best result of DLLSTM classifiers using various loss functions (Crossentropyex), (MAE) and (SSE)) with and 100 hidden neurons using (hard – sigmoid) gate function.

The performance curves for the suggested $Bi-tanh1$ DLLSTM classifier, which has the best performance, and the conventional *tanh* classifier are shown in Figure 10 and Figure 11 at (*hard – sigmoid*) function. Overall, the DLLSTM classifiers using *SSE*-based classification performs better than the loss functions *Crossentropye* and *MAE*-based classification, with a *hard – sigmoid* function outperforming the *sigmoid* function.

Figures 12 and 13 illustrate and analyze more highly AF DLLSTM classifiers. And using the *sigmoid*, $hard-sigmoid$ functions respectively, and using a variety of loss functions ((*Crossentropyex*), (*MAE*) and (*SSE*)) to train. The *Softsign* function clearly outperforms the (*tanh*) function by achieving a high classification rate of **89.5**% as compared to the latter's 86.5%, when using the (*SSE*) loss function. The *Softsign* DLLSTM classifier is also the most effective of the suggested classifiers. It appears to work but performs substantially worse when using the suggested (*MAE*) loss function. The suggested Modified-Elliott, Root-sig, Sech, *Wave*, Bi-tanh1, Bi-tanh2, and (*Softsign*) based DLLSTM classifiers often perform better than the (*tanh*) function. Additionally, the examined functions that employ the (*hard – sigmoid*) function outperform the standard function.

V. CONCLUSION

In this study, a novel robust loss function-based classification layer for Deep Learning Long Short-Term Memory (DLLSTM) has been proposed. The suggested classifiers are initially trained with *sigmoid* function and then with *hard – sigmoid* function, to visualize the classification issues. Also, a comparative study was performed using three distinct loss functions ((*Crossentropyex*), (*MAE*) and (*SSE*)), and (*adam*) optimizer. Our tests with different data sets showed that the suggested (*SSE*) worked much better than the (*Crossentropyex*), and (*MAE*) loss functions. Another recently suggested loss function, called (*MAE*) loss,

performed substantially worse and appears to be limited to certain neural network architectures. The results additionally indicated that the suggested classifiers that apply *hard* − *sigmoid* function were better than that use *dsigmoid*. The analysis indicated that certain less well-liked AFs, including Modified-Elliott, Root-sig, *Sech*, Bi-tanh1, Bi-tanh2, *wave* and Softsign, exhibited lower rates of losses than the most well-liked AFs. This means that classifiers that use these less popular AFs are more likely to get good results than those that use the *tanh* function. Finally, the choice of (*SSE*) loss could be indeed confirmed as a strong option, which improved the accuracy of DLLSTM-based classifiers, achieving **98.24** % *wave* accuracy in the Japanese Vowels dataset and **89.5**% *Softsign* accuracy in the Weather Reports dataset.

The following ideas are suggested for future research:

1) Analyzing the effectiveness of the suggested DLLSTM-based classifiers using a variety of optimization methods, such as *RMSPropSgdm*, *ADadelte*, *Adagrad*, *AMSgrad*, *AdaMax*, and *Nadam*.

2) Analyzing the effectiveness of the DLLSTM using additional loss functions like Huber and Cauchy to provide more robust loss functions rather than the crossentropyex loss function, the suggested classifiers will perform better under the constraints of classification in real systems.

REFERENCES

[1] A. Shrestha and A. Mahmood, "Review of deep learning algorithms and architectures," *IEEE Access*, vol. 7, pp. 53040–53065, 2019.

[2] M. A. Alotaibi, "Machine learning approach for short-term load forecasting using deep neural network," *Energies*, vol. 15, no. 17, p. 6261, Aug. 2022.

[3] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[4] S. K. Sahu, A. Mokhade, and N. D. Bokde, "An overview of machine learning, deep learning, and reinforcement learning-based techniques in quantitative finance: Recent progress and challenges," *Appl. Sci.*, vol. 13, no. 3, p. 1956, Feb. 2023.

[5] I. Jacobs and C. Bean, "Fine particles, thin films and exchange anisotropy (effects of finite dimensions and interfaces on the basic properties of ferromagnets)," in *Spin Arrangements and Crystal Structure, Domains, and Micromagnetics*. Schenectady, NY, USA: General Electric Research Laboratory, 2013, pp. 271–350.

[6] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, "Deep learning for wireless physical layer: Opportunities and challenges," *China Commun.*, vol. 14, no. 11, pp. 92–111, Nov. 2017.

[7] B. Barz and J. Denzler, "Deep learning on small datasets without pre-training using cosine loss," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1360–1369.

[8] K.-L. Du and M. Swamy, "Recurrent neural networks," in *Neural Networks and Statistical Learning*. Berlin, Germany: Springer, 2019, pp. 351–371.

[9] M. Bukhsh, M. S. Ali, A. Alourani, K. Shinan, M. U. Ashraf, A. Jabbar, and W. Chen, "Long short-term memory recurrent neural network approach for approximating roots (Eigen values) of transcendental equation of cantilever beam," *Appl. Sci.*, vol. 13, no. 5, p. 2887, Feb. 2023.

[10] K. Vijayaprabakaran and K. Sathiyamurthy, "Towards activation function search for long short-term model network: A differential evolution based approach," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 6, pp. 2637–2650, Jun. 2022.

[11] M. Gheisari, G. Wang, and M. Z. A. Bhuiyan, "A survey on deep learning in big data," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE), IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, vol. 2, Jul. 2017, pp. 173–180.

[12] M. Frank, D. Drikakis, and V. Charissis, "Machine-learning methods for computational science and engineering," *Computation*, vol. 8, no. 1, p. 15, Mar. 2020.

[13] K. Poulinakis, D. Drikakis, I. W. Kokkinakis, and S. M. Spottswood, "Machine-learning methods on noisy and sparse data," *Mathematics*, vol. 11, no. 1, p. 236, Jan. 2023.

[14] B. Usharani, "ILF-LSTM: Enhanced loss function in LSTM to predict the sea surface temperature," *Soft Comput.*, vol. 26, pp. 1–13, Mar. 2022.

[15] M. Kaur, S. Satapathy, R. Soundrapandiyan, and J. Singh, "Targeted style transfer using cycle consistent generative adversarial networks with quantitative analysis of different loss functions1," *Int. J. Knowl.-based Intell. Eng. Syst.*, vol. 22, no. 4, pp. 239–247, Dec. 2018.

[16] A. Farzad, H. Mashayekhi, and H. Hassanpour, "A comparative performance analysis of different activation functions in LSTM networks for classification," *Neural Comput. Appl.*, vol. 31, no. 7, pp. 2507–2521, Jul. 2019.

[17] M. H. E. Ali, A. B. Abdel-Raman, and E. A. Badry, "Developing novel activation functions based deep learning LSTM for classification," *IEEE Access*, vol. 10, pp. 97259–97275, 2022.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.

[20] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, and J. Klingner, "Google's neural machine translation system: Bridging the gap between human and machine translation," 2016, *arXiv:1609.08144*.

[21] T. Ong. (2017). *Facebook's Translations Are Now Powered Completely by AI*. [Online]. Available: https://www.theverge.com/2017/8/4/16093872/facebook-ai-translationsartificial-intelligence

[22] H. Fan, M. Jiang, L. Xu, H. Zhu, J. Cheng, and J. Jiang, "Comparison of long short term memory networks and the hydrological model in runoff simulation," *Water*, vol. 12, no. 1, p. 175, Jan. 2020.

[23] W. Khan, A. Daud, F. Alotaibi, N. Aljohani, and S. Arafat, "Deep recurrent neural networks with word embeddings for Urdu named entity recognition," *ETRI J.*, vol. 42, no. 1, pp. 90–100, Feb. 2020.

[24] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," 2014, *arXiv:1409.2329*.

[25] A. Apicella, F. Donnarumma, F. Isgro, and R. Prevete, "A survey on modern trainable activation functions," *Neural Netw.*, vol. 138, pp. 14–32, Jun. 2021.

[26] F. G. Oztiryaki and T. Piskin, "Airfoil performance analysis using shallow neural networks," in *Proc. AIAA Scitech Forum*, Jan. 2021, p. 0174.

[27] H. Burhani, W. Feng, and G. Hu, "Denoising AutoEncoder in neural networks with modified Elliott activation function and sparsity-favoring cost function," in *Proc. 3rd Int. Conf. Appl. Comput. Inf. Technol./2nd Int. Conf. Comput. Sci. Intell.*, Jul. 2015, pp. 343–348.

[28] D. Bala, "Childhood pneumonia recognition using convolutional neural network from chest X-ray images," *J. Electr. Eng., Electron., Control Comput. Sci.*, vol. 7, no. 4, pp. 33–40, 2021.

[29] T. E. Simos and C. Tsitouras, "Efficiently inaccurate approximation of hyperbolic tangent used as transfer function in artificial neural networks," *Neural Comput. Appl.*, vol. 33, no. 16, pp. 10227–10233, Aug. 2021.

[30] S. S. Sodhi and P. Chandra, "Bi-modal derivative activation function for sigmoidal feedforward networks," *Neurocomputing*, vol. 143, pp. 182–196, Nov. 2014.

[31] G. D. S. Gomes, T. B. Ludermir, and L. M. Lima, "Comparison of new activation functions in neural network for forecasting financial time series," *Neural Comput. Appl.*, vol. 20, no. 3, pp. 417–439, Apr. 2011.

[32] G. S. D. S. Gomes and T. B. Ludermir, "Complementary log-log and probit: Activation functions implemented in artificial neural networks," in *Proc. 8th Int. Conf. Hybrid Intell. Syst.*, Sep. 2008, pp. 939–942.

[33] W. Duch and N. Jankowski, "Survey of neural transfer functions," *Neural Comput. Surv.*, vol. 2, no. 1, pp. 163–212, 1999.

[34] P. Chandra and Y. Singh, "A case for the self-adaptation of activation functions in FFANNs," *Neurocomputing*, vol. 56, pp. 447–454, Jan. 2004.

[35] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*.

[36] *Constructive Side-Channel Analysis and Secure Design* (Lecture Notes in Computer Science), vol. 13211. Cham, Switzerland: Springer, 2022, doi: 10.1007/978-3-030-99766-3_2.

[37] J. Brownlee. (2019). *A Gentle Introduction to Cross-Entropy for Machine Learning*. [Online]. Available: https://machinelearningmastery.com/cross-entropy-for-machine-learning
[38] R. Nainggolan, R. Perangin-angin, E. Simarmata, and A. F. Tarigan, "Improved the performance of the K-means cluster using the Sum of Squared Error (SSE) optimized by using the elbow method," *J. Phys., Conf.*, vol. 1361, no. 1, Nov. 2019, Art. no. 012015.
[39] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *PeerJ Comput. Sci.*, vol. 7, p. e623, Jul. 2021.
[40] M. H. E. Ali and I. B. M. Taha, "Channel state information estimation for 5G wireless communication systems: Recurrent neural networks approach," *PeerJ Comput. Sci.*, vol. 7, p. e682, Aug. 2021.
[41] A. M. Abdul-Hadi, M. A. Naser, M. Alsabah, S. H. Abdulhussain, and B. M. Mahmmod, "Performance evaluation of frequency division duplex (FDD) massive multiple input multiple output (MIMO) under different correlation models," *PeerJ Comput. Sci.*, vol. 8, p. e1017, Jun. 2022.
[42] T. Gorecki, L. Smaga, and M. T. Gorecki, "Package 'mfds,'" S.home.amu.edu.pl, Tech. Rep., 2017.
[43] R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent," *Theory, Archit., Appl.*, vol. 433, p. 17, Jan. 1995.

MOHAMAD ABOU HOURAN (Senior Member, IEEE) received the B.S. degree in electrical engineering from the Faculty of Mechanical and Electrical Engineering, Damascus University, Syria, in 2008, and the M.S. and Ph.D. degrees from the School of Electrical Engineering, Xi'an Jiaotong University (XJTU), China, in 2014 and 2020, respectively. He is currently an Assistant Professor with the School of Electrical Engineering, XJTU. His research interests include Wireless Power Transfer (WPT), Power Electronics, and Power Systems, and Interdisciplinary Research. He is a member of the IEEE Power Electronics Society (PELS). He is a reviewer for some IEEE and Elsevier journals, such as IEEE TRANSACTIONS ON POWER ELECTRONICS, IEEE ACCESS, and *Applied Energy*.

EMAN A. BADRY received the B.Sc. degree in system and computer engineering from Al-Azhar University, Egypt, in 2010, and the M.Sc. degree in communication and electrical engineering from South Valley University, Qena, Egypt, in 2019. She is currently pursuing the Ph.D. degree in communication and electrical engineering. Since 2019, she has been a Teaching Assistant with the Electrical Department, Higher Institute for Engineering and Technology at Al-Tod, Luxor. Her research interests include artificial intelligence, deep learning, machine learning, and computer science.

ADEL B. ABDEL-RAMAN received the B.S. and M.S. degrees in electrical engineering, communication, and electronics from Assuit University, Assuit, Egypt, in 1991 and 1998, respectively, and the Dr.Ing. degree in communication engineering from Otto von Guericke University at Magdeburg, Magdeburg, Germany, in 2005. He was the Executive Director of information and communication technology with South Valley University, Qena, Egypt, from 2010 to 2012. Since October 2012, he has been an Associate Professor with the School of Electronics, Communications and Computer Engineering, Egypt-Japan University of Science and Technology (E-JUST), Alexandria, Egypt. Since August 2016, he has been the Dean of the Faculty of Computers and Information, South Valley University. He is currently a Professor of communication engineering with the Electrical Engineering Department, South Valley University. He has published more than 100 refereed journal articles and conference papers. He has one patent. He was the main supervisor for more than 17 M.Sc. and Ph.D. students. He is a reviewer of *IEEE Microwave and Wireless Components Letters*.

MOHAMED H. ESSAI ALI was born in El-Balyana, Sohag, Egypt, in 1978. He received the B.S. degree in electrical engineering from Al-Azhar University, Egypt, in 2001, the M.S. degree in electrical engineering from Assuit University, Egypt, in 2007, and the Ph.D. degree in mechanical engineering from Novosibirsk State Technical University, Novosibirsk, Russia, in 2012. From 2001 to 2008, he was a Demonstrator and a Lecturer Assistant with Al-Azhar University. From 2012 to 2018 he was an Assistant Professor with Al-Azhar University. From April 2014 to December 2014, he was a Guest Researcher with Novosibirsk State Technical University. Since 2018, he has been an Associate Professor with the Electrical Engineering Department, Faculty of Engineering, Al-Azhar University. He is the author of five textbooks and more than 43 articles. His research interests include the theory and applications of robust statistics, wireless communication, the channel estimation of signals in terms of a priori uncertainty for the problems of telecommunications, optical wireless communication, artificial intelligence-based signal processing applications, and FPGA-based applications.

ALAAELDIEN HASSAN was born in Qena, Egypt, in 1988. He received the B.S. and M.S. degrees in electrical engineering from the Faculty of Engineering, South Valley University, Qena, in 2010 and 2016, respectively. He is currently pursuing the Ph.D. degree with the School of Electrical Engineering, Xi'an Jiaotong University, China. His research interests include dc/ac converters, multilevel inverters, PWM techniques dc/ac and ac/dc converters, and the analysis of control strategies. Besides, his work is involved in integrating renewable energy systems, microgrids, and hybrid renewable energy systems. He has been awarded the South Valley University Prize for international publishing from 2018 to 2022. He is a reviewer of a high-ranked SCI journals.

HANY A. ATALLAH was born in Qena, Egypt. He received the B.Sc. and M.Sc. degrees in electrical engineering, electronics and communications from South Valley University, Egypt, in 2007 and 2012, respectively, and the Ph.D. degree in antennas and microwave engineering from the Egypt-Japan University for Science and Technology (E-JUST), in 2016. He was a Visiting Researcher with the E-JUST Laboratory, School of Information Science and Electrical Engineering, Kyushu University, Japan, from September 2015 to July 2016. He is currently an Associate Professor with the Electrical Engineering Department, Qena Faculty of Engineering, South Valley University. He is a reviewer of more than ten highly impacted journals. He has published more than 50 journal articles and conference papers. His research interests include antenna designs, dielectric resonators, metamaterials, tunable filters, reconfigurable antennas, antenna arrays, microwave filters, FDTD, cognitive radio (CR) antennas, wireless power transfer (WPT) for biomedical implants, electric vehicles, electronic devices, breast cancer detection, smart meters, and the Internet of Things (IoT).