IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 31, 2021, accepted May 24, 2021, date of publication June 3, 2021, date of current version June 15, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3085855

Improving the Performance of Deep Neural Networks Using Two Proposed Activation Functions

ASMAA A. ALKHOULY[1], AMMAR MOHAMMED[1,2], AND HESHAM A. HEFNY[1]
[1]Computer Science Department, Faculty of Graduate Studies of Statistical Researches, Cairo University, Giza 12613, Egypt
[2]Computer Science Department, Faculty of Computer Science, Misr International University, Cairo 11865, Egypt

Corresponding author: Asmaa Alkhouly (asmaa_alkholy@pg.cu.edu.eg)

ABSTRACT

In artificial neural networks, activation functions play a significant role in the learning process. Choosing the proper activation function is a major factor in achieving a successful learning performance. Many activation functions are sufficient universal approximators, but their performance is lacking. Thus, many efforts have been directed toward activation functions to improve the learning performance of artificial neural networks. However, the learning process involves many challenges, such as saturation, dying, and exploding/vanishing the gradient problems. The contribution of this work resides in several axes. First, we introduce two novel activation functions: absolute linear units and inverse polynomial linear units. Both activation functions are augmented by an adjustable parameter that controls the slope of the gradient. Second, we present a comprehensive study and a taxonomy of various types of activation functions. Third, we conduct a broad range of experiments on several deep neural architecture models with consideration of network type and depth. Fourth, we evaluate the proposed activation functions' performance in image and text classification tasks. For this purpose, several public benchmark datasets are utilized to evaluate and compare the performance of the proposed functions with that of a group of common activation functions. Finally, we deeply analyze the impact of several common activation functions on deep network architectures. Results reveal that the proposed functions outperform most of the popular activation functions in several benchmarks. The statistical study of the overall experiments on both classification categories indicates that the proposed activation functions are robust and superior among all the competitive activation functions in terms of average accuracy.

INDEX TERMS Artificial neural network, deep neural network, learning challenges, activation function.

I. INTRODUCTION

The successful applications of machine learning techniques rely on the approximation functions that are learned from the underline data of problems [1]. The artificial neural network (ANN) is one of the machine learning techniques that can be used as a universal approximator for complex data [2]. The ability of ANNs to approximate complex data increases when the number of hidden layers increases [3]–[5]. Thus, deep neural networks (DNNs) have become the most successful machine learning technique for learning many complex problems and they have been applied extensively to several domains such as image classification, object detection, object segmentation, and emotion recognition [6]–[9]; speech recognition; and text classification [10]–[13]. Traditionally, complex problems involve high dimensional nonlinear data. To effectively learn such problems, ANNs should use the nonlinear activation functions(AFs) of their hidden layers [14], [15]. Empirical studies have proved that nonlinear mappings are easier to optimize and they converge rapidly [16], [17]. For such purpose, many popular nonlinear activation functions have been used to fulfill the conditions of universal approximation theory [18], [19], such as Sigmoid, Tanh, SoftPlus, and ArcTan [20]. However, these functions still cause several problems, including vanishing gradient and saturation [21], which lead to a poor performance and undesirable

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

convergent learning of ANNs. To achieve an effective approximation, and address the aforementioned issues, plenty of research efforts have been proposed several activation functions [22]. Although most of the proposed activation functions do not fulfill the conditions of universal approximation theory, they achieve better empirical performance than those that follow such conditions. Despite the success of previous efforts, additional challenges, such as dying, saturation, and complexity cost problems [23]–[26], remain. For instance, the rectified linear unit *(ReLU)* [23] is one of the most simple and widely used nonlinear activation functions that achieve a relatively good performance in neural network learning. However, it causes the dying problem [20], [22]. Meanwhile, the *SoftPlus* function [27] suffers from the so-called left soft saturation. Similarly, the *ELish* [26] and *Mish* [28] functions are expensive in terms of computational cost. Thus, researchers have attempted to find activation functions so as to solve exploding/vanishing gradient problems [29]–[31], saturation, and dying; speed up learning; and obtain functions with low computational complexity. Generally, the saturation and dying problems, as two types of vanishing gradient problems, occur because of gradient-based learning. They worsen when the depth of the network increases [32], [33]. Such gradient problems can be eliminated whenever the activation units fire negative inputs with either negative values close to zero or small positive values not close to zero. In avoiding the exploding gradient problem and solving the saturation problem, a linear function is preferred to a polynomial or exponential function in mapping positive inputs. Thus, this work proposes two nonlinear activation functions to achieve high performance and resolve the dying and saturation problems. The first activation function yields negative values close to zero for negative inputs, and the second one yields small positive values for the same inputs. Both activation functions are designed with different mathematical functions for mapping positive and negative inputs.

The main contributions of this work can be summarized as follows. First, it introduces two new activation functions that can be used in shallow and deep ANNs. In particular, the first activation function, called the inverse polynomial linear unit (IpLU), has a small slope for the negative interval. The second activation function, called the absolute linear unit (AbsLU), fires small positive values for the negative interval. Both activation functions are unbounded above and comprise the so-called α parameter to adjust the negative interval slope. Second, this work extensively reviews and categorizes various types of activation functions found in the literature. Third, the proposed activation functions are applied to a wide range of network architectures with consideration of the types and depths of neural networks. In the conducted experiments, the proposed activation functions are evaluated and compared with other common activation functions. The experiments are categorized into two groups, namely, image and text classification groups. For each group of classification tasks, various neural network architectures are adopted.

The rest of the paper is organized as follows. Section II extensively surveys and categorizes research efforts on similar activation functions. Section III introduces the proposed activation functions. Section IV describes the experiments and the evaluation of the proposed activation functions on several networks and multiple benchmark datasets. Section V extensively discusses and analyzes the results of the experiments. Finally Section VI concludes the study.

II. LITERATURE REVIEW

Several research efforts have proposed different forms of activation functions that can be used inside the neurons of networks. Such activation functions are different, ranging from simple to more complex mathematical representations. A simple form of the activation function is a linear one [34]. Although this form is less expensive in computational complexity, it is limited and cannot learn complex problems. Thus, several types of non-linear activation functions have been proposed in the last decades. Most of these non-linear functions achieve state-of-the-art performance. This section extensively surveys and discusses the most popular research efforts of several activation functions. Some of those efforts cause the vanishing gradient and saturation problems, and other functions cause the dying problem. On the other hand, successful activation functions overcome the previous problems but suffer from computational complexity or consistently achieve higher performance. Still, there is no universal activation function that can achieve higher performance or make a faster convergence without causing gradient problems or high computational cost. Therefore, a lot of researches have been conducted to introduce new activation functions. In the following, we categorize most of those previous efforts into five categories based on the mathematical forms and characteristics of their underline mathematical functions. The first category contains those types of activation functions that are bounded. The second category includes those activation functions that contain rectifier units. The third category comprises those activation functions that are non-linear for the negative interval. The fourth category contains those activation functions that are non-linear and mathematically unbounded from above. The final category contains those activation functions that have increasing and decreasing property. In the following, we describe each category in more detail. Fig. 1 summarizes the categories of activation functions. Finally, the criteria which are used in evaluating the most popular activation functions are summarized in Tables 1, 2, 3 and 4.

A. CATEGORIES OF ACTIVATION FUNCTIONS

1) BOUNDED FUNCTIONS

The bounded activation functions (units) are those functions that are designed based on the properties of universal approximation theorem [18], [35], [36]. Logistic is one of the most common activation functions of this category. In machine learning, the term Sigmoid function is used especially to refer

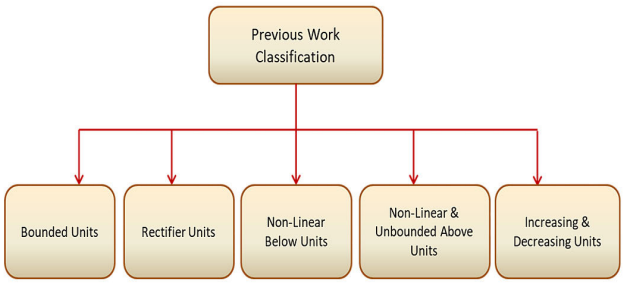

FIGURE 1. Categorizes of activation functions.

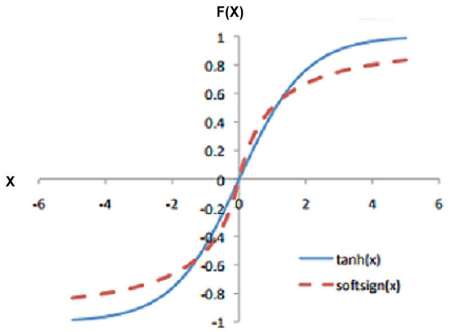

FIGURE 2. Softsign vs Tanh [40].

to the logistic function named by Pierre François Verhulst [37]. There are many functions of the form sigmoidal such as the logistic, the hyperbolic tangent, and the arctangent functions. Logistic is non-linear bounded and limits the output to be in the range between zero and one. It has a common S-shaped curve defined according to (1).

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

This function has been widely used in the early work of feed-forward neural networks. It is a differentiable and monotonically increasing function. Its outputs are smoothly continued with positive derivatives everywhere [38].

Sigmoid suffers from the saturation problem, which occurs when the gradient is too small, slowing down the learning process when the weights are updated proportionally to the gradient magnitude. If such gradient is small, it makes the network hard to train. The center of sigmoid inputs is 0.5. Therefore, it has a limited sensitivity because it is sensitive only to changes around 0.5. Likewise Sigmoid, the hyperbolic tangent (Tanh) function, is another activation belonging to the bounded group. Tanh has stronger gradients than Sigmoid, and its derivatives can reach up to 1.0, which makes the weights and bias update much larger. However, this function causes the vanishing gradients problem and has slow convergence, especially in deeper networks [27].It works better during training multi-layer neural networks than the Sigmoid function, which is more biologically plausible. Although each of Tanh and Sigmoid fulfill the approximation theory's conditions, their performance is poor in hidden layers, and both cause saturation problems.

Another known activation function, similar to Tanh, is Arctan function [35]. However, it squashes the input into the range from $-\pi/2$ to $\pi/2$ and causes the same problems of the Tanh function. Another activation function belonging to the bounded group is Softsign which was introduced in [39]. This function produces outputs in the range $[-1, +1]$ and is defined according to (2). Unlike Tanh, The Softsign has polynomial convergence, whereas the Tanh function has exponential convergence [40]. Fig. 2 illustrates the difference between Softsign and Tanh.

$$f(x) = \frac{x}{|\,x\,| + 1} \tag{2}$$

2) RECTIFIER FUNCTIONS

Rectified Linear units are firstly used within Restricted Boltzmann machines [23]. Then, the Rectified Linear Units (ReLU) was introduced by Glorot *et al.* in 2011 [41] and became the prevailing activation function used in deep neural nets, especially in Convolutional neural network. ReLU is more biologically plausible than the Sigmoid functions because biological neurons inspire this function that the lowest level of their activity can reach zero. Also, ReLU is characterized by its simplicity, computationally cheap, and fast convergence compared to the other activation functions belonging to the bounded groups. It is considered a sparse activation, as it maps all negative inputs into zeros. The term sparse activation means that fewer neurons are firing. In general, ReLU outperforms Sigmoid, and tanh activation functions in solving the saturation and vanishing gradient problems [42]. However, ReLU is not continuously differentiable, which causes some problems in gradient-based optimization. As a result of the negative input region's deactivation, the weights will not get adjusted during the gradient descent. Hence, the gradient becomes zero and causes dying for several neurons or the so-called dying ReLU problem. Several pieces of research have been proposed to remedy the ReLU problem. For example, the authors in [43] proposed a slight modification of ReLU by introducing the so-called ReLU-6 function. This modification is a restricted ReLU on the positive side; In particular, ReLU-6 bounds the rectified units into the value six. Also, ReLU-6 deactivates the negative values by mapping all negative inputs to zero and hence causes the dying problem as ReLU.

Another modification of ReLU has been proposed in [44]. The authors proposed the Leaky ReLU activation units (LReLU) to overcome the dying problem of ReLU. Instead of obtaining the dead neurons when the input $x < 0$, LReLU returns a small negative value using a small constant parameter a defined by (3). Thus, LReLU has a small negative slope for the negative region.

$$f(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \tag{3}$$

Another updated version of ReLU is the Parametric Rectified Linear Unit (PReLU) that has been introduced in [45].

PReLU generalizes the ReLU by adding a parameter a that controls the activation function's slope for negative inputs [45] aiming at learning the slope of the negative interval. The difference between LReLU and PReLU lies in the state of the parameter a, which is constant in LReLU, but it is adaptively learned in PReLU. In [25], it has been shown that PReLU may cause overfitting in the small datasets.

Randomized Leaky Rectified Linear units (RLReLU) [25] is another ReLU based activation function. It was first proposed and suggested by the winner of the Kaggle NDSB Competition [46]. RLReLU is defined by (4), where a is a random number sampled from a uniform distribution $\bigcup[l, u]$ to control the slope of the negative part. During the training stage, this parameter a takes a random value between an upper and lower bound of a uniform distribution. During the testing stage, the value of this parameter a is the average of those bounds.

$$f(x) = \begin{cases} x, & x \geq 0 \\ ax, & x < 0 \end{cases} \quad \text{where } a \sim \bigcup[l, u], l < u, \quad \text{and } l, u \in [0, 1] \tag{4}$$

Generally, although the works mentioned above of ReLU perform well in many neural networks [20], [25], there is a need for rigorous investigations on large-scale data to prove their superior performance.

3) NON-LINEAR BELOW FUNCTIONS

This category includes those activation functions which have linear and unbounded above units but not rectifier. Also, such functions map the negative inputs with negative values using non-linear functions. An example of such a category is the Exponential linear units (Elu) which was proposed in [47]. Elu is defined according to (5):

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha e^{x-1}, & x < 0 \end{cases} \tag{5}$$

In contrast to the Sigmoid family, ELU is a zero-centered activation function. The zero-centered property makes the optimization process easier since the weights' updates can be allowed to move in all possible directions (positive and negative), not only in a certain direction. ELU maps negative neurons with negative values, which can alleviate dead neurons and overcome the dying problem. Also, it has a positive constant parameter α that controls the slope of negative inputs. ELU becomes smooth slowly until its negative outputs are equal to α. The authors claim that this function pushes the mean unit closer to zero or centralizes the activation at zero, similar to the batch normalization, which speeds up the learning process [47]. However, because it contains an exponential operation, it may take a longer computational time.

The authors in [48] introduced a scaled exponential linear units(SeLU) as an activation function to perform the self-normalization without the need to normalizing the output of the activation function which required for self-normalizing neural networks (SNNs) [48]. SeLU is defined according to (6):

$$f(x) = \lambda \begin{cases} x, & x > 0 \\ \alpha e^{x-\alpha}, & x \leq 0 \end{cases} \tag{6}$$

where α and λ are two fixed parameters

Fig. 3 illustrates the behavior of SeLU, ReLU, and ELU which have the same identity function form for positive inputs. Self-normalizing networks work well with fixed configurations which makes SeLU less flexible. The authors in [48] determined two fixed parameters with value $\alpha \approx 1.6733$ and $\lambda \approx 1.0507$ to decide the best slope for both positive and negative parts.

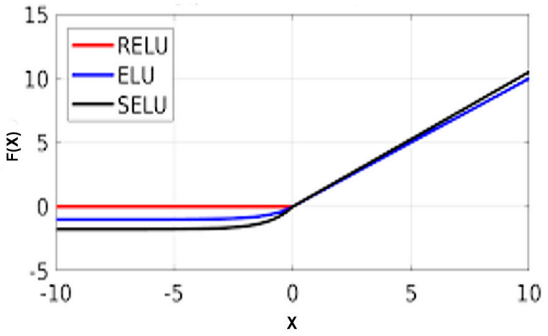

FIGURE 3. Behavior of ReLU, Elu and Selu activation functions. SeLU plotted for $\alpha = 1.6732$, $\lambda = 1.0507$ [49].

Similar to ELU, Brad Carlile *et al.* [50] proposed Inverse Square Root Linear Units (ISRLU) which is defined according to (7). ISRLU is a smooth and continuous first and second derivative. It has less computational complexity than ELUs and Tanh. The authors illustrate that ISRLU is four times faster than Tanh and twice as fast as Elu. See fig 4.

$$f(X) = \begin{cases} x, & x \geq 0 \\ \dfrac{x}{\sqrt{1 + \alpha x^2}}, & x < 0 \end{cases} \tag{7}$$

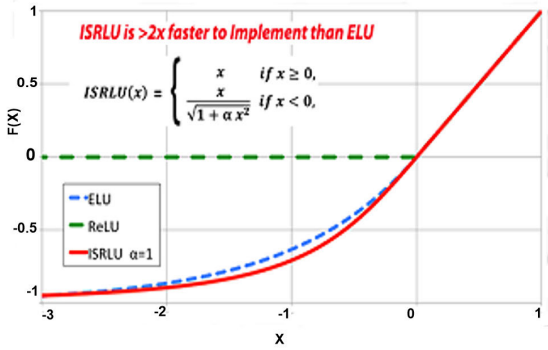

FIGURE 4. The inverse square root units [51].

Andrei Nicolae also introduced the Piecewise Linear Units (PLU) in [52] which belongs to this category. This

function has three linear segments with gradients α or 1, where α is a parameter to be chosen or trained. It is defined by (8):

$$PLU(x) \equiv max(\alpha(x+c)-c, \quad min(\alpha(x-c)+c, x)) \quad (8)$$

PLU is a hybrid of Tanh and ReLU, as shown in fig. 5. It outperforms the ReLU on some tasks and solves the vanishing gradients problem of the Tanh.

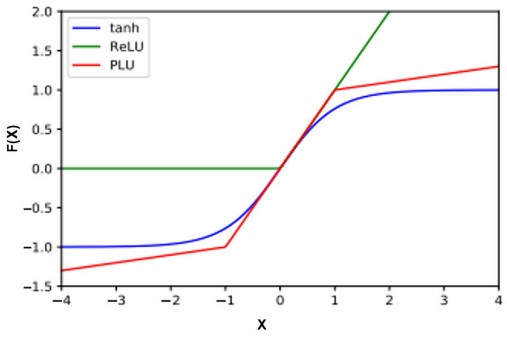

FIGURE 5. Plot of the tanh, ReLU, and PLU activation functions. For the PLU, $\alpha = 0.1$, $c = 1$ [52] .

4) NON-LINEAR AND UNBOUNDED ABOVE FUNCTIONS

In contrast to the previous categories' previous activation functions, this category contains those activation functions that do not have identity transformations. Hence, those functions may need a high computational cost, and voiding the exploding / vanishing problem is not guaranteed. An example of a function belonging to this category is SoftPlus [22], [53] which has been developed as a smooth version of Rectifier units. The first derivative of the SoftPlus function is the Sigmoid function, as shown in fig. 6. It produces outputs in the range from 0 to ∞ and is defined by (9):

$$f(x) = \log_2(1+e^x) \quad (9)$$

Also, the authors in [24] illustrated the significant role of SoftPlus compared with Sigmoid and ReLU to achieve fast convergence with fewer epochs during training. However, it is left soft saturated as Sigmoid.

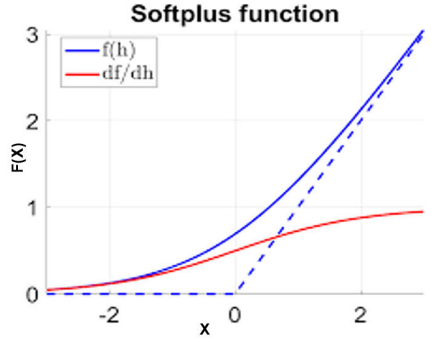

FIGURE 6. SoftPlus activation function [22].

Similar to the SoftPlus activation function, Noisy SoftPlus activation has been introduced in [54]. The Noisy SoftPlus enables spiking neural networks (SNNs) to be trained as ANNs.

Likewise the SoftPlus, the Gaussian Error Linear Units (GELU) is belonging to this category, and it was introduced in [55]. It outperformed ReLU and ELU across all considered computer vision applications, natural language processing (NLP), and speech recognition [55]. It is bounded below. GeLU is defined according to (10):

$$Gelu(x) = 0.5x(1+tanh(\frac{\sqrt{2}}{\pi}(x+0.044715x^3))) \quad (10)$$

Like Gelu, Sigmoid-Weighted Linear Units (SiLU) and its derivative function (dSiLU) are another types of activation functions belonging to this group. They have been introduced in [56] for reinforcement learning. SilU is defined by the Sigmoid function multiplied by its input as shown in (11):

$$a_k(s) = z_k\sigma(z_k) \quad (11)$$

Such function and its derivative outperformed the previous state-of-the-art results [56]. It a is non-linear monotonically increasing function as Sigmoid, but it is unbounded above. Unlike ReLU, the non-linearity in firing positive neurons may cause exploding gradient problem.

In [57], the authors developed a new activation function called a Self-Gated Activation Function (Swish) that works better on deeper architectures across a number of benchmarks [57]. It is defined by (12):

$$f(x;\beta) = 2x.\sigma(\beta x) \quad (12)$$

where $\sigma(x)$ is the logistic function. Swish is smooth, non-monotonic, unbounded above, and bounded below. In (12) with the parameter $\beta = 1$, it was shown that Swish activation function did not show any clear improvement over SiLU activation function.

The weighted Sigmoid Gate Units (WiG) has been introduced in [58]. This function is defined by element-wise product of the input and the Sigmoid activation with weighting parameter w_g and bias parameter b_g as shown in (13):

$$f(x) = x \cdot \sigma(xw_g + b_g) \quad (13)$$

WiG function achieved the best performance in specific tasks compared to the other popular activation functions. Such function becomes an identity function if the bias and weighting parameters have zero value [58].

Like WiG, the authors in [59] proposed the Exponential linear Squashing activation functions (ELiSH) and the HardELiSH. Its negative input part is ELU and Sigmoid's multiplication, while its positive part is like Swish. The latter function is a multiplication of HardSigmoid and ELU in negative part and HardSigmoid and Linear in positive part. Both ELiSH and HardELiSH are computationally more costly.

Both ELiSH and HardELiSH are defined according to (14) and (15) respectively.

$$ELiSH(X) = \begin{cases} \frac{x}{1+e^{-x}}, & x \geq 0 \\ \frac{e^{x}-1}{1+e^{-x}}, & x < 0 \end{cases} \quad (14)$$

$$HardELiSH(X) = \begin{cases} x * max(0, min(1, \frac{x+1}{2})), & x \geq 0 \\ e^{x}-1 * max(0, min(1, \frac{x+1}{2})), & x < 0 \end{cases} \quad (15)$$

A Self Regularized Non-Monotonic Neural Activation Function (Mish), has been introduced in [28]. It has an infinite order of continuity and is a non-monotonic function. Also it is unbounded above and bounded below, and approximates identity at origin. Although it doesn't cause saturation, its computational cost is higher than its former counterparts. Mish is defined according to (16):

$$f(x) = x * tanh(SoftPlus(x)) \quad (16)$$

5) INCREASING AND DECREASING FUNCTIONS

Yue Wu *et al.*, in [60] have proposed a Gaussian activation function as a special class of known radial basis functions (RBFs). It squashes its output into the range [0, 1]. It is characterized by its bell-shaped curve. Such function is non-monotonic since it rises and falls. This function is defined according to (17):

$$f(x) = e^{\frac{-x^2}{2}} \quad (17)$$

A sinusoid function is another activation function belonging to the increasing and decreasing units group that goes up and goes down. It is a non-monotonic and periodic function such as sine and cosine. It has been proposed as an activation function in [61] to fit the time-series data. It is defined according to (18):

$$f(x) = \sin x \quad (18)$$

Due to its shape's nature, it can be stuck in the local minimum, and the learning becomes very difficult. It does not represent a big issue when the data have a low frequency which is expected for many real-world datasets. Therefore Sinusoid contributes effectively in time series forecasting.

Truncated Sin is another activation function that belongs to this category. It was developed in [62]. The authors compared the results with the monotonic function Tanh using the same Trained networks. This function is defined by (19). According to the conducted experiments, the authors have demonstrated that on specific tasks, sinusoidal activation functions can potentially learn faster and better than the common monotonic functions [62].

$$tr.sin = \begin{cases} 0, & \text{if } -\pi/2 > x \\ sin(x), & \text{if } -\pi/2 \leq x \leq \pi/2 \\ 1, & \text{if } \pi/2 < x \end{cases} \quad (19)$$

B. SUMMARY OF ACTIVATION FUNCTIONS

In addition to the characteristics of the existing activation functions, this section summarizes the other criteria that are considered in the experimental evaluation of these functions. The aim is to establish a standard environment for comparing the empirical performances of activation functions. These criteria include the following: (1) type of machine learning, whether i.e., supervised or nonsupervised; (2) type and size of the benchmark datasets used in the evaluation; (3) problem tas; (4) type of deep neural architecture used in the evaluation; (5) variation of depth of the neural networks used if any; (6) different types of DNN architectures used if any; (7) whether or not the hyperparameters are fixed in all experiments; (8) validation techniques used, including cross or data split validation; (9) other activation functions used in the comparison; (10) whether or not the activation functions used in comparison, are fixed in each experiment; (11) the number of times the model runs during training.

Tables 1, 2, 3, and 4 summarize and illustrate the experimental features of the most popular activation functions. Symbol(-) means that the author did not refer to the criterion.

III. PROPOSED ACTIVATION FUNCTIONS

This section presents the proposed activation functions. It begins by explaining the intuitions behind the proposed activation functions. Then, the properties of each proposed function are described in detail.

A. METHODOLOGY OF PROPOSED FUNCTIONS

The study and analysis of the behavior and properties of the previous activation functions and their performance, one can make the following inferences: (1) The activation functions bounded above perform poorly relative to similar functions without such property, including ReLU, Elu, and LeakyReLU. (2) Boundedness is an undesirable property for any activation function, as it slows down the training time when gradients approach zero. (3) Successful activation functions are usually composed of two different subfunctions,namely, unbounded linear and nonlinear functions. Most unbounded linear functions are useful when they fire positive inputs. Linear functions are also preferable when firing positive inputs to avoid the exploding problem. On the contrary, nonlinear functions, which approach zero and have small slope approaching zero, are preferable for mapping negative inputs. In this setting, the nonlinear function acts as a regularizer as it discards small negative values. Hence, it improves network performance because it helps prevent the dying, saturation, and vanishing/exploding gradient problems.

TABLE 1. ELU, GELU, and SELU experimental criteria.

Criteria	Activation Functions		
	ELU [47]	**GELU** [60]	**SELU** [48]
Learning Type	Supervised Unsupervised	Supervised Unsupervised	Supervised
Benchmark	Gray images Color images	Gray images Color images Audio Text	Image Signal Text
	MNIST CIFAR10/100 ImageNet	MNIST CIFAR10 CIFAR 100 TIMIT Twitter Pos tagging	MNIST CIFAR10 HTRU2 datasetfor Astronomy 121 UCI benchmark datasets TOX21 challenge dataset
	6k training,10k testing 50 k training,10 k testing 1.3 M training , 100 k testing	-MNIST: 6k for training, 10k for testing -CIFAR10/100:50 k for training, 10 k for testing -TIMIT: 3696 training, 1152 validation,192 testing -Tweets:1000 for training, 327 for validation, 500 for testing	-MNIST: 6 k for training, 10 k for testing -CIFAR10: 50 k for training,10 k for testing -UCI :10 to 30.000 data points. -TOX21:12000 chemical compounds , each has 12 toxic effects
Task	Image classification	Image classification NLP Speech recognition	-Prediction of pulsars (signal detection) - Image classification -UCI tasks -Drug Discovery
Network Type	Elu networks: -CNN with 18 layer -Autoencoder Model	Deep CNN Autoencoder Shallow CNN Deep CNN Wide residual 40-4	Selu Networks: -Self-normalizing neural networks (SNN) -SNN-CNN (2x Conv,MaxPool, 2x fully connected) for MNIST experiments
Different depths of the model	YES	YES	YES
Different types of the network	YES	YES	YES
Fixing hyper-parameters in all experiments	- Different random initialization for MNIST evaluation - 4 different learning rate for autoencoder evaluation	With 3 different learning rates	NO
Validation Technique	-	- Cross validation for CIFAR 10/100 - Validation split for MNIST	10 Cross validation folds for HTRU2 dataset
The competitive activation functions	Elu networks compared with these networks: -AlexNet -DSN -NIN -Maxout -All-CNN -Highway -Fractional Maxpooling using:ReLU-Leaky , ReLU-Shifted Leaky ReLU activation functions	ReLU, Elu	-24 Machine learning techniques -Seven FNNs like : -FNNs with weight nornalization, -FNNs with layer normalization , -FNNs without normalization and with ReLU -CNN using ReLU -ResNet -Highway Network
Fixing the competitive activation functions in all experiments	NO	YES	YES
Times of running the model	- 5 runs for MNIST experiment -10 runs for autoencoder experiment - 10 runs for CIFAR10/100 with different weight initialization for each run	-3 runs for CIFAR10/100 and MNIST autoencoder -5 runs for the other datasets	5 times for Tox21 dataset

From the previous discussion, we can conclude that choosing an unbounded function for positive region is preferable to achieve better network performance. Meanwhile, a function with a small slope for the negative region is superior for firing negative values that approach zero, to allow gradients to flow on. Additionally, we need to reveal the function's performance, which can fire the negative inputs with small positive values.

Thus, to design a new activation function, it is preferable to combine linear and the nonlinear parts to form a single

TABLE 2. SoftPlus, SWISH, and SIN experimental criteria.

Criteria	Activation Functions		
	SoftPlus [24]	SWISH [57]	SIN [61]
Learning Type	Supervised	Supervised	Supervised
Benchmark	Acoustic	-Gray images -Color images	-Gray images - Text
	TIMIT	-MNIST -CIFAR10/100 -ImageNet -WMT 2014 (English-German translation)	MNIST Reuters
	- Train set: 3696 utterances by 462 speakers. - Validation set: 400 utterances by 50 speakers. - Test set: 192 utterances by 24 speakers.	MNIST: 6k training,10k testing CIFAR 10/100: 50 k training, 10 k testing ImageNet:1.3 M training , 100 k testing WMT:4.5 M training Sentences	MNIST: 6k training,10k testing Reuters: 11288 newswires labeled over 46 topics
Task	Phoneme recognition task.	Image classification Machine Translation	Image classification Text classification
Network Type	- DBN -Traditional RBM pre-training (DBNs are pretrained by stacked RBM) Revised RBM	-Fully connected with varying depths. -Residual (32-wrn) -Inception(V3-V4) -MobileNet -Mobile -NASNet-A -Dense100 -Inception-ResNet2 -Transformer Model	DNN RNN
Different depths of the model	YES	YES	YES
Different types of the network	NO	YES	YES
Fixing hyper-parameters in all experiments	NO	- Different Batch size using ResNet32 on CIFAR 10 for MNIST evaluation - 3 different learning rate for ImageNet experiments	NO
Validation Technique	-	-	-
The competitive activation functions	- Sigmoid - ReLU	-ReLU -PReLU -Selu -Elu -SoftPlus	- Tanh
Fixing the competitive activation functions in all experiments	YES	YES	YES
Times of running the model	-	- 3 runs for MNIST and ImageNet Experiments -5 runs for CIFAR10/100	-

TABLE 3. ReLU, LeakyReLU, and PReLU experimental criteria.

Criteria	Activation Functions		
	ReLU [41]	**LeakyReLU** [44]	**PReLU** [25]
Learning Type	- Supervised - Unsupervised - Semi Supervised		Supervised
Benchmark	-Gray images -Color images - Text	Acoustic	Color images
	- MNIST - CIFAR 10 - NORB - NISTP - Restaurant Reviews	Experiments on Switchboard conversational telephone speech corpus (LDC97S62)	ImageNet
	-MNIST :50k/10k/10k, 28 ×28 digit images, 10 classes. - CIFAR10 50k/5k/5k, 32 × 32 × 3 RGB images, 10 classes. -NISTP: 81,920k/80k/20k, 32 × 32 character images from the NIST database 19, with randomized distortions - NORB: 233,172/58,428/58,320, Stereo-pair images of toys on a cluttered background, 6 classes. -The restaurant review (site www.opentable.com) 10,000 labeled, 300,000 unlabeled training ,test set 10,000 .	Experiments on 300 hours. Evaluation on 25000 frame from training set	Image Net: 1.2 million training images, 50,000 validation images, and 100,000 test images.
Task	- Image classification - Sentiment Analysis	-large vocabulary continuous -speech recognition (LVCSR)	Image classification
Network Type	Stacked denoising auto-encoders	DNN-HMM	- MSRA - GoogLeNet - VGG-16 - Spatial pyramid pooling (Spp-Nets) - 3CNN Models with different depths\end{tabular}
Different depths of the model	YES	YES	YES
Different types of network	YES	NO	YES
Fixing hyper-parameters in all experiments	NO	NO	NO
Validation Technique	10-fold cross-validation	-	-
The competitive activation functions	- Tanh - SoftPlus	-Tanh -ReLU	-ReLU
Fixing the competitive activation functions in all experiments	NO	YES	YES
Times of running the model	-	-	-

TABLE 4. RReLU,WIG,ISRLU experimental criteria.

Criteria	Activation Functions		
	RReLU [25]	**WIG** [58]	**ISRLU** [50]
Learning Type	Supervised	Supervised	Supervised
Benchmark	-Color images -Gray images	-Gray images -Color images	Gray images
	-Cifar 10 -Cifar100 -Kaggle National Data Science Bowl Competition	- Cifar 10/100 - Collected images from : Urban 100, General l00, Yang 91	MNIST
	- Cifar10/100: 50 K training, 10 K testing - NDSB: 25 k training, 5336 testing in 121 classes	- Cifar 10/100: 50 k training, 10 k testing - Collected dataset: 80 k mini-batches for training, seven 512x512 sized images for testing	MNIST: 6k training, 10k testing
Task	Image classification task.	- Object recognition - Image denoising	Image classification
Network Type	- NIN - NDSB Net - Inception	-VGG like network - Denoising network	CNN
Different depths of the model	YES	NO	NO
Different types of the network	YES	YES	YES
Fixing hyper-parameters in all experiments	NO	NO	YES
Validation Technique	-	-	-
The competitive activation functions	- ReLU - LReLU - PLReLU	-ReLU -PReLU -Selu -Elu -SoftPlus -Leaky ReLU -Swish -Sil	- ReLU - Elu
Fixing the competitive activation functions in all experiments	YES	NO	NO
Times of running the model	-	-	-

function. For this purpose, we introduce two new activation functions. Both functions share linearity in the positive region, but they differ when handling nonlinearity in the negative region.

Both of the proposed activation functions have a hyper-parameter (α) that can be adjusted automatically during the learning process. This process is called hyper-parameter optimization or tuning, which is used to find the best hyper-parameter value for a given dataset. Several methods, such as Bayesian optimization, evolutionary algorithms, any search technique like random or grid search technique, are used to adjust any hyper-parameter during learning [63], [64]. In this work, we manually tuned α like the other hyper-parameters.

B. PROPOSED ACTIVATION FUNCTIONS

In this study, we suggest two nonlinear activation functions. The first one is the **IpLU,** which is mathematically expressed as (20):

$$f(x) = \begin{cases} x, & x \geq 0 \\ \dfrac{x}{1+ \mid x \mid^{\alpha}}, & x < 0 \end{cases} \tag{20}$$

where α is a positive hyper-parameter greater than 1 that guarantees a small slope for negative interval.

We represent the negative interval in a polynomial form such that the denominator $(1+ \mid x \mid^{\alpha})$ with α aims at obtaining a small slope for the negative region. Simultaneously, the function is also unbounded above to activate positive inputs, and thereby avoid a vanishing gradient. We select the linear function for firing positive neurons to avoid the exploding gradient problem, which occurs when large error gradients accumulate. Fig. 7 plots the graph of the proposed activation function with values of $\alpha = 1, 2, 3, 1.4$ respectively.

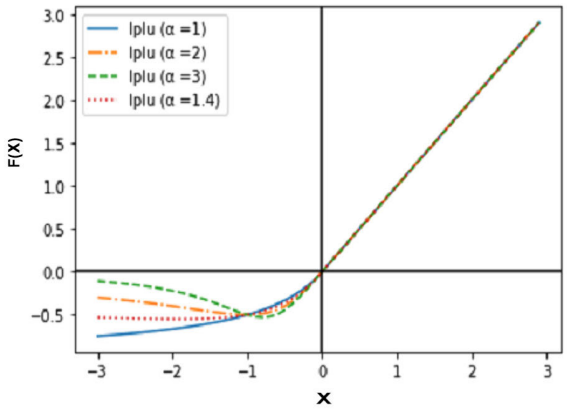

FIGURE 7. The proposed IpLU.

The proposed IpLU is zero-centered, which means that it produces negative outputs that optimize the network cost by shifting weights and biases in the right direction. It is also a differentiable function, as shown in (21). Table 5 summarizes the characteristics of the IpLU with C^n representing the n^{th} order of continuity. IpLU and its first derivative are continuous only if $\alpha = 1$. Therefore its order of continuity is C^1. Additionally, it is non-monotonic except when $\alpha = 1$. Whenever the value of α increases, the activated negative values approach zero. Hence, large values should not be assigned to α to prevent the large occurrence of dead neurons. Fig. 8 illustrates the differentiation of this function.

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ \dfrac{1+ \mid x \mid^{\alpha} -x^2\alpha \mid x \mid^{\alpha-2}}{(1+ \mid x \mid^{\alpha})^2}, & x < 0 \end{cases} \tag{21}$$

TABLE 5. Properties of proposed IpLU.

Monotonic	Yes only if $\alpha = 1$
Zero-centered	Yes
Monotonic Derivative	Yes only if $\alpha = 1$
Order of continuity	C^1 when $\alpha = 1$ otherwise C^0

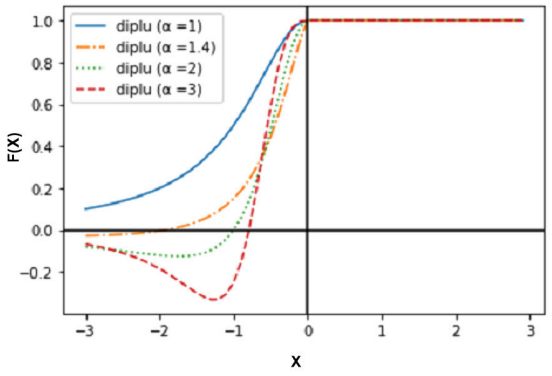

FIGURE 8. The differentiation of proposed IpLU with $\alpha = 1, 1.4, 2, 3$.

The second proposed activation function is the **AbsLU**. This function is similar to Leaky ReLU, but it activates the negative inputs with positive sign values. This function is non-zero-centered, which means that the gradient always has the same sign. It might thus inappropriate for gradient-based optimization algorithms because all weights' updates can only move in one direction. Therefore, applying this function requires inputs to be normalized in advance. The normalization can help weight updates to move in all possible directions. Table 6 summarizes the properties of AbsLU. The proposed function and its first derivative are defined according to (22), and (23) respectively.

$$f(x) = \begin{cases} x, & x \geq 0 \\ \alpha. \mid x \mid, & x < 0 \end{cases} \tag{22}$$

where α is hyper-parameter in range [0:1]

$$f'(x) = \begin{cases} 1, & x \geq 0 \\ \alpha, & x < 0 \end{cases} \tag{23}$$

TABLE 6. Properties of proposed AbsLU.

Monotonic	No
Zero-centered	No
Monotonic Derivative	No
Order of continuity	C^0

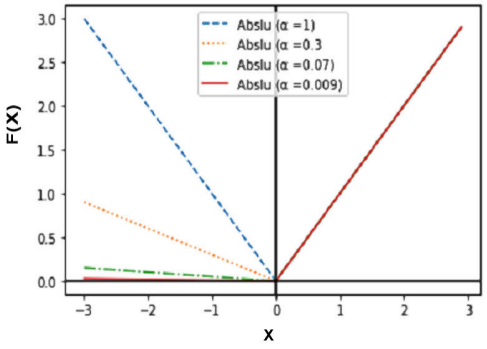

FIGURE 9. Proposed AbsLU with $\alpha = 1, 0.3, 0.07.0.009$ respectively.

Fig. 9 illustrates AbsLU function with different value of α.

The proposed AbsLU is a non-monotonic activation. Similar to ReLU, its derivative is not continuous.

Fig. 10 shows the derivative of the AbsLU function.

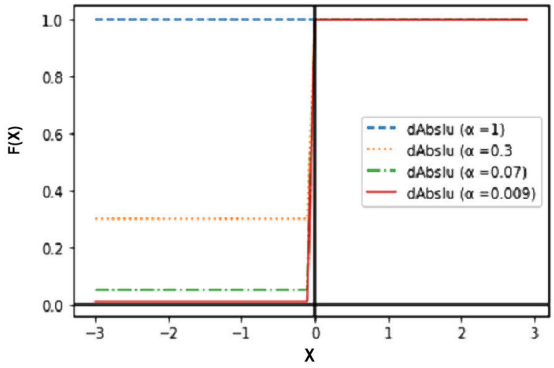

FIGURE 10. The differentiation of proposed AbsLU with $\alpha = 1$.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

To evaluate the performance of the proposed activation units, we consider several criteria, including the different types and depths of networks, as well as the classification tasks. In particular, we use several DNN architectures; convolutional neural network (CNN) [65] with different layers, AlexNet [66], VGG16 and VGG19 [67], ResNet56 V1 and ResNet56 V2 [68], ResNet110 [69], ResNet-wrn 22-10 [70], Inception V3 [71], DenseNet100 [72], bidirectional LSTM [73] as a type of Recurrent neural network (RNN); one dimensional convolutional network (1D-CNN) [74], and fully connected network (FCNN). Additionally, we consider two categories of classification tasks, namely image and text classification. For both tasks, we use several popular benchmarks to conduct a wide range of experiments.

The conducted experiments are divided into two categories on the basis of the classification task. The first category evaluates the performance of the functions in image classification tasks by considering the type and depth of the network architecture. This category of experiments are applied on several image benchmarks, including MNIST [75], Fashion-MNIST [76], Cifar10 and Cifar100 [77], Aptos Blindness Detection [78], and COVID-19 [79]. Given the limitations of the image size and image type of the MNIST and Fashion-MNIST benchmark datasets, we cannot use them in our experiments involving the ResNet-wrn 22-10 network [70]; in particular, such images require deep neural architectures with a fewer hidden layers to fit their type and size. Also, the structure of Inception V3 [71] network requires colored images for training, and Aptos Blindness Detection is a large dataset. Faced with the resources limitation, we experiment with the latter by using all image classification architectures, except Inception-V3 and ResNet-wrn 22-10. The second category of our experiments evaluates the performance on text classification tasks using several types of network architectures and datasets, namely, Reuters Newswire [80], and IMDB [81] benchmark datasets.

This section begins by describing the benchmark datasets used in the experiments. Then, the two categories of experiments are discussed in detail. We also summarize the performance of proposed activation functions in both categories. Fig. 11 presents a summary of all types of the DNNs used in the experiments conducted.

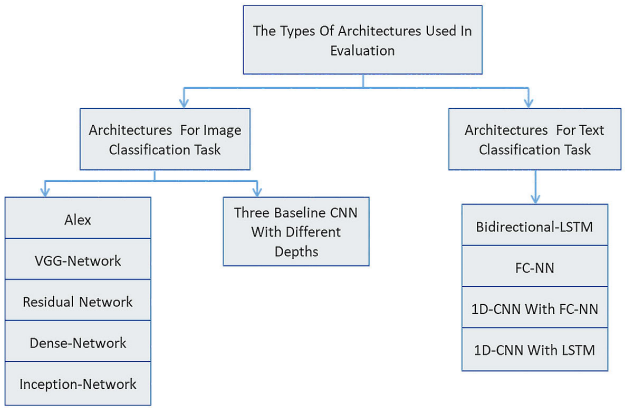

FIGURE 11. Categories of deep networks architectures used in the experiments.

The proposed activation functions are implemented with Keras [82], a high-level neural network API written in Python. Keras is also an open-source neural network library. The code is written using online Jupyter notebooks hosted by Colab [83], and Kaggle [84]. A total of 90 experiments are conducted on the text classification, and each experiment is conducted on five different splits per dataset. In the image classification task,594 experiments are conducted, with each experiment conducted in five runs. Thus, the total number of conducted experiments is **3,420**.

A. DATA DESCRIPTION

The benchmark datasets used in the experiments are categorized into two groups: image and text classification. Table 7 describes the properties of the benchmark datasets used in our experiments.

TABLE 7. Description of benchmarks datasets.

Benchmark	Properties			
	Task	Size	Data Type	#of Classes
MNIST [75]	Image Classification	70,000 handwritten digits with 28×28	Grayscale images	10
Fashion MNIST [76]	Image Classification	70,000 clothes images with 28×28	Grayscale images	10
Cifar10 [77]	Image Classification	60000 images with 32x32	RGB images	10
Cifar100 [77]	Image Classification	60000 images with 32x32	RGB images	100
COVID-19 [79]	Image Classification	10000 lung images	CT Scans (RGB images)	3
Aptos2019 Blindness Detection [78]	Image Classification	5593 retina images (9.52 GB)	Fundus-photography (RGB images)	5
Reuters Rewswire [80]	Text Classification	11,228 newswires	Topics	46
IMDB Review [81]	Text Classification	25,000 movies review	Sentiment reviews	2

The first group of datasets includes MNIST, Fashion-MNIST, Cifar10, Cifar100, and medical images from "Aptos Blindness Detection" and "COVID-19" datasets. Both MNIST handwritten and Fashion-MNIST benchmark datasets contain $70,000$ grayscale examples with 28×28 pixel size for classifying 10 types of classes. Both Cifar10 and Cifar100 consist of $60,000$ colored images with 32×32 pixel size distributed in 10 and 100 classes. Moreover, the first group includes two types of medical image datasets. The first type is the Aptos Blindness Detection dataset consisting of 5,593 retina images for classifying the severity of diabetic retinopathy on a scale of 0 to 3. The second type of medical image is the COVID-19 image dataset which contains $10,000$ lung images collected to detect Pneumonia and COVID-19 patient.

The second group of datasets includes benchmark datasets dedicated to text classification tasks. In particular, this group contains IMDB movie reviews and Reuters newswire topics. The IMDB dataset contains $25,000$ movie reviews labeled on two sentiment classes, namely, positive and negative. Reuters newswire, on the other hand, consists of $11,228$ newswires labeled on 46 topics.

B. EXPERIMENTS ON IMAGE CLASSIFICATION TASK

To conduct this group of experiments on the image benchmark datasets, we further divide this group into two subgroups with consideration of the type and depth of the network. Thus, the first subgroup evaluates the proposed activation functions' performance by fixing the network architecture but varying its depth. We use CNN architecture on three variants of depths. The second subgroup evaluates the proposed activation functions on the basis of different types of popular deep architectures.

1) EXPERIMENTS USING DIFFERENT DEPTHS OF CNN ARCHITECTURE

In this set of experiments, we use three simple CNN architectures, namely, $Model_1$, $Model_2$, and $Model_3$, but with different depths. We compare the performance of proposed activation functions against those of seven of the most successful popular activation functions. The $Model_1$ architecture is stacked by two convolution layers, one fully connected layer and one softmax classification layer. The first convolutional layer convolves the input with 32 filters with a size of (3×3) with feature-maps passed through a batch normalization layer; then, the activation function is applied. The Max pooling method is applied with a stride of 2, followed by a dropout method that attempts to prevent overfitting. The second convolutional layer convolves the previous layer's outputs with 64 filters with a size of (3×3), followed by the same techniques in the first layer. Then, a flattened convolutional layer outputs to pass through a fully connected layer.

Similarly, the structure of $Model_2$ is stacked in the same way of $Model1$, but with additional two extra convolution layers. The first additional convolutional layer uses 128 filters with a size of (3×3). The last convolutional layer uses 256 filters of the same size. As for $Model3$, it is deeper with overall six convolution layers with additional two convolutional layers having 512 and 1024 filters with a size of (3×3), respectively.

In all experiments, the average performance of five runs on the data is considered. All hyperparameters are fixed with the Adam optimizer, learning rate (1e-4), number of epochs $= 20$, and batch size $= 32$.

In all the following tables, the values in the bold style represents the highest accuracy achieved.

The proposed AbsLU and IpLU with different α values are evaluated against the other popular activation functions on the Cifar10, Cifar100, MNIST, Fashion-MNIST, Aptos Blindness Detection, and COVID-19 datasets by using the aforementioned CNN models.

The full accuracy results on the Cifar100, Cifar10, Fashion-MNIST, MNIST are presented in Tables 8, 9, 10, and 11 respectively, which compare some of the well-known activation functions with the proposed activation functions. The full results of experiments on the medical images are shown in Tables 12 and 13. In all the results, the last column of each table reflects the average accuracy for each activation function at the level of all three CNN models.

TABLE 8. Test accuracy results on Cifar 100.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	42.4%	55.8%	57.7%	51.97%
Selu	41.4%	51.4%	54.6%	49.13%
ReLU	43.3%	58.6%	56.4%	52.77%
LeakyReLU	43.1%	57.8%	59.3%	53.4%
Gelu	42.9%	**58.8%**	59.8%	53.83%
Swish	43%	58.5%	59.6%	53.7%
Mish	38.3%	46.8%	48.9%	44.67%
IpLU	43.7%	58.7%	59.1%	53.83%
AbsLU	**44.7%**	**58.8%**	**59.9%**	**54.47%**

TABLE 9. Test accuracy results on Cifar 10.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	41.9%	82.8%	82.1%	68.93%
Selu	41.8%	80.7%	80.1%	67.53%
ReLU	43.4%	85.6%	85.4%	71.47%
LeakyReLU	42.3%	84%	86.1%	70.8%
Gelu	43.4%	85.4%	86.1%	71.63%
Swish	42.4%	85.4%	85.9%	71.23%
Mish	38.7%	75.7%	73.5%	62.63
IpLU	44.2%	85.2%	85.9%	71.77%
AbsLU	**44.70%**	**86.15%**	**86.67%**	**72.51%**

The results on Cifar100 indicate that AbsLU achieves the best accuracy in $Model_1$ and $Model_3$. Gelu and AbsLU achieve the best accuracy in $Model_2$. The proposed activation function IpLU achieves the second-best accuracy in $Model_1$ and $Model_2$. The average accuracy for all three models reveals that AbsLU ranks first among the popular activation functions, using the three models.

On Cifar10, AbsLU also achieves the best accuracy in all three models, whereas IpLU ranks second, fifth, and fourth in $Model_1$, $Model_2$, and $Model_3$ respectively. However, all models' average accuracy indicates that AbsLU and IpLU rank first and second, respectively. For Cifar10 and Cifar100, the used α values of AbsLU and IpLU are 0.01 and 2, respectively. Fig. 12 represents the superiority of AbsLU over LeakyReLU on Cifar10 using $Model_2$.

Similarly, all models' average accuracy on Fashion-MNIST and MINIST in Tables 10 and 11,respectively, shows that AbsLU achieves the highest accuracy. IpLU ranks third.

To train $Model_1$, $Model_2$,and $Model_3$ on Aptos and COVID-19, there are different α values have been experimented and the best have been chosen. The selected α values for IpLU are 1.5, 2.2, and 1 respectively. Those for AbsLU are 1e-2 and 1e-3 for $Model_1$ and $Model_2$ respectively, and 1e-1 for $Model_3$. While, the selected α value for IpLU is 1 to

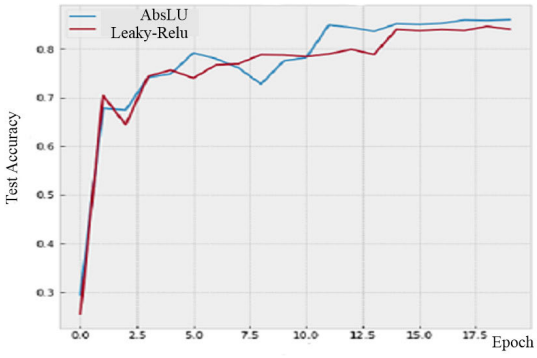

FIGURE 12. Test accuracy for AbsLU against LeakyReLU when $\alpha = 0.01$ on Cifar10 using $Model_2$.

TABLE 10. Test accuracy results on Fahion-MNIST.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	92.9%	92.9%	93.37%	93.06 %
Selu	92.1%	93.1%	92.91%	92.70%
ReLU	93.5%	93.9%	94.21%	93.87%
LeakyReLU	93%	92.9%	93.72%	93.21%
Gelu	93.2%	93.9%	**94.30%**	93.8%
Swish	93.3%	93.7%	94.16%	93.72%
Mish	91.2%	91.6%	91.59%	91.46
IpLU	93.4%	93.8%	94.21%	93.80%
AbsLU	**93.51%**	**94.17%**	94.26%	**93.98%**

TABLE 11. Test accuracy results on MNIST.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	99.1%	99.5%	99.39%	99.33%
Selu	99%	99.4%	99.28%	99.23%
ReLU	99.3%	**99.6%**	99.50%	99.47%
LeakyReLU	99%	99.5%	99.45%	99.32%
Gelu	**99.4%**	99.5%	99.49%	99.46%
Swish	99.3%	99.5%	**99.52%**	99.44%
Mish	98.5%	99.3%	99.11%	98.97%
IpLU	99.3%	**99.6%**	99.48%	99.46%
AbsLU	**99.4%**	**99.6%**	**99.52%**	**99.51%**

train the three models on MNIST and Fashion-MNIST. Also, the selected α value for AbsLU is 0.1.

The experimental results on the medical images indicate that IpLU achieves the best accuracy in $Model_1$ and $Model_3$ and is ranked second in $Model_2$ in the case of the Aptos Blindness Detection classification. For the same benchmark dataset, AbsLU ranks second in $Model_1$ and $Model_3$, but it ranks third in $Model_2$. As for the COVID-19 dataset, the results indicate that IpLU ranks first in $Model_3$ and ranks second in both $Model_1$ and $Model_2$. Whereas AbsLU

TABLE 12. Test accuracy of Aptos blindness detection with the proposed activation functions against the most superior activation functions using different depths of CNN model.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	49.05%	64.55%	71.7%	61.77%
Selu	48.07%	**68.83%**	54.4%	57.1%
gelu	35.2%	49.31%	73%	52.50%
LeakyReLU	37.96 %	61.71%	72.9%	57.52%
Swish	35.16%	57.93%	72.5%	55.20%
Mish	46.76%	55.6%	69.9%	57.42%
ReLU	43.49%	63.46%	70.8%	59.25%
IpLU	**53.16%**	68.44%	**73.8%**	**65.13%**
AbsLU	49.64%	67.49%	73.6%	63.58%

TABLE 13. Test accuracy of COVID-19 with the proposed activation functions against the most Superior activation functions using different depths of CNN model.

AFs \ Model	$Model_1$	$Model_2$	$Model_3$	Accuracy Average
Elu	83.80%	81.19%	70.1%	78.36%
Selu	83.13%	74.61%	51.2%	69.65%
gelu	80.77%	86.58%	63.1%	76.82%
LeakyReLU	82.08 %	84.34%	60.5%	75.64%
Swish	84.06%	83.03%	70.7%	79.26%
Mish	**85.11%**	87.11%	55.4%	75.87%
ReLU	80.90%	87.63%	74.5%	81.01%
IpLU	85.10%	87.76%	**76.3%**	**83.05%**
AbsLU	84.84%	**89.08%**	74.1%	82.67%

ranks first in $Model_2$ and third in $Model_1$ and $Model_3$. The average accuracy of all three models on the medical images indicates that AbsLU and IpLU rank first and second, respectively.

2) EXPERIMENTS USING POPULAR DEEP ARCHITECTURES

We also experiment with the activation functions on different models of popular deep architectures. First, the image benchmarks are tested on different depths of residual networks. In particular, ResNet56V1, ResNet56V2, and ResNet110 are used to classify MNIST, Fashion-MNIST, Cifar10, Cifar100, COVID-19, and Aptos Blindness Detection. Additionally, ResNet-wrn-22 is used to classify Cifar10, Cifar100, and COVID-19. Second, the image benchmarks are tested on other models of common architectures. In particular, Alex, Vgg16, Vgg19, and DenseNet-100 are used to train MNIST, Fashion-MNIST, Cifar10, Cifar100, COVID-19, and Aptos Blindness Detection. Furthermore, Inception-V3 is used to train Cifar10, Cifar100, and COVID-19. The full results of the experiments are depicted in Tables 14, 15, 16, 17, 18, and 19. The last column of each table reflects the average accuracy

TABLE 14. Test accuracy results on Cifar10 using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	ResNet wrn 22-10	Accuracy Average
Elu	75.4%	78.5%	78.5%	71%	75.85%
Selu	68.9%	74.1%	74.8%	74.6%	73.1%
ReLU	74.8%	71.6%	81.2%	87.1 %	78.68%
LeakyReLU	76.5%	69.6%	82.7%	86.6%	78.85%
Gelu	74.8%	79.8%	84.2%	82.1%	80.23%
Swish	76.6%	79.7%	80.1%	78.8%	78.8%
Mish	38.3%	70.4%	10%	54.7%	43.35%
IpLU	**78.3%**	**80.9%**	**84.8%**	80.1%	81.03%
AbsLU	77.5%	76.1%	82.5%	**88.1%**	**81.05%**

TABLE 15. Test accuracy results on Cifar 100 using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	ResNet wrn 22-10	Accuracy Average
Elu	42.5%	44.4%	52.6%	39.5%	44.75%
Selu	38%	35.3%	50.4%	45%	42.18%
ReLU	39.8%	39.1%	55.3%	59.5%	48.43%
LeakyReLU	40.0 %	42.6 %	53.1 %	59.6%	48.83%
Gelu	40.7%	43%	52.4%	53.2%	47.33%
Swish	**45.2%**	**47.7%**	55.3%	46.5%	47.93%
Mish	24.4%	27.9%	1%	26.6%	19.98%
IpLU	44.5%	**47.7%**	53.8%	43.2%	46.55%
AbsLU	42.5%	42.3 %	**55.5%**	**60.7%**	**50.25%**

TABLE 16. Test accuracy results on MNIST using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	Accuracy Average
Elu	97.5%	98.4%	98.79%	98.23%
Selu	97.4%	97.2%	98.78%	97.79%
ReLU	97.5%	97.5%	98.81%	97.94%
LeakyReLU	97.8%	98%	98.90%	98.23%
Gelu	98.2%	98.4%	98.72%	98.44%
Swish	98.4%	98.1%	98.46 %	98.32%
Mish	91.5%	95.7%	9%	65.4%
IpLU	**98.7%**	**98.6%**	**98.97%**	**98.76%**
AbsLU	98%	98.4%	98.91%	98.44%

for each activation function, at the level of all residual models used for each benchmark dataset.

In this type of experiments, different values of α are tuned. For example, $\alpha = 1e-2$ is adopted for AbsLU to train ResNet110 on MNIST and Fashion-MNIST datasets. Also, $\alpha = \{1.5, 3.3, 1, 3.2\}$ are used for IpLU to train ResNet110 on Mnist, Fashion-Mnist, Cifar10, Cifar100, and Aptos Blindness Detection datasets respectively. IpLU with $\alpha = 2$ is used to train ResNet56V2 and ResNet110 on

TABLE 17. Test accuracy results on Fashion-Mnist using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	Accuracy Average
Elu	88.6 %	89.9 %	**90.81%**	89.77%
Selu	87.8 %	89.6%	90.70%	89.37%
ReLU	88.9%	90.1%	90.39%	89.79%
LeakyReLU	89%	90.6%	90.31%	89.97%
Gelu	89.9%	90.7%	90.58%	90.39%
Swish	89.8%	**91.1%**	90.5%	90.47%
Mish	79.4%	84.5%	10%	57.97%
IpLU	89.7%	**91.1%**	90.79%	**90.53%**
AbsLU	**90.1%**	90.8%	90.66%	90.28%

TABLE 18. Test accuracy results on Aptos blindness detection competition using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	Accuracy Average
Elu	73.45%	73.78%	69.97%	72.4%
Selu	72.73%	73.27%	69.16	71.72%
ReLU	**74.40%**	73.2%	70.94%	72.85%
LeakyReLU	73.46%	**74.47%**	68.62%	72.18%
Gelu	73.85%	73.09%	70.47%	72.47%
Swish	74.04%	73.38%	71.31%	72.91%
Mish	71.16%	73.31 %	58.43%	67.63%
IpLU	74.36%	74.26%	71.13%	73.25%
AbsLU	74.22%	73.89%	**71.93%**	**73.35%**

TABLE 19. Test accuracy results on COVID-19 dataset using different residual models.

AFs \ Model	ResNet 56V1	ResNet 56V2	ResNet 110	ResNet wrn 22-10	Accuracy Average
Elu	76.29%	82.27%	80.66%	66.59%	76.45%
Selu	72.21%	80.48%	70.26%	80.12%	75.77%
ReLU	82.51%	81.20%	78.16%	82.27%	80.60%
LeakyReLU	81.32%	80.24%	80.88%	81.68%	81.02%
Gelu	83.35%	81.44%	80.18%	68.86%	78.46%
Swish	78.68%	80.24%	79.02%	72.46%	77.6%
Mish	69.22%	72.93%	37.05%	67.91%	61.78%
IpLU	80.72%	**84.19%**	**82.31%**	72.34%	79.89%
AbsLU	**84.07%**	80.12%	79.93%	**87.18%**	**82.83%**

MNIST and COVID-19 respectively. Additionally, IpLU with $\alpha = 2$ is used to train ResNet-wrn-22 on Cifar100, while $\alpha = \{1, 1.5\}$ are the best values to train such network on Cifar10 and COVID-19 respectively.

The experimental result of IpLU ranks first on Cifar10 with ResNet56V1, ResNet56V2, and ResNet110, but it is ranks fifth with ResNet-wrn. On the same benchmark, AbsLU

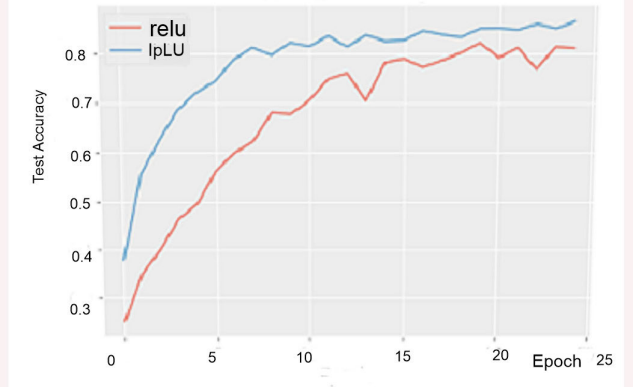

FIGURE 13. Test accuracy for IpLU against ReLU when $\alpha = 2$ on Cifar10 using Inception V3.

TABLE 20. Test accuracy results on Cifar 10 using different popular networks.

AF \ Model	Alex	Vgg16	Vgg19	DenseNet 100	Inception V3	Accuracy Average
Elu	57.9%	82.7%	82.2%	82.1%	85.5%	77.76%
Selu	56.2%	80.1%	77.5%	76.8%	10%	60.12%
ReLU	59.4%	80.8%	80%	78.1%	81.2%	75.9%
LeakyReLU	61%	**83.6%**	**82.7%**	81.1%	82.5%	78.18%
Gelu	59.6%	75.8%	74.2%	81.5%	77.2%	73.66%
Swish	59.6%	75 %	74.1%	**83.9%**	61.3%	70.78%
Mish	56%	79.5%	80.5%	69.6%	10%	59.12%
IpLU	60.2%	82.6%	81.8%	82.9%	**86%**	**78.7%**
AbsLU	**61.9%**	80.8%	81.13%	78.77%	80.5%	76.62%

TABLE 21. Test accuracy results on Cifar 100 using different popular networks.

AFs \ Model	Alex	Vgg16	Vgg19	DenseNet 100	Inception V3	Accuracy Average
Elu	40%	53.9%	45.5%	49.4%	**58.5%**	49.46%
Selu	29%	39.9%	46.7%	43.2%	1%	31.96%
ReLU	44.7%	36.6%	35.8%	45.9%	43.9%	41.38%
LeakyReLU	42.6%	46.04%	47.2%	49.1%	51.9%	47.37%
Gelu	42.6%	36.2%	43.7%	50.2%	1%	34.74%
Swish	44.3%	35.3%	30.4%	**52.9%**	1%	32.78%
Mish	10%	41.7%	49%	35%	51.6%	37.46%
IpLU	**47.9%**	**54.4%**	**52.3%**	51.4%	57.8%	**52.76%**
AbsLU	44.86%	38.1%	36.5%	45.6%	44.1%	41.83%

ranks second and fifth with ResNet56V1 and ResNet56V2, respectively, but it ranks fourth and first with ResNet110 and ResNet-wrn respectively. Generally, all residual architectures' average accuracy indicates that AbsLU and IpLU respectively rank first and second on Cifar10.

On Cifar100, IpLU ranks second, first, fourth, and sixth with ResNet56V1, ResNet56V2, ResNet110, and ResNet-wrn, respectively. AbsLU ranks third, sixth, and first with ResNet56V1, ResNet56V2, and ResNet110, respectively. Also, AbsLU ranks first with ResNet-wrn. Even though AbsLU is not ranked first on all previous models, its average

TABLE 22. Test accuracy results on MNIST using different popular networks.

Model / AFs	Alex	Vgg16	Vgg19	DenseNet 100	Accuracy Average
Elu	99.38%	99.09%	98.91%	98.53%	98.98%
Selu	99.30%	99%	99.03%	98.34%	98.92%
ReLU	**99.43%**	99.30%	11.35%	98.55%	77.16%
LeakyReLU	99.32%	99.31%	99.35%	98.68%	99.17%
Gelu	99.41%	11.35%	11.35%	98.68%	55.20%
Swish	99.39%	11.35%	11.35%	98.79%	55.22%
Mish	9.8%	99.08%	99.01%	94.16%	75.51%
IpLU	99.40%	99.04%	98.94%	**98.80%**	99.05%
AbsLU	**99.43%**	**99.33%**	**99.36%**	98.64%	**99.19%**

TABLE 23. Test accuracy results on Fashion-MNIST using different popular networks.

Model / AFs	Alex	Vgg16	Vgg19	DenseNet 100	Accuracy Average
Elu	92.9%	90.3%	90.66%	91.33%	91.30%
Selu	92.5%	90.78%	89.53%	89.39%	90.55%
ReLU	93.06%	92.3%	92.35%	90.76%	92.12%
LeakyReLU	92.45%	92.23%	92.15%	91.4%	92.06%
Gelu	92.9%	10%	10%	**92.27%**	51.29%
Swish	93.02%	10%	10%	91.63%	51.16%
Mish	10%	90.5%	91.03%	80.45%	68%
IpLU	93.1%	91.14%	90.97%	91.84%	91.76%
AbsLU	**93.45%**	**92.50%**	**92.41%**	91.66%	**92.51%**

accuracy indicates its first-place ranking while IpLU ranks sixth.

The average accuracy of all residual architectures on MNIST (Table 16) and Fashion-MNIST (Table 17), shows variations in the rank of IpLU and AbsLU. The average accuracy indicates that IpLU achieves the best rank on MNIST and Fashion-MNIST, whereas AbsLU achieves the second and fourth on MNIST and Fashion-MNIST. The average accuracy on Aptos Blindness Detection (Table 18) and COVID-19 (Table 19) shows that AbsLU is the best rank, and that IpLU ranks second and fourth respectively.

Further experiments are conducted with Alex, Vgg16, Vgg19, and DenseNet100 networks using all image benchmarks. Inception V3 model only applied to Cifar10, Cifar100, and COVID-19. Fig. 13 illustrates the proposed IpLU against ReLU in terms of the results on Cifar10 using Inception V3. The results are summarized in Tables 20, 21, 22,23, 24, and 25.

Despite the diversity of the rankings of IpLU and AbsLU in the models, their results on Cifar10, (Table 20) indicate that the average accuracies of IpLU and AbsLU respectively rank first and fourth for all models.

TABLE 24. Test accuracy results on COVID-19 dataset using different popular networks.

Model / AFs	Alex	Vgg16	Vgg19	DenseNet 100	Inception V3	Accuracy Average
Elu	76%	76.82%	73.61%	78.25%	76.38%	76.21%
Selu	77.37%	77.22%	73.09%	77.62%	**76.75%**	76.41%
ReLU	73%	74.57%	71.2%	81.7%	71.13%	76.32%
LeakyReLU	76%	76.73%	72.69%	71.25%	71.63%	73.66%
Gelu	75.5%	75.76%	58.74%	74.99%	76.50%	72.30%
Swish	76.37%	76.95%	50.52%	*81.62%*	76.50%	72.39%
Mish	81.13%	80.13%	73.31%	70%	71.75%	75.26%
IpLU	**82.23%**	**81.99%**	**78.43%**	**82.87%**	76.25%	**80.35%**
AbsLU	75.76%	78.01%	69.32%	82.75%	76.25%	76.42%

TABLE 25. Test accuracy results on Aptos blindness detection competition using different popular networks.

Model / AFs	Alex	Vgg16	Vgg19	DenseNet 100	Accuracy Average
Elu	74.36%	72.07%	72.73%	71.38%	72.64%
Selu	73.13%	73.05%	71.67%	70.84%	72.17%
ReLU	72.58%	72.51%	73.02%	64.76%	70.72%
LeakyReLU	73.49%	73.09%	72.84%	69.49%	72.23%
Gelu	71.93%	72.69%	62.76%	68.58%	68.99%
Swish	72.29%	71.56%	71.85%	66.25%	70.49%
Mish	73.2%	73.13%	70.69%	70.18%	71.8%
IpLU	**75.11%**	72.37%	72.73%	**72.33%**	**73.14%**
AbsLU	73.2%	**73.93%**	**73.64%**	70.4%	72.79%

Similarly, the average accuracy of all different deep architectures depicted in Table 21 indicates that IpLU achieves the best rank on Cifar100, and that AbsLU ranks fourth. Meanwhile, AbsLU achieves the first rank on MNIST and Fashion-MNIST as shown in Tables 22 and 23, respectively. IpLU achieves the best average accuracy on COVID-19 and Aptos Blindness Detection as depicted in Tables 24 and 25, respectively; for the same benchmarks, AbsLU ranks second.

C. EXPERIMENTS FOR TEXT CLASSIFICATION TASK

This category of experiments evaluates the accuracy of proposed activation functions in text classification task. All experiments are performed on Reuters and IMDB-Movie benchmarks. We experiment with different models by using three baseline architectures. The first architecture is the RNN with two different models called **Model_A1** and **Model_A2**. **Model_A1** consists of an embedding layer followed by one bidirectional LSTM layer with 32 units and one fully connected layer followed by a softmax layer. **Model_A2** is similar to **Model_A1** in terms of structure, but it has two bidirectional LSTM layers with 64 and 32 units. The second type of architecture is the 1D-CNN with two variant models called **Model_B1** and **Model_B2**. **Model_B1** comprises an embedding layer, a dropout layer, 1D convolutional layer, one fully connected layer, and one classification layer. **Model_B2** is similar to **Model_B1**, but its 1D convolutional layer is followed by one LSTM layer. The third type of architecture used in the experiments is a dense network. In particular, we use

two models of this architecture: **Model_C1** and **Model_C2**. **Model_C1** contains six fully connected layers with 16 units each, while **Model_C2** contains 10 fully connected layers with 256 units each. The experiments are conducted on IMDB movie reviews by using the aforementioned architectures.By contrast, the second architecture models' experiments are not appropriate for the Reuters newswire dataset. Specifically, the second architecture cannot learn with relevant features, and it causes saturation, as demonstrated by the experimental results. Therefore, we limit the Reuters experiments to the models of the first and third architectures only. Table 26 and 27 show the accuracy results for IMDB and Reuters datasets respectively.

TABLE 26. Test accuracy of IMDB-Movie with the proposed activation functions against the others using different models.

AFs \ Model	Model_A1	Model_A2	Model_B1	Model_B2	Model_C1	Model_C2
Elu	84.60%	83.2%	88.2%	84.2%	81.24%	**81.21**%
Selu	84.19%	83.7%	88.4%	84.4%	81.03%	77.35%
Swish	85.04%	83.5%	88.3%	84.5%	81.09%	80.78%
ReLU	85.14%	83.1%	88.1%	84.7%	81.18%	80.93%
Mish	84.28%	83.2%	**88.5**%	84.5%	50.00%	50.00%
LeakyReLU	84.60%	83.3 %	87.8%	84.4%	**81.56**%	81.08%
gelu	84.53%	83.3%	88.3%	84.7%	81.19%	80.68%
IpLU	84.54%	83.4 %	88.2 %	**84.9**%	81.42%	81.17%
AbsLU	**85.36**%	**84.2**%	88.4%	84.4%	81.14%	80.70%

TABLE 27. Test accuracy of reuters newswire topics dataset with the proposed activation functions against the others using different models.

AFs \ Model	Model_A1	Model_A2	Model_C1	Model_C2
Elu	45.12%	44.96%	63.6%	74.08%
Selu	45.07%	45.36%	63.2 %	72.33%
Swish	44.67%	45.09%	50.5%	71.3%
ReLU	44.96%	46.12%	52.8%	71.61%
Mish	43.58%	46.03%	**66.8**%	71.61%
LeakyReLU	44.05%	45.8 %	61.6%	72.79%
Gelu	**45.14**%	46%	61.9%	**74.77**%
IpLU	44.59%	**46.16**%	66.2%	74.43%
AbsLU	44.74%	46%	53.5%	73.68%

The experimental results on the text benchmarks reveal that the proposed activation functions are competitive at each model's level and with different ranks of accuracy. The average accuracies for the text benchmarks on all running models are listed in Table 28. The latter part of the table shows that AbsLU achieves the best accuracy on the IMDB benchmark and that IpLU ranks second. For the Reuters dataset, IpLU achieves the best results in Reuters newswire while AbsLU ranks the seventh.

The experimental results illustrate that the proposed activation functions outperform the other competitive functions in general. The proposed activation functions are tested on a wide range of experiments. In each experiment, their performance is relatively stable. No saturation or dying is observed. The proposed activation factions are thus considered robust.

TABLE 28. Accuracy average of the proposed functions against the others using different popular nets on text benchmarks.

AFs \ Dataset	IMDB Movie	Reuters Newswire
Elu	83.78 %	56.94%
Selu	83.16 %	56.50%
ReLU	83.86%	53.88%
LeakyReLU	83.79%	56.07%
Gelu	83.79%	56.96%
Swish	83.88%	52.89%
Mish	73.41	57.01%
IpLU	83.94%	**57.85**%
AbsLU	**84.03**%	54.48 %

The proper value of the parameter α exerts an impact on the improvement of the performance. As network deepens, AbsLU with a minimal value $\alpha \leq 0$ can achieve a significant improvement. The value of α also differs according to the type of features that must be fired. On the other hand, for the less deep networks, a larger value of α can be applied, with consideration of the type and complexity of the dataset. For AbsLU, α with values between 0 and 0.2 is recommanded. For IpLU, α must be ≥ 1 to guarantee a small slope for a negative interval function. α should not be ≥ 4 because no significant improvement is realized with a large value.

V. OVERALL EVALUATION AND DISCUSSION

In this section, we deeply analyze the performance of all activation functions in 3,420 experiments involving different models and benchmark datasets. The analysis is conducted in four different dimensions. The first dimension of analysis is aimed at estimating the overall performance of all activation functions in the image and text classification tasks. The second dimension is aimed at identifying the most successful activation functions in all running models. The third dimension is aimed at ranking the overall performance of each

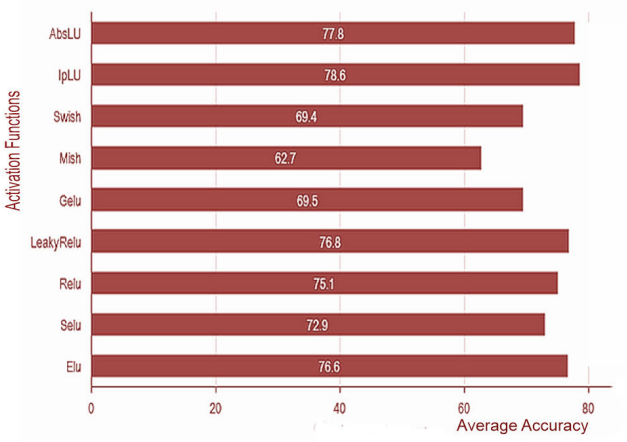

FIGURE 14. Accuracy Average of the proposed functions against the others using different popular nets on image benchmarks.

TABLE 29. The successful activation functions for each deep networks on all image and text benchmarks.

AFs / Deep Nets	Elu	Selu	ReLU	LeakyReLU	Gelu	Swish	Mish	IpLU	AbsLU
$Model_1$								✓	
$Model_2$									✓
$Model_3$								✓	
Alex								✓	
Vgg16								✓	
Vgg19								✓	
ResNet-56V1									✓
ResNet-56V2								✓	
ResNet-110								✓	
ResNet-Wrn									✓
DenseNet-100								✓	
Inception-V3	✓								
Model_A1			✓						✓
Model_A2									✓
Model_B1							✓		
Model_B2								✓	
Model-C1								✓	
Model-C2		✓						✓	

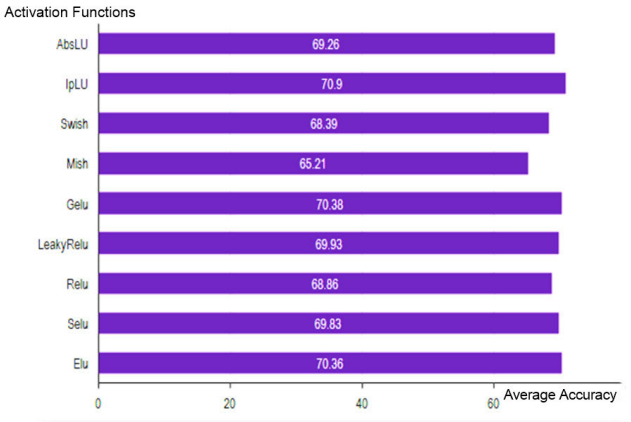

FIGURE 15. Accuracy average of the proposed functions against the others using different popular nets on text benchmarks.

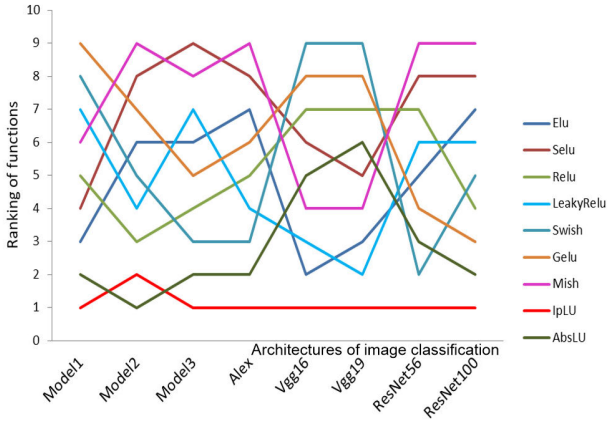

FIGURE 16. Activation functions ranking according to different depths of baseline CNN & residual network.

activation function on the basis of each network architecture's depth in the experiments. The fourth dimension of analysis is aimed at ranking the overall performance of each activation function on the basis of the variation of deep network types. In the following, we discuss the four dimensions in detail.

In the first dimension, we average each activation function of all the running models on the overall image benchmarks. Fig. 14 summarizes the average accuracy for all image benchmarks.

Although the proposed activation functions have different ranks in each image benchmark dataset, the overall average performance reveals that the proposed IpLU ranks best among all activation functions with an average accuracy of 78.6%. AbsLU ranks second with an average accuracy of 77.8%. For the text benchmark datasets, Fig. 15 illustrates all activation functions' average accuracy on the text benchmarks. Even though the ranking of IpLU varies for each benchmark dataset, the average accuracy shows that IpLU outperforms the other activation functions with an average accuracy of 70.9%. AbsLU ranks sixth with an average accuracy of 69.26%.

In the second dimension of analysis, we need to get insight into each model's most successful activation function on all datasets. Table 29 displays the most successful combinations between activation functions and the running models. This table is described as a matrix where the columns represent the activation functions, and the rows represent the models. In this matrix, the marked cell(C_{ji}), indicated by (24), means that the activation f_j at column j is the best in terms of

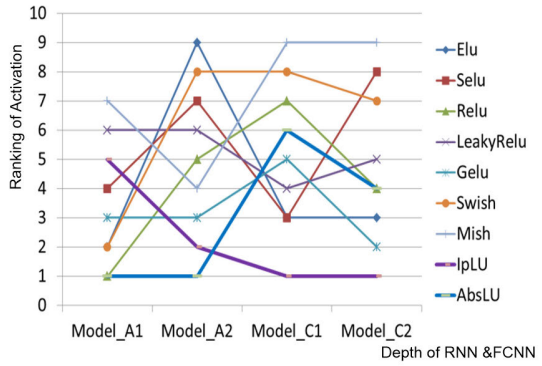

FIGURE 17. Activation functions ranking according to different depths of bidirectional-RNN & FCNN.

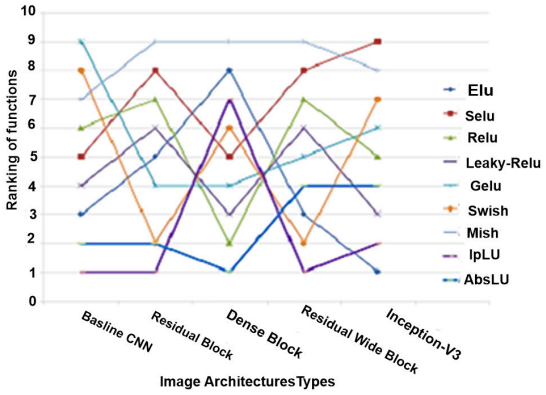

FIGURE 18. Activation function ranking based on the image models types.

the average accuracy of all experiments conducted using the model $m_i \in \mathcal{M}$ at row i in the set B of datasets, the set F activation functions and the set $\mathcal{M}$ of models. N is the number of benchmark datasets.

$$C_{ji} = \underset{f_j}{\text{argmax}} \frac{1}{N} \sum_{b \in \mathcal{B}} accuracy\ (m_i \mid b \mid f_j)$$
$$\text{where} \quad f_j \in \mathcal{F},\ m_i \in \mathcal{M} \qquad (24)$$

Table 29 shows that IpLU represents the most successful function in all the models, followed by AbsLU. This indicates that the proposed activation functions achieved a significant superiority over their counterparts using most running models.

In the third dimension, we evaluate all activation functions by using the variations of the depths of the baseline CNN models ($Model_1$, $Model_2$, $Model_3$, Alex, Vgg16, and Vgg19), and two depths of the residual networks which are ResNet-56 and ResNet-110. Fig. 16 illustrates all activation functions' rank when the depth of layers increases. The line's steepness changes denote a correlation between the changes in the depths of the models and the activation function's performance.

The analysis of these results indicates a correlation between network depth and the performance of AbsLU; that is, AbsLU achieves significant improvement using less deep baseline CNN, but it achieves better rank with a deeper Residual network. The analysis also shows that the rank of IpLU is not affected by changes in depth. Hence IpLU has the ability to work well with all different depths. Similarly, fig.17 shows the impact of increasing the depth of the bidirectional RNN with LSTM and FCNN. The results show that by increasing the layers of RNN and FCNN, Iplu and AbsLU achieve a significant improvement relative to other activation functions.

The fourth dimension of analysis ranks the activation function on the basis of the types of neural networks. Fig. 18, ranks all activation functions on each type of image network architectures. Each line in the figure represents the rank changes according to the type of network architecture. The analysis reveals that IpLU achieves a significant performance with the baseline CNN, residual networks, and dense networks. Therefore, it is ranked as the best activation function. Meanwhile, it ranks second for the Inception networks and ranks relatively low for the residual wide networks. By contrast, AbsLU achieves the best rank with the residual wide networks and ranks second with baseline CNN and Residual networks. In the case of the dense and inception networks, AbsLU achieves lower rank.

Similarly, fig. 19 ranks the activation functions on the basis of four types of networks for text benchmarks. These networks are bidirectional RNN, 1D-CNN with dense layer, 1D-CNN with LSTM layer, and a FCNN. The analysis of the results indicates that Abslu is ranked as the best activation function at the level of bidirectional RNN and it ranks second using 1D-CNN with the dense layer.

Whereas it ranks relatively lower for the 1D-CNN with LSTM and fully connected networks. On the contrary, IpLU achieves the best rank for the 1D-CNN with the LSTM layer and the fully connected networks. While it ranks lower for bidirectional RNN and 1D-CNN with the dense layer.

Further to all the previous explanations of the results, we extract the proposed activation function's overall average performance on all image and text datasets. Table 30 shows each activation function's overall performance in terms of rank.

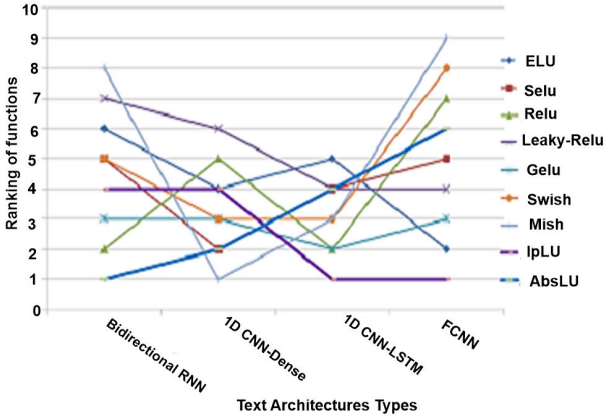

FIGURE 19. Activation function ranking based on the text models types.

TABLE 30. Overall average performance for each activation function.

AFs	Accuracy Average
IpLU	**74.75** %
AbsLU	73.53 %
Elu	73.48%
LeakyReLU	73.37%
ReLU	71.98%
Selu	71.37%
Gelu	69.94%
Swish	68.9%
Mish	63.96%

As shown in Table 30, IpLU is ranked the best activation function followed by AbsLU over all the datasets used in all experiments conducted.

VI. CONCLUSION

In neural networks, the conditions of the universal approximation theorem need not be fulfilled when selecting activation functions as long as these functions are used in the hidden layers. However, a particular activation function's effectiveness should result in significant improvement, including speeding up the learning process or boosting accuracy. A reliable activation function also tries to prevent the problems that occur during the training and optimization process, such as learning saturation, exploding/vanishing gradients, and dying problems.

This research introduced two new activation functions called IpLU and AbsLU to improve the learning performance of DNNs. In both functions, a control parameter α is used to control the effectiveness of nonlinear transformations. The proposed functions were evaluated within the framework of different neural network types, with consideration of the depths of hidden layers. Several popular benchmark datasets were used in the experiments. We categorized the experiments on the basis of the classification tasks of these benchmarks. We also extensively reviewed and categorized other activation functions found in the literature. The experimental results were compared with those of other common activation functions. A total of 3,420 experiments were conducted for the image and text classification tasks. The experimental results showed that the proposed functions outperformed the most popular activation functions in most experiments with several neural network architectures. In some classification tasks, the performance of the proposed functions was not superior but was acceptable relative to the performance of the best activation function. Nevertheless, the proposed activation function achieved superiority over the competitive activation functions across all experiments which thus highlighted the proposed functions' robustness.

The analysis of the experimental results revealed that no rule of thumb is available when generalizing superior activation functions to approximate any problem. In obtaining the best approximation, several factors such as the number of hidden neurons, number of hidden layers, network architecture, and type of real-world problems, should be considered. This research demonstrated some customized combinations of network architectures, types of models (i.e., shallow and deep), and type of problem tasks, to identify the best activation function.

REFERENCES

[1] C. Beck, W. E, and A. Jentzen, "Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations," *J. Nonlinear Sci.*, vol. 29, no. 4, pp. 1563–1619, Aug. 2019.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[3] M. Madhiarasan and S. N. Deepa, "Comparative analysis on hidden neurons estimation in multi layer perceptron neural networks for wind speed forecasting," *Artif. Intell. Rev.*, vol. 48, no. 4, pp. 449–471, Dec. 2017.

[4] M. A. Nielsen, *Neural Networks and Deep Learning*, vol. 2018. San Francisco, CA, USA: Determination Press, 2015.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 28, no. 6, pp. 1303–1314, Jun. 2018.

[7] K. Zhang, Y. Su, X. Guo, L. Qi, and Z. Zhao, "MU-GAN: Facial attribute editing based on multi-attention mechanism," *IEEE/CAA J. Automatica Sinica*, early access, Sep. 24, 2020, doi: 10.1109/JAS.2020.1003390.

[8] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen, "Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 12475–12485.

[9] M. Megahed and A. Mohammed, "Modeling adaptive E-Learning environment using facial expressions and fuzzy logic," *Expert Syst. Appl.*, vol. 157, Nov. 2020, Art. no. 113460.

[10] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A survey of deep learning and its applications: A new paradigm to machine learning," *Arch. Comput. Methods Eng.*, vol. 27, no. 4, pp. 1071–1092, Sep. 2019.

[11] B. Zhao, J. Feng, X. Wu, and S. Yan, "A survey on deep learning-based fine-grained object classification and semantic segmentation," *Int. J. Autom. Comput.*, vol. 14, no. 2, pp. 119–135, Apr. 2017.

[12] A. Kumar, S. Verma, and H. Mangla, "A survey of deep learning techniques in speech recognition," in *Proc. Int. Conf. Adv. Comput., Commun. Control Netw. (ICACCCN)*, Oct. 2018, pp. 179–185.

[13] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981314.

[14] J. de Villiers and E. Barnard, "Backpropagation neural nets with one and two hidden layers," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 136–141, Jan. 1993.

[15] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[16] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton, "On rectified linear units for speech processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3517–3521.

[17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. ICML*, vol. 30. Citeseer, 2013, p. 3.

[18] A. Pinkus, "Approximation theory of the MLP model in neural networks," *Acta Numerica*, vol. 8, pp. 143–195, Jan. 1999.

[19] S. Hayou, A. Doucet, and J. Rousseau, "On the impact of the activation function on deep neural networks training," 2019, *arXiv:1902.06853*. [Online]. Available: http://arxiv.org/abs/1902.06853

[20] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, "Activation functions: Comparison of trends in practice and research for deep learning," 2018, *arXiv:1811.03378*. [Online]. Available: http://arxiv.org/abs/1811.03378

[21] M. Cilimkovic, "Neural networks and back propagation algorithm," Inst. Technol. Blanchardstown, Dublin, Ireland, 2015, vol. 15.
[22] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, *arXiv:1710.05941*. [Online]. Available: http://arxiv.org/abs/1710.05941
[23] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.
[24] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving deep neural networks using softplus units," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–4.
[25] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*. [Online]. Available: http://arxiv.org/abs/1505.00853
[26] J. Yang, J. Wright, T. Huang, and Y. Ma, "Image super-resolution as sparse representation of raw image patches," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *J. Mach. Learn. Res.*, vol. 9, pp. 249–256, May 2010.
[28] D. Misra, "Mish: A self regularized non-monotonic neural activation function," 2019, *arXiv:1908.08681*. [Online]. Available: https://arxiv.org/abs/1908.08681
[29] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients," 2018, *arXiv:1801.03744*. [Online]. Available: https://arxiv.org/abs/1801.03744
[30] M. Chen, J. Pennington, and S. S. Schoenholz, "Dynamical isometry and a mean field theory of RNNs: Gating enables signal propagation in recurrent neural networks," 2018, *arXiv:1806.05394*. [Online]. Available: http://arxiv.org/abs/1806.05394
[31] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli, "Exponential expressivity in deep neural networks through transient chaos," 2016, *arXiv:1606.05340*. [Online]. Available: http://arxiv.org/abs/1606.05340
[32] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," 2018, *arXiv:1811.03804*. [Online]. Available: http://arxiv.org/abs/1811.03804
[33] R. Srivastava, K. Greff, and J. Schmidhuber, "Training very deep networks," 2015, *arXiv:1507.06228*. [Online]. Available: https://arxiv.org/abs/1507.06228
[34] J. Feng and S. Lu, "Performance analysis of various activation functions in artificial neural networks," *J. Phys., Conf. Ser.*, vol. 1237, Jun. 2019, Art. no. 022030.
[35] J. Kamruzzaman, "Arctangent activation function to accelerate backpropagation learning," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. 85, no. 10, pp. 2373–2376, 2002.
[36] M. I. Jordan, "The logistic function? A tutorial discussion on probabilities and neural networks," Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. 9503, 1995.
[37] P.-F. Verhulst, "Mathematical researches into the law of population growth increase," in *Nouveaux Mémoires de l'Académie Royale des Sciences et Belles-Lettres de Bruxelles*, vol. 18. 1845, pp. 1–42.
[38] R. Rojas, *Neural Networks: A Systematic Introduction*. Berlin, Germany: Springer-Verlag, 1996.
[39] D. L. Elliott, "A better activation function for artificial neural networks," ISR, Tech. Rep. T.R.93-8, 1993.
[40] J. Bergstra, G. Desjardins, P. Lamblin, and Y. Bengio, "Quadratic polynomials learn better image features," DIRO, Univ. Montreal, Montreal, QC, Canada, Tech. Rep. 1337, 2009.
[41] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.
[42] B. Xu, R. Huang, and M. Li, "Revise saturated activation functions," 2016, *arXiv:1602.05980*. [Online]. Available: http://arxiv.org/abs/1602.05980
[43] A. Krizhevsky and G. Hinton, "Convolutional deep belief networks on CIFAR-10," *Unpublished Manuscript*, vol. 40, no. 7, pp. 1–9, 2010.
[44] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier non-linearities improve neural network acoustic models," in *Proc. ICML*, vol. 30, 2013, p. 3.
[45] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.
[46] Kaggle. (Feb. 2015). *Nation Data Science Bowl Kaggle Description*. [Online]. Available: http://www.kaggle.com/c/datasciencebowl
[47] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," 2015, *arXiv:1511.07289*. [Online]. Available: http://arxiv.org/abs/1511.07289
[48] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 971–980.
[49] A. Marchisio, M. A. Hanif, S. Rehman, M. Martina, and M. Shafique, "A methodology for automatic selection of activation functions to design hybrid deep neural networks," 2018, *arXiv:1811.03980*. [Online]. Available: http://arxiv.org/abs/1811.03980
[50] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney, "Improving deep learning by inverse square root linear units (ISRLUs)," 2017, *arXiv:1710.09967*. [Online]. Available: http://arxiv.org/abs/1710.09967
[51] Brad. *New DNN Research: Isrlu—An Innovative Activation Function*. Accessed: Dec. 8, 2019. [Online]. Available: http://aiperf.com/blog/isrlu-innovation/
[52] A. Nicolae, "PLU: The piecewise linear unit activation function," 2018, *arXiv:1809.09534*. [Online]. Available: http://arxiv.org/abs/1809.09534
[53] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating second-order functional knowledge for better option pricing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 472–478.
[54] Q. Liu and S. Furber, "Noisy softplus: A biology inspired activation function," in *Proc. Int. Conf. Neural Inf. Process.* Japan: Springer, 2016, pp. 405–412.
[55] D. Hendrycks and K. Gimpel, "Gaussian error linear units (GELUs)," 2016, *arXiv:1606.08415*. [Online]. Available: http://arxiv.org/abs/1606.08415
[56] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, Nov. 2018.
[57] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: A self-gated activation function," 2017, *arXiv:1710.05941*. [Online]. Available: https://arxiv.org/abs/1710.05941
[58] M. Tanaka, "Weighted sigmoid gate unit for an activation function of deep neural network," *Pattern Recognit. Lett.*, vol. 135, pp. 354–359, Jul. 2020.
[59] M. Basirat and P. M. Roth, "The quest for the golden activation function," 2018, *arXiv:1808.00783*. [Online]. Available: http://arxiv.org/abs/1808.00783
[60] Y. Wu, H. Wang, B. Zhang, and K.-L. Du, "Using radial basis function networks for function approximation and classification," *ISRN Appl. Math.*, vol. 2012, pp. 1–34, Mar. 2012.
[61] M. S. Gashler and S. C. Ashmore, "Training deep Fourier neural networks to fit time-series data," in *Intelligent Computing in Bioinformatics*, D.-S. Huang, K. Han, and M. Gromiha, Eds. Cham, Switzerland: Springer, 2014, pp. 48–55.
[62] G. Parascandolo, H. Huttunen, and T. Virtanen, "Taming the waves: Sine as activation function in deep neural networks," in *Proc. ICLR*, 2017.
[63] T. Yu and H. Zhu, "Hyper-parameter optimization: A review of algorithms and applications," 2020, *arXiv:2003.05689*. [Online]. Available: http://arxiv.org/abs/2003.05689
[64] N. Tran, J.-G. Schneider, I. Weber, and A. K. Qin, "Hyper-parameter optimization in classification: To-do or not-to-do," *Pattern Recognit.*, vol. 103, Jul. 2020, Art. no. 107245.
[65] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, Dec. 2020.
[66] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[67] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556
[68] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Computer Vision—ECCV*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 630–645.
[69] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[70] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*. [Online]. Available: http://arxiv.org/abs/1605.07146
[71] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[72] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.
[73] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Netw.*, vol. 18, nos. 5–6, pp. 602–610, Jul. 2005.
[74] J. P. A. Vieira and R. S. Moura, "An analysis of convolutional neural networks for sentence classification," in *Proc. XLIII Latin Amer. Comput. Conf. (CLEI)*, Sep. 2017, pp. 1–5.
[75] Y. LeCun, C. Cortes, and C. J. Burges, "MNIST handwritten digit database," 2010, vol. 7, p. 23. [Online]. Available: http://yann.lecun.com/exdb/mnist
[76] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747
[77] A. Krizhevsky, V. Nair, and G. Hinton. (Jun. 2009). *Cifar-10 and Cifar-100 Datasets*. [Online]. Available: https://www.cs.toronto.edu/kriz/cifar.html
[78] *APTOS 2019 Blindness Detection*. Accessed: May 1, 2020. [Online]. Available: https://www.kaggle.com/c/aptos2019-blindness-detection
[79] *COVID-19 Open Research Dataset Challenge (CORD-19)*. Accessed: Aug. 10, 2020. [Online]. Available: https://www.kaggle.com/allen-institute-for-ai/CORD-19-research-challenge
[80] Keras Team. *Keras Documentation: Reuters Newswire Classification Dataset*. Accessed: Sep. 2, 2020. [Online]. Available: https://keras.io/api/datasets/reuters/
[81] Keras Team. *Keras Documentation: IMDB Movie Review Sentiment Classification Dataset*. Accessed: Nov. 28, 2020. [Online]. Available: https://keras.io/api/datasets/imdb/
[82] F. Chollet, "Keras: The Python deep learning library," in *Proc. ASCL*, 2018, p. 1806.
[83] T. Carneiro, R. V. Medeiros Da Nobrega, T. Nepomuceno, G.-B. Bian, V. H. C. De Albuquerque, and P. P. R. Filho, "Performance analysis of Google colaboratory as a tool for accelerating deep learning applications," *IEEE Access*, vol. 6, pp. 61677–61685, 2018.
[84] *Explore and Run Machine Learning Code With Kaggle Notebooks*. Accessed: Dec. 1, 2018. [Online]. Available: https://www.kaggle.com/kernels

ASMAA A. ALKHOULY received the master's (Diploma) degree in information technology from the Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt, in 2010, and the master's (Diploma) and M.Sc. degrees in computer science from the Arab Academy for Science, Technology, and Maritime Transport, Cairo, in 2012 and 2016, respectively. She is currently pursuing the Ph.D. degree in computer science with the Faculty of Graduate Studies for Statistical Research, Cairo University. Her research interests include neural networks, deep learning, computer vision applications, artificial intelligence algorithms in healthcare, and knowledge-based systems.

AMMAR MOHAMMED received the bachelor's and master's degrees in computer science from Cairo University, Egypt, and the Ph.D. degree in computer science from the University of Koblenz and Landau, Germany, in 2010. He worked as a Researcher and a Research Fellow with the Artificial Intelligence (AI) Research Group, University of Koblenz and Landau. He is an Associate Professor of computer science with Cairo University and Misr International University, Egypt. His research interests include machine and deep learning techniques, methods, and algorithms and their applications in several domains. He has supervised a group of Ph.D. and master's students and established the Machine/Deep Learning Research Group, Department of Computer Science, Faculty of Graduate Studies for Statistical Research, Cairo University.

HESHAM A. HEFNY received the B.Sc., M.Sc., and Ph.D. degrees in electronics and communication engineering from Cairo University, in 1987, 1991, and 1998, respectively. He is currently a Professor of computer science with the Faculty of Graduate Studies for Statistical Research (FGSSR), Cairo University. He is also the Vice Dean of graduate studies and research of FGSSR. He has authored more than 180 articles in international conferences, journals, and book chapters. His major research interests include computational intelligence (neural networks, fuzzy systems, genetic algorithms, and swarm intelligence), data mining, and uncertain decision making. He is a member of professional societies, such as the IEEE Computer, IEEE Computational Intelligence, and IEEE System, Man and Cybernetics.