Activation Functions: Comparison of Trends in Practice and Research for Deep Learning

Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall

Abstract

Deep neural networks have been successfully used in diverse emerging domains to solve real world complex problems with may more deep learning(DL) architectures, being developed to date. To achieve these state-of-the-art performances, the DL architectures use activation functions (AFs), to perform diverse computations between the hidden layers and the output layers of any given DL architecture.

This paper presents a survey on the existing AFs used in deep learning applications and highlights the recent trends in the use of the activation functions for deep learning applications. The novelty of this paper is that it compiles majority of the AFs used in DL and outlines the current trends in the applications and usage of these functions in practical deep learning deployments against the state-of-the-art research results. This compilation will aid in making effective decisions in the choice of the most suitable and appropriate activation function for any given application, ready for deployment.

This paper is timely because most research papers on AF highlights similar works and results while this paper will be the first, to compile the trends in AF applications in practice against the research results from literature, found in deep learning research to date.

Index Terms

activation function, activation function types, activation function choices, deep learning, neural networks, learning algorithms

I. INTRODUCTION

Deep learning algorithms are multi-level representation learning techniques that allows simple non-linear modules to transform representations from the raw input into the higher levels of abstract representations, with many of these transformations producing learned complex functions. The deep learning research was inspired by the limitations of the conventional learning algorithms especially being limited to processing data in raw form, [1] and the human learning techniques by changing the weights of the simulated neural connections on the basis of experiences, obtained from past data [2].

The use of representation learning, which is the technique that allow machines to discover relationships from raw data, needed to perform certain tasks likes classification and detection. Deep learning, a subfield of machine learning, is more recently being referred to as representation learning in some literature [3]. The direct relationships between deep learning and her associated fields can be shown using the relationship Venn diagram in Figure 1.

Over the past six decades, machine learning field, a branch of artificial intelligence started rapid expansion and research in the field gained momentum to diversify into different aspects of human existence. Machine learning is a field of study that

Chigozie Enyinna Nwankpa, Winifred Ijomah are with Design Manufacturing and Engineering Management, Faculty of Engineering, University of Strathclyde, Glasgow, UK, e-mail: chigozie.nwankpa@strath.ac.uk, e-mail: w.l.ijomah@strath.ac.uk

Anthony Gachagan, Stephen Marshall are with Electronic and Electrical Engineering, Faculty of Engineering, University of Strathclyde, Glasgow, UK, e-mail: a.gachagan@strath.ac.uk, e-mail: stephen.marshall@strath.ac.uk
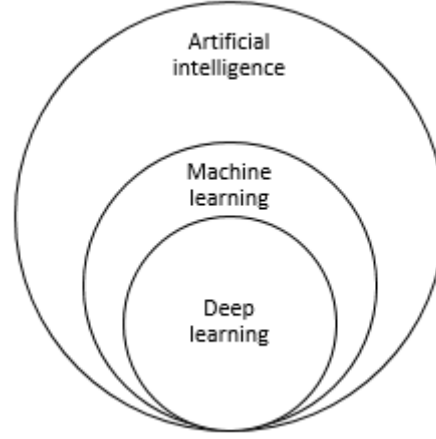
Fig. 1. Venn diagram of the components of artificial intelligence

uses the statistics and computer science principles, to create statistical models, used to perform major tasks like predictions and inference. These models are sets of mathematical relationships between the inputs and outputs of a given system. The learning process is the process of estimating the models parameters such that the model can perform the specified task [4]. For machines, the learning process tries to give machines the ability to learn without being programmed explicitly.

The typical artificial neural networks (ANN) are biologically inspired computer programmes, designed by the inspiration of the workings of the human brain. These ANNs are called networks because they are composed of different functions [5], which gathers knowledge by detecting the relationships and patterns in data using past experiences known as training examples in most literature. The learned patterns in data are modified by an appropriate activation function and presented as the output of the neuron as shown in Figure 2
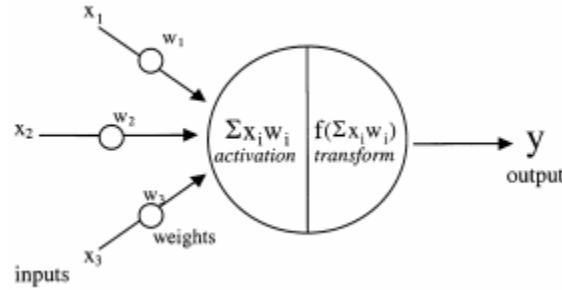

Fig. 2. Typical biological inspired neuron [6]

The early deep learning algorithms used for recognition tasks had few layers in their entire architecture, with LeNet5, having just five layers [7]. These network layers have witnessed depth increases since then, with AlexNet having twelve layers [8], VGGNet having sixteen and nineteen layers in its two variants [9], twenty-two layers in GoogleNet [10], one hundred and fifty-two layers in the largest ResNet architecture [11] and over one thousand two hundred layers in Stochastic Depth networks, already trained successfully [12], with the layers still increasing to date. With the networks getting deeper, the need to understand the makeup of the hidden layers and the successive actions taking place within the layers becomes inevitable.

However, the major issue of using deep neural network architectures is the difficulty of developing the algorithms to effectively

learn the patterns in data and studies on these issues associated with the training of neural networks, have been a key research area to date.

To improve the performance of these learning algorithms, Turian et al., 2009 highlighted that three key areas are usually of interest namely: to improve the model, to engineer better features, and to improve inference [13]. The use of better feature models has worked in many deep learning applications to obtain good model feature for the applications alongside the improvement of models but the key interest area lies in inference improvement of models for deep learning applications.

Several techniques for model improvement for deep learning algorithms exist in literature which include the use of batch-normalization and regularization, [14], [15], [16], [17], dropout [14], proper initialization [18], good choice of activation function to mention a few [15], [18], [19], [20]. These different techniques offer one form of improvement in results or training improvement but our interest lies in the activation functions used in deep learning algorithms, their applications and the benefits of each function introduced in literature.

A common problem for most learning based systems is, how the gradient flows within the network, owing to the fact that some gradients are sharp in specific directions and slow or even zero in some other directions thereby creating a problem for an optimal selection techniques of the learning parameters.

The gradients contribute to the main issues of activation function outlined in most literature include the vanishing and exploding gradients [21], [22] among others, and to remedy these issues, an understanding of the activation functions, which is one of the parameters used in neural network computation, alongside other hyperparameters drives our interest, since these functions are important for better learning and generalisation. The hyperparameters of the neural network are variables of the networks that are set, prior to the optimisation of the network. Other examples of typical hyperparameters of the network include filter size, learning rate, strength of regularisation and position of quantisation level [23], with Bergstra et al. outlining that these parameters affect the overall performance of the network.

However, these advances in configuration of the DL architectures brings new challenges specially to select the right activation functions to perform in different domains from object classification, segmentation, scene description, machine translation, cancer detection, weather forecast, self-driving cars and other adaptive systems to mention a few. With these challenges in mind, the comparison of the current trends in the application of AFs used in DL, portrays a gap in literature as the only similar research paper compared the AFs used in general regression and classification problems in ANNs, with the results reported in Turkish language, which makes it difficult to understand the research results by non Turkish scholars [24].

We summarize the trends in the application of existing AFs used in DL and highlight our findings as follows. This compilation is organized into six sections with the first four sections introducing deep learning, activation functions, the summary of AFs used in deep learning and the application trends of AFs in deep architectures respectively. Section five discusses these functions and section six provides the conclusion, alongside some future research direction.

II. ACTIVATION FUNCTIONS

Activation functions are functions used in neural networks to computes the weighted sum of input and biases, of which is used to decide if a neuron can be fired or not. It manipulates the presented data through some gradient processing usually gradient descent and afterwards produce an output for the neural network, that contains the parameters in the data. These AFs are often referred to as a transfer function in some literature.

Activation function can be either linear or non-linear depending on the function it represents, and are used to control the outputs of out neural networks, across different domains from object recognition and classification [8], [10], [25], [26], to speech

recognition [27], [28], segmentation [29], [30], scene understanding and description [31], [32], machine translation [33], [34] test to speech systems [35], [36], cancer detection systems [37], [38], [39], finger print detection [40], [41], weather forecast [42], [43], self-driving cars [44], [45], and other domains to mention a few, with early research results by [46], validating categorically that a proper choice of activation function improves results in neural network computing.

For a linear model, a linear mapping of an input function to an output, as performed in the hidden layers before the final prediction of class score for each label is given by the affine transformation in most cases [5]. The input vectors x transformation is given by

$$f(x) = w^T x + b \quad - (1.1)$$

where x = input, w = weights, b = biases.

Furthermore, the neural networks produce linear results from the mappings from equation (1.1) and the need for the activation function arises, first to convert these linear outputs into non-linear output for further computation, especially to learn patterns in data. The output of these models are given by

$$y = (w_1 x_1 + w_2 x_2 + .. + w_n x_n + b) \quad - (1.2)$$

These outputs of each layer is fed into the next subsequent layer for multilayered networks like deep neural networks until the final output is obtained, but they are linear by default. The expected output determines the type of activation function to be deployed in a given network.

However, since the output are linear in nature, the nonlinear activation functions are required to convert these linear inputs to non-linear outputs. These AFs are transfer functions that are applied to the outputs of the linear models to produce the transformed non-linear outputs, ready for further processing. The non-linear output after the application of the AF is given by

$$y = \alpha(w_1 x_1 + w_2 x_2 + .. + w_n x_n + b) \quad - (1.3)$$

Where α is the activation function.

The need for these AFs include to convert the linear input signals and models into non-linear output signals, which aids the learning of high order polynomials beyond one degree for deeper networks. A special property of the non-linear activation functions is that they are differentiable else they cannot work during backpropagation of the deep neural networks [5].

The deep neural network is a neural network with multiple hidden layers and output layer. An understanding of the makeup of the multiple hidden layers and output layer is our interest. A typical block diagram of a deep learning model is shown in Figure 3, which shows the three layers that make up a DL based system with some emphasis on the positions of activation functions, represented by the dark shaded region in the respective blocks.
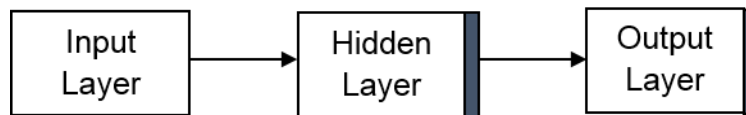

Fig. 3. Block diagram of a DL based system model showing the activation function

The input layer accepts the data for training the neural network which comes in various formats from images, videos, texts, speech, sounds, or numeric data, while the hidden layers are made up of mostly the convolutional and pooling layers, of which the convolutional layers detect the local the patterns and features in data from the previous layers, presented in array-like forms for images while the pooling layers semantically merges similar features into one [1]. The output layer presents the network results which are often controlled by AFs, specially to perform classifications or predictions, with associated probabilities.

The position of an AF in a network structure depends on its function in the network thus when the AF is placed after the hidden layers, it converts the learned linear mappings into non-linear forms for propagation while in the output layer, it performs predictions.

The deep architectures are composed of several processing layers with each, involving both linear and non-linear operations, that are learned together to solve a given task. These deeper networks come with better performances though common issues likes vanishing gradients and exploding gradient arises, as a result of the derivative terms which are usually less than 1. With successive multiplication of this derivative terms, the value becomes smaller and smaller and tends to zero, thus the gradient vanishes. Consequently, if the values are greater than 1, successive multiplication will increase the values and the gradient tends to infinity thereby exploding the gradient. Thus, the AFs maintains the values of these gradients to specific limits. These are achieved using different mathematical functions and some of the early proposals of activation functions, used for neural network computing were explored by Elliott, 1993 as he studied the usage of the AFs in neural network [47].

The compilation of the existing activation functions is outlined with the advantages offered by most of the respective functions as highlighted by the authors as found in the literature.

III.   SUMMARY OF ACTIVATION FUNCTIONS

This section highlights the different types of AFs and their evolution over the years. The AF research and applications in deep architectures, used in different applications has been a core research field to date. The state-of -the -art research results are outlined as follows though, It is worthy to state categorically that this summary of the AFs are not reported in chronological order but arranged with the main functions first, and their improvements following as their variants. These functions are highlighted as follows:

A.   *Sigmoid Function*

The Sigmoid AF is sometimes referred to as the logistic function or squashing function in some literature [13]. The Sigmoid function research results have produced three variants of the sigmoid AF, which are used in DL applications. The Sigmoid is a non-linear AF used mostly in feedforward neural networks. It is a bounded differentiable real function, defined for real input values, with positive derivatives everywhere and some degree of smoothness [48]. The Sigmoid function is given by the relationship

$$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right) - (1.4)$$

The sigmoid function appears in the output layers of the DL architectures, and they are used for predicting probability based output and has been applied successfully in binary classification problems, modeling logistic regression tasks as well as other neural network domains, with Neal [49] highlighting the main advantages of the sigmoid functions as, being easy to understand and are used mostly in shallow networks. Moreover, Glorot and Bengio, 2010 suggesting that the Sigmoid AF should be avoided when initializing the neural network from small random weights [18].

However, the Sigmoid AF suffers major drawbacks which include sharp damp gradients during backpropagation from deeper hidden layers to the input layers, gradient saturation, slow convergence and non-zero centred output thereby causing the gradient updates to propagate in different directions. Other forms of AF including the hyperbolic tangent function was proposed to remedy some of these drawbacks suffered by the Sigmoid AF.

1) Hard Sigmoid Function: The hard sigmoid activation is another variant of the sigmoid activation function and this function is given by

$$f(x) = clip\left(\frac{(x+1)}{2}, 0, 1\right) - (1.5)$$

The equation(1.5) can be re-written in the form

$$f(x) = max\left(0, min\left(1, \frac{(x+1)}{2}\right)\right) - (1.6)$$

A comparison of the hard sigmoid with the soft sigmoid shows that the hard sigmoid offer lesser computation cost when implemented both in a specialized hardware or software form as outlined [50], and the authors highlighted that it showed some promising results on DL based binary classification tasks.

2) Sigmoid-Weighted Linear Units (SiLU): The Sigmoid-Weighted Linear Units is a reinforcement learning based approximation function. The SiLU was proposed by Elfwing et al., 2017 and the SiLU function is computed as Sigmoid multiplied by its input [51]. The AF a_k of a SiLU is given by

$$a_k(s) = z_k\alpha(z_k) - (1.7)$$

where s = input vector, z_k = input to hidden units k. The input to the hidden layers is given by

$$z_k = \sum_i w_{ik}s_i + b_k - (1.8)$$

Where b_k is the bias and w_{ik} is the weight connecting to the hidden units k respectively.

The SiLU function can only be used in the hidden layers of the deep neural networks and only for reinforcement learning based systems. The authors also reported that the SiLU outperformed the ReLU function as seen in the response in Figure 4.

3) Derivative of Sigmoid-Weighted Linear Units (dSiLU): The derivative of the Sigmoid-Weighted Linear Units is the gradient of the SiLU function and referred to as dSiLU. The dSiLU is used for gradient-descent learning updates for the neural network weight parameters, and the dSiLU is given by

$$a_k(s) = \alpha(z_k)(1 + z_k(1 - \alpha(z_k))) - (1.9)$$

The dSiLU function response looks like an overshooting Sigmoid function as shown in Figure 4. The authors highlighted that the dSiLU outperformed the standard Sigmoid function significantly [51].
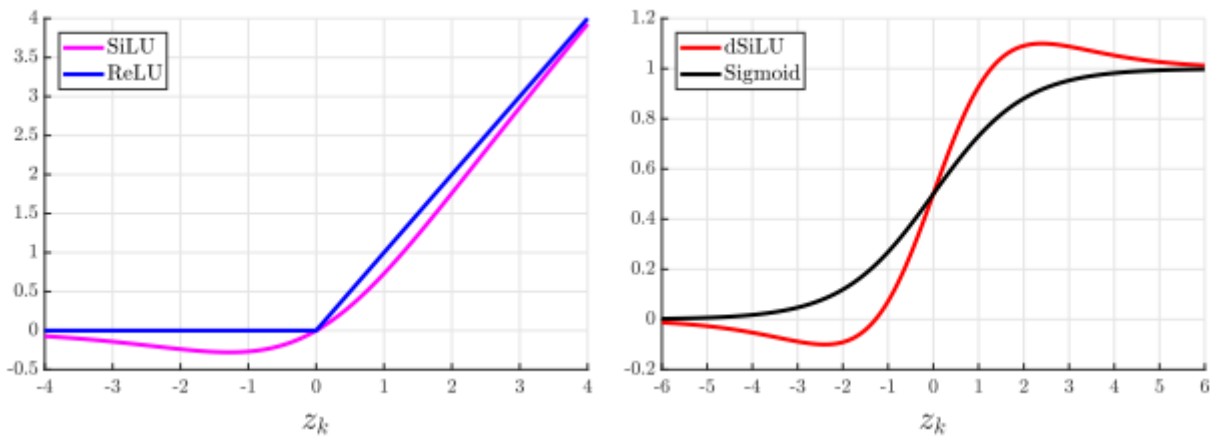

Fig. 4. SiLU response comparison [51]

B. *Hyperbolic Tangent Function (Tanh)*

The hyperbolic tangent function is another type of AF used in DL and it has some variants used in DL applications. The hyperbolic tangent function known as tanh function, is a smoother [1] zero-centred function whose range lies between -1 to 1, thus the output of the tanh function is given by

$$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right) - (1.10)$$

The tanh function became the preferred function compared to the sigmoid function in that it gives better training performance for multi-layer neural networks [46], [49]. However, the tanh function could not solve the vanishing gradient problem suffered by the sigmoid functions as well. The main advantage provided by the function is that it produces zero centred output thereby aiding the back-propagation process.

A property of the tanh function is that it can only attain a gradient of 1, only when the value of the input is 0, that is when x is zero. This makes the tanh function produce some dead neurons during computation. The dead neuron is a condition where the activation weight, rarely used as a result of zero gradient. This limitation of the tanh function spurred further research in activation functions to resolve the problem, and it birthed the rectified linear unit (ReLU) activation function.

The tanh functions have been used mostly in recurrent neural networks for natural language processing [52] and speech recognition tasks [53].
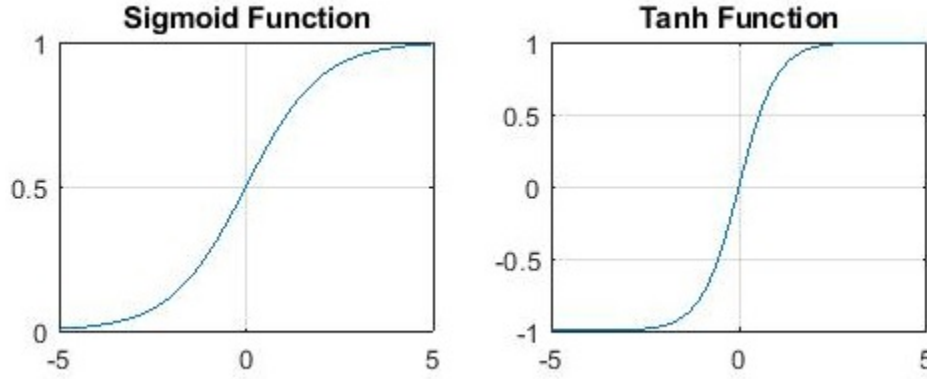

Fig. 5. Pictorial representation of Sigmoid and Tanh activation function responses

1) Hard Hyperbolic Function: The hard hyperbolic function known as the Hardtanh function is another variant of the tanh activation function used in deep learning applications. The Hardtanh represents a cheaper and more computational efficient version of tanh. The Hardtanh function lies within the range of -1 to 1 and it is given by

$$f(x) = \begin{pmatrix} -1 & if \ x < -1 \\ x \ if \ -1 = x \leqslant 1 \\ 1 & if \ x > 1 \end{pmatrix} - (1.11)$$

The hardtanh function has been applied successfully in natural language processing [54], with the authors reporting that it provided both speed and accuracy improvements.

C. *Softmax Function*

The Softmax function is another types of activation function used in neural computing. It is used to compute probability distribution from a vector of real numbers. The Softmax function produces an output which is a range of values between 0 and 1, with the sum of the probabilities been equal to 1. The Softmax function [5] is computed using the relationship

$$f(x_i) = \frac{exp\ (x_i)}{\sum_j exp\ (x_j)} \quad - (1.12)$$

The Softmax function is used in multi-class models where it returns probabilities of each class, with the target class having the highest probability. The Softmax function mostly appears in almost all the output layers of the deep learning architectures, where they are used [8], [29], [55].

The main difference between the Sigmoid and Softmax AF is that the Sigmoid is used in binary classification while the Softmax is used for multivariate classification tasks.

D. *Softsign*

The Softsign is another type of AF that is used in neural network computing. The Softsign function was introduced by Turian et al., 2009 and the Softsign is another non-linear AF used in DL applications [13]. The Softsign function is a quadratic polynomial, given by

$$f(x) = \left(\frac{x}{|x|\ +\ 1}\right) \quad - (1.13)$$

Where $|x|$ = absolute value of the input.

The main difference between the Softsign function and the tanh function is that the Softsign converges in polynomial form unlike the tanh function which converges exponentially.

The Softsign has been used mostly in regression computation problems [56] but has also been applied to DL based test to speech systems [36], with the authors reporting some promising results using the Softsign function.

E. *Rectified Linear Unit (ReLU) Function*

The rectified linear unit (ReLU) activation function was proposed by Nair and Hinton 2010, and ever since, has been the most widely used activation function for deep learning applications with state-of-the-art results to date [57]. The ReLU is a faster learning AF [1], which has proved to be the most successful and widely used function [19]. It offers the better performance and generalization in deep learning compared to the Sigmoid and tanh activation functions [58], [59]. The ReLU represents a nearly linear function and therefore preserves the properties of linear models that made them easy to optimize, with gradient-descent methods [5].

The ReLU activation function performs a threshold operation to each input element where values less than zero are set to zero thus the ReLU is given by

$$f(x) = max\ (0, x) = \begin{cases} x_{i,} & if\ x_i\ \geq\ 0 \\ 0, & if\ x_i\ <\ 0 \end{cases} \quad - (1.14)$$

This function rectifies the values of the inputs less than zero thereby forcing them to zero and eliminating the vanishing gradient problem observed in the earlier types of activation function. The ReLU function has been used within the hidden units of

the deep neural networks with another AF, used in the output layers of the network with typical examples found in object classification [8], [25] and speech recognition applications [53].

The main advantage of using the rectified linear units in computation is that, they guarantee faster computation since it does not compute exponentials and divisions, with overall speed of computation enhanced [58]. Another property of the ReLU is that it introduces sparsity in the hidden units as it squishes the values between zero to maximum. However, the ReLU has a limitation that it easily overfits compared to the sigmoid function although the dropout technique has been adopted to reduce the effect of overfitting of ReLUs and the rectified networks improved performances of the deep neural networks [20].

The ReLU and its variants have been used in different architectures of deep learning, which include the restricted Boltzmann machines [57] and the convolutional neural network architectures [8], [10], [25], [60], [26], although (Nair and Hinton, 2010) outlined that the ReLU has been used in numerous architectures because of its simplicity and reliability [57].

The ReLU has a significant limitation that it is sometimes fragile during training thereby causing some of the gradients to die. This leads to some neurons being dead as well, thereby causing the weight updates not to activate in future data points, thereby hindering learning as dead neurons gives zero activation [5]. To resolve the dead neuron issues, the leaky ReLU was proposed.

1) Leaky ReLU (LReLU): The leaky ReLU, proposed in 2013 as an AF that introduce some small negative slope to the ReLU to sustain and keep the weight updates alive during the entire propagation process [53]. The alpha parameter was introduced as a solution to the ReLUs dead neuron problems such that the gradients will not be zero at any time during training. The LReLU computes the gradient with a very small constant value for the negative gradient α in the range of 0.01 thus the LReLU AF is computed as

$$f(x) = \alpha x + x = \begin{cases} x & if\ x\ >\ 0 \\ \alpha x & if\ x\ \leq\ 0 \end{cases} - (1.15)$$

The LReLU has an identical result when compared to the standard ReLU with an exception that it has non-zero gradients over the entire duration thereby suggesting that there no significant result improvement except in sparsity and dispersion when compared to the standard ReLU and tanh functions [53]. The LReLU was tested on automatic speech recognition dataset.

2) Parametric Rectified Linear Units (PReLU): The parametric ReLU known as PReLU is another variant of the ReLU AF proposed by He et al., 2015, and the PReLU has the negative part of the function, being adaptively learned while the positive part is linear [25]. The PReLU is given by

$$f(x_i) = \begin{pmatrix} x_i\ , & if\ x_i\ >\ 0 \\ a_i x_i\ , & if\ x_i\ \leq 0 \end{pmatrix} - (1.16)$$

Where a_i is the negative slope controlling parameter and its learnable during training with back-propagation. If the term $a_i\ =\ 0$ the PReLU becomes ReLU.

The PReLU can be written in compact form as

$$f(x_i\ ) = max\ (o,\ x_i)\ +\ a_i\ min\ (0,\ x_i) - (1.17)$$

The authors reported that the performance of PReLU was better than ReLU in large scale image recognition and these results from the PReLU was the first to surpass human-level performance on visual recognition challenge [25].

3) Randomized Leaky ReLU (RReLU): The randomized leaky ReLU is a dynamic variant of leaky ReLU where a random number sampled from a uniform distribution $U(l, u)$ is used to train the network. The randomized ReLU is given by

$$f(x_i) = \begin{pmatrix} x_{ji}, & if \ x_{ji} \geq 0 \\ a_{ji}x_{ji}, & if \ x_{ji} < 0 \end{pmatrix} - (1.18)$$

Where $a_i \sim U(l, u)$, $l < u$ and $l, u \in [0, 1]$

The test phase uses an averaging technique of all a_{ji} as in the training, without the dropout used in the learned parameter to a deterministic value, obtained as $a_{ji} = \frac{l+u}{2}$. Thus, the real test output is obtained using the following relationship.

$$y_{ji} = \frac{x_{ji}}{\frac{l+u}{2}} - (1.19)$$

The RReLU has been tested on standard classification datasets and compared against the other variants of the ReLU AF and Xu et al., 2015 validated that LReLU, RReLU and PReLU performs better than the ReLU on classification tasks [61].

4) S-shaped ReLU (SReLU): The S-shaped ReLU is another variant of the ReLU AF used to learn both convex and non-convex functions, inspired by the laws of neural sciences and psychophysics. This SReLU AF was proposed by Jin et al., 2015, and it consists of three-piece wise linear functions, formulated by four learnable parameters, which are learned during training of the deep neural network using backpropagation [62].

The SReLU is defined by the following mapping relationship

$$f(x) = \begin{pmatrix} t_i^r + a_i^r(x_i - t_i^r), & x_i \geq t_i^r \\ x_i, & t_i^r > x_i > t_i^l \\ t_i^l + a_i^l(x_i - t_i^l), & x_i \leq t_i^l \end{pmatrix} - (1.20)$$

where t_i^l, t_i^r and a_i^l are learnable parameters of the network and *i* indicates that the SReLU can vary in different channels. The parameter a_i^r represents slope of the right line with input above the set threshold t_i^r and t_i^l are thresholds in positive and negative directions respectively.

The authors highlighted that SReLU was tested on some of the award-winning CNN architectures, the Network in Network architecture alongside GoogLeNet, for on image recognition tasks specifically CIFAR-10, ImageNet, and MNIST standard datasets, and it showed improved results, compared to the other AFs [62].

F. Softplus Function

The Softplus AF is a smooth version of the ReLU function which has smoothing and nonzero gradient properties, thereby enhancing the stabilization and performance of deep neural network designed with softplus units.

The Softplus was proposed by Dugas et al., 2001, and the Softplus function is a primitive of the sigmoid function, given by the relationship [63]

$$f(x) = \log(1 + exp^x) - (1.21)$$

The Softplus function has been applied in statistical applications mostly however, a comparison of the Softplus function with the ReLU and Sigmoid functions by Zheng et al., 2015, showed an improved performance with lesser epochs to convergence during training, using the Softplus function [64].

The Softplus has been used in speech recognition systems [64] [65], of which the ReLU and sigmoid functions have been the dominant AFs, used to achieve auto speech recognition.

G. *Exponential Linear Units (ELUs)*

The exponential linear units (ELUs) is another type of AF proposed by Clevert et al., 2015, and they are used to speed up the training of deep neural networks. The main advantage of the ELUs is that they can alleviate the vanishing gradient problem by using identity for positive values and also improves the learning characteristics. They have negative values which allows for pushing of mean unit activation closer to zero thereby reducing computational complexity thereby improving learning speed [66]. The ELU represents a good alternative to the ReLU as it decreases bias shifts by pushing mean activation towards zero during training.

The exponential linear unit (ELU) is given by

$$f(x) = \begin{pmatrix} x, & if \ x \ > \ 0 \\ \alpha \ exp(x) \ - \ 1, & if \ x \leq 0 \end{pmatrix} \ - (1.22)$$

The derivative or gradient of the ELU equation is given as

$$f' = \begin{pmatrix} 1, & if \ x \ > \ 0 \\ f(x) \ + \ \alpha, & if \ x \ \leq \ 0 \end{pmatrix} \ - (1.23)$$

Where α= ELU hyperparameter that controls the saturation point for negative net inputs which is usually set to 1.0

The ELUs has a clear saturation plateau in it negative regime thereby learning more robust representations, and they offer faster learning and better generalisation compared to the ReLU and LReLU with specific network structure especially above five layers and guarantees state-of-the-art results compared to ReLU variants. However, a critical limitation of the ELU is that the ELU does not centre the values at zero, and the parametric ELU was proposed to address this issue [67].

1) Parametric Exponential Linear Unit (PELU): The parametric ELU is another parameterized version of the exponential linear unit (ELUs), which tries to address the zero centre issue found in the ELUs. The PELU was proposed by Trottier et al., 2017, and it uses the PELU in the context of vanishing gradient to provide some gradient-based optimization framework used to reduce bias shifts while maintaining the zero centre of values [67].

The PELU has two additional parameters compared to the ELU and the modified ELU is given by

$$f(x) = \begin{pmatrix} cx, & if \ x \ > \ 0 \\ \alpha \ exp^{\frac{x}{b}} \ - \ 1, & if \ x \leq 0 \end{pmatrix} \ - (1.24)$$

Where a, b, and c > 0 and c causes a change in the slope in the positive quadrant, b controls the scale of the exponential decay, and α controls the saturation in the negative quadrant.

Solving the equation of the modified ELU by constraining the projection during training given the parametric ELU which is given by the relationship

$$f(x) = \begin{pmatrix} \frac{\alpha}{b}x, & if \ x \ \geq \ 0 \\ \alpha \ exp^{\frac{x}{b}} \ - \ 1, & if \ x \ < \ 0 \end{pmatrix} \ - (1.25)$$

when α and b > 0. In compact form, the PELU is given by

$$f(x_i) = max(0, \ x_i \ ) + \ a_i \ min(0, \ x_i \ ) - (1.26)$$

Thus, setting the values of a, b, and c = 1 recovers the original ELU activation function.

The PELU promises to be a good option for applications that requires less bias shifts and vanishing gradients like the CNNs.

2) Scaled Exponential Linear Units (SELU): The scaled exponential linear units(SELU) is another variant of the ELUs, proposed by Klambauer et al., 2017. The SELU was introduced as a self-normalising neural network that has a peculiar property of inducing self-normalising properties. It has a close to zero mean and unit variance that converges towards zero mean and unit variance when propagated through multiple layers during network training, thereby making it suitable for deep learning application, and with strong regularisation, learns robust features efficiently [68].

The SELU is given by

$$f(x) = \tau \begin{pmatrix} x, & if \ x > 0 \\ \alpha \ exp(x) - \alpha, & if \ x \le 0 \end{pmatrix} - (1.27)$$

Where τ is the scale factor. The approximate values of the parameters of the SELU function are $\alpha \approx 1.6733 \ and \ \lambda \approx 1.0507$.

The SELUs are not affected by vanishing and exploding gradient problems and the authors have reported that they allow the construction of mappings with properties leading to self normalizing neural networks which cannot be derived using ReLU, scaled ReLU, sigmoid, LReLU and even tanh functions.

The SELU has been applied successfully to classification tasks [68] alongside some deep genetic mutation computation [69].

H. Maxout Function

The Maxout AF is a function where non-linearity is applied as a dot product between the weights of a neural network and data. The Maxout, proposed by Goodfellow et al., 2013, generalizes the leaky ReLU and ReLU where the neuron inherits the properties of ReLU and leaky ReLU where no dying neurons or saturation exist in the network computation [70]. The Maxout function is given by

$$f(x) = max(w_1^T \ x + b_1 \ , \ w^T_2 \ x \ + b_2 \ ) - (1.28)$$

Where w = weights, b = biases, T = transpose.

The Maxout function has been tested successfully in phone recognition applications [71].

The major drawback of the Maxout function is that it is computationally expensive as it doubles the parameters used in all neurons thereby increasing the number of parameters to compute by the network.

I. Swish Function

The Swish AF is one of the first compound AF proposed by the combination of the sigmoid AF and the input function, to achieve a hybrid AF. The Swish activation was proposed by Ramachandran et al., 2017, and it uses the reinforcement learning based automatic search technique to compute the function. The properties of the Swish function include smoothness, non-monotonic, bounded below and unbounded in the upper limits. The smoothness property makes the Swish function produce better optimization and generalization results when used in training deep learning architectures [19].

The Swish function is given by

$$f(x) = x \ \cdot \ sigmoid(x) \ = \ \frac{x}{1 \ + \ e^{-x}} - (1.29)$$

The authors highlighted that the main advantages of the Swish function is the simplicity and improved accuracy as the Swish does not suffer vanishing gradient problems but provides good information propagation during training and reported that the Swish AF outperformed the ReLU activation function on deep learning classification tasks.

J. ELiSH

The Exponential linear Squashing AF known as the ELiSH function is one of the most recent AF, proposed by Basirat and Roth, 2018. The ELiSH shares common properties with the Swish function. The ELiSH function is made up of the ELU and Sigmoid functions and it is given by

$$f(x) = \begin{cases} \left(\frac{x}{1+e^{-x}}\right), & x \geq 0 \\ \left(\frac{e^x-1}{1+e^{-x}}\right), & x < 0 \end{cases} \quad - (1.30)$$

The properties of the ELiSH function varies in both the negative and positive parts as defined by the limits. The Sigmoid part of the ELiSH function improves information flow while the Linear parts eliminates the vanishing gradient issues. The ELiSH function has been applied successfully on ImageNet dataset using different deep convolutional architectures [72].
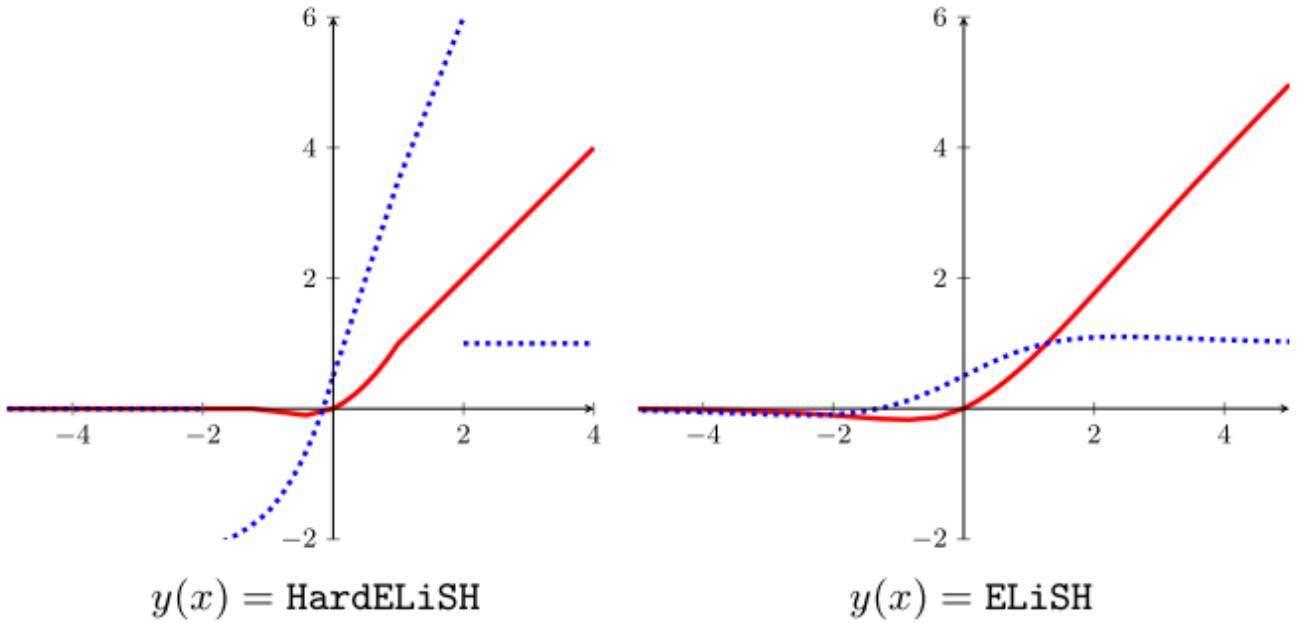

Fig. 6. The ELiSH and HardELiSH function responsees [72]

1) HardELiSH: The HardELiSH is the hard variant of the ELiSH activation function. The HardELiSH is a multiplication of the HardSigmoid and ELU in the negative part and a multiplication of the Linear and the HardSigmoid in the positive part [72]. The HardELiSH is given by

$$f(x) = \begin{cases} x \times max\left(0, min\left(1, \left(\frac{x+1}{2}\right)\right)\right), & x \geq 0 \\ (e^x - 1) \times max\left(0, min\left(1, \left(\frac{x+1}{2}\right)\right)\right), & x < 0 \end{cases} \quad - (1.31)$$

The HardELiSH function was tested on ImageNet classification dataset.

IV. COMPARISON OF THE TRENDS IN ACTIVATION FUNCTIONS USED IN DEEP LEARNING ARCHITECTURES

The search for the AFs are based on the published research results of the winners of ImageNet Image Large Scale Visual recognition Challenge (ILSVRC) competitions alongside some cited research output from the AF research results found in literature. The ImageNet competition was chosen for trend comparison because it is the competition that produced the first deep learning success [8]. These forms the core basis for both the search of the functions outlined in this paper and the current

trends in the use of AFs, although the scope of the trends goes beyond image recognition as other applications that birthed new DL AFs were included like the natural language processing [13]

The Image Large Scale Visual Recognition Challenge (ILSVRC) also known as ImageNet is a database of images used for visual recognition competitions. The ImageNet competition is an annual competition where researchers and their teams evaluate developed algorithms on specific datasets, to review improvements in achieved accuracy in visual recognition challenges.

The deep learning architectures are the architectures that has more than one hidden layer, often referred to as multilayer perceptron. These architectures are numerous which include deep feedforward neural networks, convolutional neural networks, long short term memory, recurrent neural networks and the deep generative models like deep Boltzmann machines, deep belief networks, generative adversarial networks and so on [1], [5]. The use of the architectures of DL includes to learn patterns in data, map some input function to outputs and many more, and these can only be achieved with specialized architectures.

The AFs used in DL architectures dates back from the beginning of the adoption of the deeper architectures for neural network computations. Prior to 2012, image recognition challenges used shallow architectures and had few AFs embedded in them [73]. The adoption of the deeper architectures for neural computing was first explored by Krizhevsky et al., 2012 in the Image Large Scale Visual Recognition Challenge (ILSVRC) [8], which is an annual event that started in 2010. Since then, the use of the deeper architectures has witnessed huge research advances in optimization techniques for training the deeper architectures, since the deeper the network, the more difficult to train and optimize [10], [74], though the better the performance guarantee.

The AF is a key component for training and optimization of neural networks, implemented on different layers of DL architectures, is used across domains including natural language processing, object detection, classification and segmentation etc.

The trends of DL architectures that emerged as the winning architectures of ImageNet competition and other remarkable architectures include: AlexNet; winner of ILSVRC 2012 [8], ZFNet; the 2013 ILSVRC competition winner [60], NiN; the Network in Network architecture [55], GoogleNet; the winner of 2015 ILSVRC classification challenge [10], VGG16 and VGG19 architectures which emerged as second in classification, winner in localization tasks in 2015 ILSVRC [9], Network in Network(NiN) [55], ResNet; the 2016 winner of both the ILSVRC & COCO competitions [11], Squeeze and Excitation Network(SeNet); the latest winner of the ILSVRC 2017 competition [75] alongside SegNet [29], DenseNet [76], FractalNet [77], ResNeXt architecture [78], and the modified networks like SqueezeNet architectures [79], which had multiple AFs embedded in it.

The Table I highlights some of the state-of-the-art architectures of deep neural networks that emerged as a significant improvement to the existing architectures, used for large scale image recognition challenge (ILSVRC), showing the positions of the activation functions used in those architectures.

The AFs used in more recent DL architectures have the ReLU activation function embedded in them which include the DenseNet [76], MobileNets, a mobile version of the convolutional networks [80], ResNeXt [78] as well as the softmax function used in FractalNet [77], and many some other architectures.

From Table Iabove, it is evident that the ReLU and the Softmax activation functions are the dormant AFs used in practical DL applications. Furthermore, the Softmax function is used in the output layer of most common practice DL applications however, the most recent DL architecture used the sigmoid function to achieve it prediction at the output layer, while the ReLU units are used in the hidden layers.

In choosing the state-o-the-art architecture for recognition tasks, it is handy from Table I to select the current and most

TABLE I

TYPES AND POSITIONS OF AFS USED IN DL ARCHITECTURES.

Architecture	Hidden Layers	Output Layer	Cross Reference
AlexNet	ReLU	Softmax	Krizhevsky et al., 2012
NiN	No activation	Softmax	Lin et al., 2013
ZFNet	ReLU	Softmax	Zeiler & Fergus, 2013
VGGNet	ReLU	Softmax	Simonyan & Zisserman, 2015
SegNet	ReLU	Softmax	Badrinarayanan et al., 2015
GoogleNet	ReLU	Softmax	Szegedy et al., 2015
SqueezeNet	ReLU	Softmax	Golkov et al., 2016
ResNet	ReLU	Softmax	He et al., 2016
ResNeXt	ReLU	Softmax	Xie et al., 2017
MobileNets	ReLU	Softmax	Howard et al., 2017
SeNet	ReLU	Sigmoid	Fu et al., 2017

recent architecture and understand the architectural make-up of the network. The SeNet architecture is the current state-of-the-art architecture for recognition tasks as found in literature.

A summary table of all the discussed AFs used in DL is shown in Table II, with their respective equations, used in computation of these AFs.

V. DISCUSSIONS

We have identified the key AFs used in deep learning applications and from the literature, we have outlined that there are four variants of the rectified linear units (ReLU), apart from the original ReLU. Furthermore, the Sigmoid has three variants, two variants for the exponential linear units(ELU) functions, with the Hyperbolic tangent and ELiSH functions having their respective hard versions as their only variants. The Softmax, Softsign, Softplus, Maxout and Swish functions has no variants.

The summary of the various AFs include the Sigmoid with HardSigmoid, SiLU and dSiLU variants, the ELU having the PELU and SELU as its variants, the ReLU with LReLU, PReLU, RReLU and SReLU as its variants.

Perhaps, the DL architectures have multiple AFs embedded to every given network and it is noteworthy to highlight that these multiple AFs are used at different layers, to perform gradient computations in a single network, thereby obtaining an effective learning and characterisation of the presented inputs.

The linear units of AF are the most studied types of AF with the rectified and exponential variants, providing eight different variants of activation functions, used in DL applications and these eight linear variants include ReLU, LReLU, PReLU, RReLU, SReLU, ELU, PELU and SELU.

A summary of the AFs used across various domains is shown in Table 2 alongside the formulae for the computation of these AFs, used in DL applications.

The ELU's has been highlighted as a faster learning AF compared to their ReLU counterpart [81], [67], and this assertion has been validated by Pedamonti, 2018, after an extensive comparison of some variants of the ELU and ReLU AF on the MNIST recognition dataset [82].

The PELU and PReLU are two different parametric AFs which are developed using exponential and linear units but both have learnable parameters. It is also evident that there are other AFs that performs better than the ReLU, which has been the most consistent AF for DL applications since invention [19] [25]. Xu et al., 2015 validated that the other variants of the

TABLE II

DL ACTIVATION FUNCTIONS AND THEIR CORRESPONDING EQUATIONS FOR COMPUTATION.

S/N	Function	Computation Equation
1	Sigmoid	$f(x) = \left(\frac{1}{(1 + exp^{-x})} \right)$
2	HardSigmoid	$f(x) = max \left(0, min \left(1, \frac{(x+1)}{2} \right) \right)$
3	SiLU	$a_k(s) = z_k \alpha(z_k)$
4	dSiLU	$a_k(s) = \alpha(z_k)(1 + z_k(1 - \alpha(z_k)))$
5	Tanh	$f(x) = \left(\frac{e^x - e^{-x}}{e^x + e^{-x}} \right)$
6	Hardtanh	$f(x) = \begin{pmatrix} -1 & if\ x < -1 \\ x\ if\ -1 = x \leqslant 1 \\ 1 & if\ x > 1 \end{pmatrix}$
7	Softmax	$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$
8	Softplus	$f(x) = log(1 + exp^x)$
9	Softsign	$f(x) = \left(\frac{x}{|x| + 1} \right)$
10	ReLU	$f(x) = max(0, x) = \begin{cases} x_i, & if\ x_i \geq 0 \\ 0, & if\ x_i < 0 \end{cases}$
11	LReLU	$f(x) = \alpha x + x = \begin{cases} x & if\ x > 0 \\ \alpha x & if\ x \leq 0 \end{cases}$
12	PReLU	$f(x_i) = \begin{pmatrix} x_i, & if\ x_i > 0 \\ a_i x_i, & if\ x_i \leq 0 \end{pmatrix}$
13	RReLU	$f(x_i) = \begin{pmatrix} x_{ji}, & if\ x_{ji} \geq 0 \\ a_{ji} x_{ji}, & if\ x_{ji} < 0 \end{pmatrix}$
14	SReLU	$f(x) = \begin{pmatrix} t_i^r + a_i^r(x_i - t_i^r), & x_i \geq t_i^r \\ x_i, & t_i^r > x_i > t_i^l \\ t_i^l + a_i^l\left(x_i - t_i^l\right), & x_i \leq t_i^l \end{pmatrix}$
15	ELU	$f(x) = \begin{pmatrix} x, & if\ x > 0 \\ \alpha\ exp(x) - 1, & if\ x \leq 0 \end{pmatrix}$
16	PELU	$f(x) = \begin{pmatrix} cx, & if\ x > 0 \\ \alpha\ exp^{\frac{x}{b}} - 1, & if\ x \leq 0 \end{pmatrix}$
17	SELU	$f(x) = \tau \begin{pmatrix} x, & if\ x > 0 \\ \alpha\ exp(x) - \alpha, & if\ x \leq 0 \end{pmatrix}$
18	Maxout	$f(x) = max(w_1^T x + b_1, w_2^T x + b_2)$
19	Swish	$f(x) = x \cdot sigmoid(x) = \frac{x}{1 + e^{-x}}$
20	ELiSH	$f(x) = \begin{cases} \left(\frac{x}{1 + e^{-x}} \right), & x \geq 0 \\ \left(\frac{e^x - 1}{1 + e^{-x}} \right), & x < 0 \end{cases}$
21	HardELiSH	$f(x) = \begin{cases} x \times max\left(0, min\left(1, \left(\frac{x+1}{2} \right) \right) \right), & x \geq 0 \\ (e^x - 1) \times max\left(0, min\left(1, \left(\frac{x+1}{2} \right) \right) \right), & x < 0 \end{cases}$

ReLU, including LRELU, PReLU and RReLU performs better than the ReLU but some of these functions lack theoretical justifications to support their state-of-the-art results [61].

Furthermore, the parametric AFs have been a development in emerging applications where the AF was used as a learnable parameter from the dataset, thus looks to be a promising approach in the new functions developed most recently as observed in SReLU, PELU and PReLU.

Apart from the ReLU and SiLU AFs that were developed on restricted Boltzmann machines [51], [57], and LReLU developed with neural network acoustic models [53], almost all the other AFs were developed on convolutional neural networks and tested

on classification datasets. This further demonstrated that deep learning tasks are thriving mostly in classification tasks, using convolutional neural networks.

The most notable observation on the use of AFs for DL applications is that the newer activation functions seem to outperform the older AFs like the ReLU, yet even the latest DL architectures rely on the ReLU function. This is evident in SeNet where the hidden layers had the ReLU activation function and the Sigmoid output [75]. However, current practices does not use the newly developed state-of-the-art AFs but depends on the tested and proven AFs, thereby underlining the fact that the newer activation functions are rarely used in practice.

VI. CONCLUSION

This paper provides a comprehensive summary of AFs used in deep learning and most importantly, highlights the current trends in the use of these functions in practice against the state -of-the-art research results, which until now, has not been published in any literature.

We first presented a brief introduction to deep learning and activation functions, and then outlined the different types of activation functions discussed, with some specific applications where these functions were used in the development of deep learning based architectures and systems.

The AFs have the capability to improve the learning of the patterns in data thereby automating the process of features detection and justifying their use in the hidden layers of the neural networks, and usefulness for classification purposes across domains.

These activation functions have developed over the years with the compounded activation functions looking towards the future of activation functions research. Furthermore, it is also worthy to state that there are other activation functions that have not been discussed in this literature as we focused was on the activation functions, used in deep learning applications. A future work would be to compare all these state-of-th-art functions on the award-winning architectures, using standard datasets to observe if there would be improved performance results.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[2] N. Jones, "Computer science: The learning machines,," *Nature*, vol. 505, no. 7482, pp. 146–148, 1 2014.

[3] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, 2014. [Online]. Available: https://doi.org/10.1017/atsip.2013.9

[4] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, M. Hasan, B. V. Esesn, and V. K. Asari, "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," *arXiv*, 2018. [Online]. Available: http://arxiv.org/abs/1803.01164

[5] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[6] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research," *Journal of Pharmaceutical and Biomedical Analysis*, vol. 22, no. 5, pp. 717–727, 2000. [Online]. Available: https://doi.org/10.1016/S0731-7085

[7] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989. [Online]. Available: https://doi.org/10.1162/neco.1989.1.4.541

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," 2012. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary

[9] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, 2015. [Online]. Available: https://arxiv.org/pdf/1409.1556.pdf

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions,," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2015. IEEE, 2015, pp. 1–9.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *IEEE Conf. Comput. Vis. Pattern Recognit.* IEEE, 2016, pp. 770–778. [Online]. Available: https://doi.org/10.1109/CVPR.2016.90

[12] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," *ArXiv*, vol. 9908, no. LNCS, pp. 646–661, 2016. [Online]. Available: 10.1007/978-3-319-46493-0_39;http://arxiv.org/abs/1603.09382

[13] J. Turian, J. Bergstra, and Y. Bengio, "Quadratic features and deep architectures for chunking," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics,*, vol. Companion Volume:, 2009, pp. 245–248. [Online]. Available: https://dl.acm.org/citation.cfm

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and undefined R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[15] D. J. Matas, "All you need is a good init," *ArXiv*, 2016. [Online]. Available: http://arxiv.org/abs/1511.06422

[16] P. Luo, X. Wang, W. Shao, and Z. Peng, "Understanding Regularization in Batch Normalization," *arXiv*, 2018. [Online]. Available: http://export.arxiv.org/abs/1809.00846

[17] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," in *Proceedings of the 32nd International Conference on Machine Learning, PMLR* . JMLR, 2015, pp. 448–456.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *International Conference on Artificial Intelligence and Statistics*, vol. 9. PLMR, 2010, pp. 249–256. [Online]. Available: http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf?hc_location=ufi

[19] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions," *ArXiv*, 2017. [Online]. Available: 1710.05941;http://arxiv.org/abs/1710.05941

[20] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *International Conference on Machine Learning*, 2011. [Online]. Available: http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf

[21] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. [Online]. Available: https://doi.org/10.1109/72.279181

[22] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning*, vol. 28, 2013, pp. III–1310. [Online]. Available: https://dl.acm.org/citation.cfm

[23] J. Bergstra, D. Yamins, and D. D. Cox, "Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures," in *Machine Learning research*, 2013. [Online]. Available: http://proceedings.mlr.press/v28/bergstra13.pdf

[24] C. Bircanoglu and N. Arica, "A comparison of activation functions in artificial neural networks," in *Signal Processing and Communications Applications Conference (SIU)*, vol. 26. IEEE', 2018, pp. 1–4.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *arXiv*, 2015. [Online]. Available: http://arxiv.org/abs/1502.01852

[26] S. M. Noor, J. Ren, S. Marshall, and K. Michael, "Hyperspectral Image Enhancement and Mixture Deep-Learning Classification of Corneal Epithelium Injuries," *Sensors. Multidisciplinary Digital Publishing Institute*, vol. 17, no. 11, p. 2644, 2017.

[27] T. N. Sainath, B. Kingsbury, G. Saon, H. Soltau, A. Mohamed, G. Dahl, and B. Ramabhadran, "Deep Convolutional Neural Networks for Large-scale Speech Tasks," *Neural Networks*, vol. 64, pp. 39–48, 2015. [Online]. Available: https://doi.org/10.1016/J.NEUNET.2014.08.005

[28] A. Graves, A. Mohamed, and undefined G. Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 6645–6649.

[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv*, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[30] R. Hu, P. Dollár, K. He, T. Darrell, and R. Girshick, "Learning to Segment Every Thing," *arXiv*, 2017. [Online]. Available: 10.1109/CVPR.2018.00445;http://arxiv.org/abs/1711.10370

[31] A. Karpathy and L. Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions," in *Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015, pp. 3128–3137. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Karpathy_Deep_Visual-Semantic_Alignments_2015_CVPR_paper.html

[32] P. O. Pinheiro and R. Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling," in *International Conference on Machine Learning*, 2014. [Online]. Available: http://proceedings.mlr.press/v32/pinheiro14.pdf

[33] O. Vinyals, L. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a Foreign Language," *arXiv*, 2014. [Online]. Available: 10.1146/annurev.neuro.26.041002.131047;http://arxiv.org/abs/1412.7449

[34] Y. Liu and J. Zhang, "Deep Learning in Machine Translation," in *in Deep Learning in Natural Language Processing*. Singapore: Springer, 2018, pp. 147–183.

[35] S. O. Arik, M. Chrzanowski, A. Coates, G. Diamos, A. Gibiansky, Y. Kang, and M. Shoeybi, "Deep Voice: Real-time Neural Text-to-Speech," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1702.07825

[36] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, and J. Miller, "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning," in *International Conference on Learning Representations - ICLR*, vol. 79, 2018, pp. 1094–1099. [Online]. Available: https://doi.org/10.1001/jam

[37] S. Albarqouni, C. Baur, F. Achilles, V. Belagiannis, S. Demirci, and N. Navab, "AggNet: Deep Learning From Crowds for Mitosis Detection in Breast Cancer Histology Images,," *IEEE Trans. Med. Imaging*, vol. 35, no. 5, pp. 1313–1321, 5 2016.

[38] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep Learning for Identifying Metastatic Breast Cancer," *arXiv*, 2016. [Online]. Available: 1606.05718;http://arxiv.org/abs/1606.05718

[39] A. A. Cruz-Roa, J. A. Ovalle, A. Madabhushi, and F. G. Osorio, "A Deep Learning Architecture for Image Representation," *Springer*, pp. 403–410, 2013. [Online]. Available: https://doi.org/10.1007/978-3-642-40763-5_50

[40] L. Lazimul and D.Binoy, "Fingerprint liveness detection using convolutional neural network and fingerprint image enhancement," 2017. [Online]. Available: 10.1109/ICECDS.2017.8389533

[41] H. Jung and Y. S. Heo, "Fingerprint liveness map construction using convolutional neural network," *Electronics Letters*, vol. 54, no. 9, pp. 564–566, 2018. [Online]. Available: 10.1049/el.2018.0621

[42] A. Grover, A. Kapoor, and E. Horvitz, "A Deep Hybrid Model for Weather Forecasting." New York, New York, USA: ACM Press, 2015, pp. 379–386. [Online]. Available: https://doi.org/10.1145/2783258.2783275

[43] M. Hossain, B. Rekabdar, S. J. Louis, and S. Dascalu, *Forecasting the weather of Nevada: A deep learning approach*, 2015. [Online]. Available: https://doi.org/10.1109/IJCNN.2015.7280812

[44] A. L. Xar, Y. Demir, C. Neyt, and G. Zelis, "Object recognition and detection with deep learning for autonomous driving applications,," *Modeling and Simulation International*, vol. 93, no. 9, pp. 759–769, 2017. [Online]. Available: 10.1177/0037549717709932

[45] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, *DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving*, 2015. [Online]. Available: https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Chen_DeepDriving_Learning_Affordance_ICCV_2015_paper.html

[46] B. Karlik and A. Vehbi, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," *International Journal of Artificial Intelligence and Expert Systems (IJAE)*, vol. 1, no. 4, pp. 111–122, 2011. [Online]. Available: http://www.cscjournals.org/library/manuscriptinfo.php

[47] D. L. Elliott, "A Better Activation Function for Artifical Neural Networks," 1993. [Online]. Available: https://drum.lib.umd.edu/handle/1903/5355

[48] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning,," in *Natural to Artificial Neural Computation. IWANN 1995. Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 1995, pp. 195–201.

[49] R. M. Neal, "Connectionist learning of belief networks," *Artificial Intelligence*, vol. 56, no. 1, pp. 71–113, 1992. [Online]. Available: https://doi.org/10.1016/0004-3702(92)90065-6

[50] M. Courbariaux, Y. Bengio, and J. P. David, "BinaryConnect: Training Deep Neural Networks with binary weights during propagations." *arXiv*, pp. 3123–3131, 2015. [Online]. Available: https://arxiv.org/pdf/1511.00363.pdf

[51] S. Elfwing, E. Uchibe, and K. Doya, ""Sigmoid-Weighted Linear Units for," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1702.03118

[52] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Language Modeling with Gated Convolutional Networks,," *arXiv*, 2017.

[53] A. Maas, A. Hannun, and A. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models," in *International Conference on Machine Learning (icml)*, 2013.

[54] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[55] M. Lin, Q. Chen, and S. Yan, "Network In Network," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1312.4400

[56] P. Le and W. Zuidema, "Compositional Distributional Semantics with Long Short Term Memory," *Association for Computational Linguistics*, 2015. [Online]. Available: https://doi.org/10.18653/v1/S15-1002

[57] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," Haifa, 2010, pp. 807–814. [Online]. Available: https://dl.acm.org/citation.cfm

[58] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, and G. E. Hinton, "On rectified linear units for speech processing," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 3517–3521, IEEE. https://doi.org/10.1109/ICASSP.2013.6638312.

[59] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.

[60] M. D. Zeiler and R. Fergus, "Visualizing and Understanding Convolutional Networks," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[61] B. Xu, N. Wang, H. Kong, T. Chen, and M. Li, "Empirical Evaluation of Rectified Activations in Convolution Network," *arXiv*, 2015. [Online]. Available: https://arxiv.org/abs/1505.00853

[62] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan, "Deep Learning with S-shaped Rectified Linear Activation Units,," *arXiv*, pp. 1737–1743, 2015. [Online]. Available: https://arxiv.org/pdf/1512.07030.pdf

[63] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating Second-Order Functional Knowledge for Better Option Pricing," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 13, 2001, pp. 472–478. [Online]. Available: https://papers.nips.cc/paper/1920-incorporating-second-order-functional-knowledge-for-better-option-pricing

[64] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li, "Improving deep neural networks using softplus units," in *International Joint Conference on Neural Networks (IJCNN)*, vol. 2015.   IEEE, 2015, pp. 1–4.

[65] A. Senior and X. Lei, "Fine context, low-rank, softplus deep neural networks for mobile speech recognition," in *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.   IEEE, 2014, pp. 7644–7648, IEEE. https://doi.org/10.1109/ICASSP.2014.6855087.

[66] D. A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *arXiv*, 2015. [Online]. Available: http://arxiv.org/abs/1511.07289

[67] L. Trottier, P. Giguere, and B. Chaib-draa, "Parametric Exponential Linear Unit for Deep Convolutional Neural Networks," *arXiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1605.09332v1.pdf

[68] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," *arXiv*, 2017. [Online]. Available: https://arxiv.org/pdf/1706.02515v5.pdf

[69] J. Lehman, J. Chen, J. Clune, and K. O. Stanley, "Safe mutations for deep and recurrent neural networks through output gradients."   New York, New York, USA: ACM Press, 2018, pp. 117–124. [Online]. Available: https://doi.org/10.1145/3205455.3205473

[70] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," 2013, p. 1319. [Online]. Available: https://dl.acm.org/citation.cfm

[71] L. Tóth, "Phone recognition with hierarchical convolutional deep maxout networks,," *EURASIP J. Audio, Speech, Music Process*, vol. 2015, no. 1, p. 25, 12 2015.

[72] M. Basirat and P. M. Roth, "The Quest for the Golden Activation Function," *arXiv*, 2018. [Online]. Available: https://arxiv.org/pdf/1808.00783.pdf

[73] Y. LeCun, "Backpropagation Applied to Handwritten Zip Code Recognition,," *Neural Computing*, vol. 1, no. 4, pp. 541–551, 1989.

[74] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv*, 2013. [Online]. Available: http://arxiv.org/abs/1312.6120

[75] J. Fu, H. Zheng, and T. Mei, "Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.   IEEE, 2017, pp. 4476–4484. [Online]. Available: https://doi.org/10.1109/CVPR.2017.476

[76] G. Huang, Z. Liu, L. der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.   IEEE, 2017, pp. 2261–2269. [Online]. Available: https://doi.org/10.1109/CVPR.2017.243

[77] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-Deep Neural Networks without Residuals," *arXiv*, 2016. [Online]. Available: http://arxiv.org/abs/1605.07648

[78] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.   IEEE, 1 2017, pp. 5987–5995. [Online]. Available: https://doi.org/10.1109/CVPR.2017.634

[79] V. Golkov, A. Dosovitskiy, J. I. Sperl, M. I. Menzel, M. Czisch, P. Sämann, and D. Cremers, "q-Space Deep Learning: Twelve-Fold Shorter and Model-Free Diffusion MRI Scans," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1344–1351, 2016. [Online]. Available: https://doi.org/10.1109/TMI.2016.2551324

[80] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv*, 2017. [Online]. Available: http://arxiv.org/abs/1704.04861

[81] Z. Deng, Z. Wang, and undefined S. Wang, "Stochastic area pooling for generic convolutional neural network," *Front. Artif. Intell. Appl*, vol. 285, pp. 1760–1761, 2016.

[82] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task,," *arXiv*, 2018. [Online]. Available: http://arxiv.org/abs/1804.02763