Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark

Shiv Ram Dubey[1], Satish Kumar Singh[1], Bidyut Baran Chaudhuri[2]

[1]*Computer Vision and Biometrics Laboratory, Indian Institute of Information Technology, Allahabad, India.*
[2]*Techno India University, Kolkata, India and Indian Statistical Institute, Kolkata, India.*

srdubey@iiita.ac.in, sk.singh@iiita.ac.in, bidyutbaranchaudhuri@gmail.com

Abstract

Neural networks have shown tremendous growth in recent years to solve numerous problems. Various types of neural networks have been introduced to deal with different types of problems. However, the main goal of any neural network is to transform the non-linearly separable input data into more linearly separable abstract features using a hierarchy of layers. These layers are combinations of linear and nonlinear functions. The most popular and common non-linearity layers are activation functions (AFs), such as Logistic Sigmoid, Tanh, ReLU, ELU, Swish and Mish. In this paper, a comprehensive overview and survey is presented for AFs in neural networks for deep learning. Different classes of AFs such as Logistic Sigmoid and Tanh based, ReLU based, ELU based, and Learning based are covered. Several characteristics of AFs such as output range, monotonicity, and smoothness are also pointed out. A performance comparison is also performed among 18 state-of-the-art AFs with different networks on different types of data. The insights of AFs are presented to benefit the researchers for doing further research and practitioners to select among different choices. The code used for experimental comparison is released at: `https://github.com/shivram1987/ActivationFunctions`.

1. Introduction

In recent years, deep learning has shown a tremondous growth to solve the challenging problems such as object detection [1], semantic segmentation [2], person re-identification [3], image retrieval [4], anomaly detection [5], skin disease diagnosis [6], and many more. Various types of neural networks have been defined in deep learning to learn abstract features from data, such as Multilayer Perceptron (MLP) [7], Convolutional Neural Networks (CNN) [8], Recurrent Neural Networks (RNN) [9], and Generative Adversarial Networks (GAN) [10]. The important aspects of neural networks include weight initialization [11], loss functions [12], different layers [13], overfitting [14], and optimization [15].

The activation functions (AFs) play a very crucial role in neural networks [16] by learning the abstract features through nonlinear transformations. Some common properties of the AFs are as follows: a) it should add the non-linear curvature in the optimization landscape to improve the training convergence of the network; b) it should not increase the computational complexity of the model extensively; c) it should not hamper the gradient flow during training; d) it should retain the distribution of data to facilitate the better training of the network. Several AFs have been explored in recent years for deep learning to achieve the above mentioned properties. This survey is dedicated to the developments in the area of AFs in neural networks. The insights of the different AFs are presented along with the reasoning to benefit the deep learning community. The major contributions of this survey are outlined as follows:

1. This survey provides a detailed classification for a wide range of AFs. It also includes the AFs very comprehensively, including Logistic Sigmoid/Tanh, Rectified Unit, Exponential Unit, and Adaptive AFs.

2. This survey enriches the reader with the state-of-the-art AFs with analysis from various perspectives. It specifically covers the progress in AFs for deep learning.

3. This survey also summarizes the AFs with brief highlights and important discussions to depict its suitability for different types of data (Refer to Table 6).

4. This survey is compared with the existing survey and performance analysis to show its importance (Refer to Table 7).

5. This paper also presents the performance comparisons on 4 benchmark datasets of different modalities using 18 state-of-the-art AFs with different types of networks (Refer to Tables 8, 9 and 11).

The evolution of AFs is illustrated in Section 2. The progress in Logistic Sigmoid and Tanh, rectified, exponential, adaptive and miscellaneous AFs are summarized in Section 3, 4, 5, 6, and 7, respectively. Some aspects of AFs are discussed in Section 8. A comprehensive performance analysis is conducted in Section 9. A summary with conclusions and recommendations is provided in Section 10.
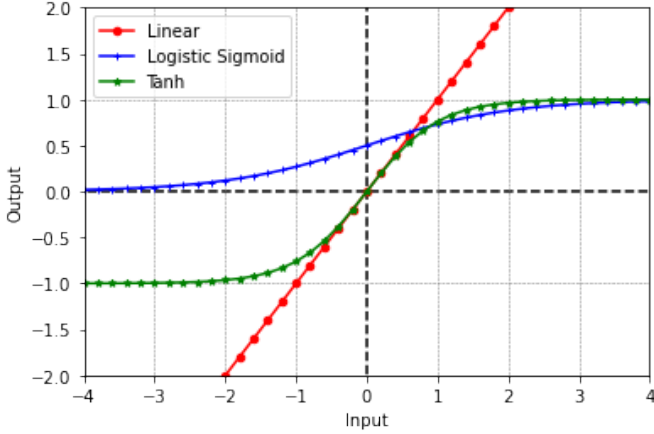
Figure 1: An illustration of Linear, Logistic Sigmoid and Tanh AFs.

Figure 2: Classification of activation functions.

2. Evolution of Activation Functions

A linear function can be thought of as a simple AF which outputs $c \times x$ for input x with c as a constant. The linear AF is illustrated in Fig. 1 for $c = 1$, i.e., identity function. Note that the linear AF does not add non-linearity into the network. However, the non-linearity needs to be introduced in the neural networks. Otherwise, a neural network produces the output as a linear function of inputs inspite of having several layers. Moreover, in practice data is generally not linearly separable; hence, the non-linear layers help to project the data in non-linear fashion in feature space which can be used with different objective functions. This section provides an overview of the evolution of AFs for deep learning. A classification is presented in Fig. 2 in terms of the different properties and characteristic types.

Logistic Sigmoid/Tanh Unit Based Activation Functions: In order to introduce the non-linearity into the neural networks, the Logistic Sigmoid and Tanh AFs have been used in the early days. The firing of bilogical neurons was the motivation of using the Logistic Sigmoid and Tanh AFs with artificial neurons. The Logistic Sigmoid AF is a very popular and traditional non-linear function. It is given as,

$$\text{Logistic Sigmoid}(x) = \frac{1}{1 + e^{-x}}. \tag{1}$$

This AF squashes the output between [0, 1] as shown in Fig. 1. The output of the Logistic Sigmoid function is saturated for higher and lower inputs, which leads to vanishing gradient problem. The vanishing gradient problem depicts to a scenario where the gradient of objective function w.r.t. a parameter becomes very close to zero and leads to almost no update in the parameters during the training of the network using stochastic gradient descent technique. Hence, the training is almost killed under vanishing gradient scenario. Moreover, the output not following a zero-centric nature leads to poor convergence. The Tanh function has also been used as the AF in neural networks. It is similar to the Logistic Sigmoid function while exhibiting the zero centric property as depicted in Fig. 1. The Tanh func-
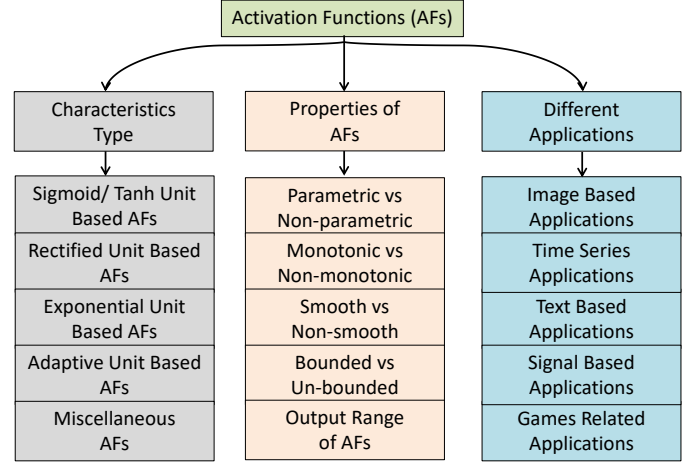
tion is written as,

$$\text{Tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2}$$

The Tanh function also squashes the inputs, but in $[-1, 1]$. The drawbacks of Logistic Sigmoid function such as vanishing gradient and computational complexity also exist with Tanh function. The Logistic Sigmoid and Tanh AFs majorly suffer from vanishing gradient. Several improvements have been proposed based on the Logistic Sigmoid and Tanh AFs which are described in Section 3 in detail.

Rectified Linear Unit Based Activation Functions: The saturated output and increased complexity are the key limitations of above-mentioned Logistic Sigmoid and Tanh based AFs. The Rectified Linear Unit (ReLU) [17] has become the state-of-the-art AF due to its simplicity and improved performance. The ReLU was also used in the AlexNet model [8]. Various variants of ReLU have been investigated by tackling its drawbacks, such as non-utilization of negative values, limited non-linearity and unbounded output, as detailed in Section 4.

Exponential Unit Based Activation Functions: The major problem faced by the Logistic Sigmoid and Tanh based AFs is with its saturated output for large positive and negative input. Similarly, the major problem with ReLU based AFs is with the under-utilization of negative values leading to vanishing gradient. In order to cope up with these limitations the exponential function based AFs have been used in the literature. The Exponential Linear Unit (ELU) [27] based AF utilizes the negative values with the help of the exponential function. Several AFs have been introduced in the literature as the ELU variants which are presented in Section 5 in detail.

Learning/Adaptive Activation Functions: Most of the Sigmoid, Tanh, ReLU, and ELU based AFs are designed manually which might not be able to exploit the data complexity. The learning based adaptive AFs are the recent trends. This class of AFs contains learnable parameters, e.g. Adaptive Piecewise Linear (APL) [28] and Swish [29] AFs contain two and one learnable parameters, respectively. Recently, several learning based AFs have been proposed as illustrated in Section 6.

Table 1: Advantage and disadvantage of primary AFs.

AFs	Diminishing gradients	Limited non-linearity	Optimization difficulty	Lack of adaptibility	Computational inefficiency
Sigmoid	Yes	No	Yes	Yes	Yes
Tanh	Yes	No	Partial	Yes	Yes
ReLU	Partial	Yes	Partial	Yes	No
ELU	No	Partial	No	Yes	Partial
APL	No	Partial	No	No	No
Swish	No	Partial	No	No	Partial

Table 2: Summary of Logistic Sigmoid and Tanh based activation functions.

Name of AF	Parametric	Monotonic	Smooth	Bounded
Logistic Sigmoid	No	Yes	Yes	Yes
Tanh	No	Yes	Yes	Yes
Scaled Tanh (sTanh), 1998 [18]	Yes	Yes	Yes	Yes
Rectified Hyperbolic Secant (ReSech), 2016 [19]	No	No	Yes	Yes
Scaled Sigmoid (sSigmoid), 2016 [20]	No	Yes	Yes	Yes
Penalized Tanh (pTanh), 2016 [20]	No	Yes	No	Yes
Hexpo, 2017 [21]	No	Yes	Yes	Yes
Improved Sigmoid (ISigmoid), 2018 [22]	No	Yes	Yes	No
Sigmoid-Weighted Linear Units (SiLU), 2018 [23]	No	No	Yes	For negative inputs
Linearly Scaled Hyperbolic Tangent (LiSHT), 2019 [24]	No	No	Yes	No
Elliott, 2019 [25]	No	Yes	Yes	Yes
Soft-Root-Sign (SRS), 2020 [26]	Yes	No	Yes	Yes

Miscellaneous Activation Functions: In recent years, many other AFs have also been investigated as presented in Section 7. These activations include Softplus units, probabilistic functions, polynomial functions, and kernel functions.

Table 1 highlights the advantage and disadvantage of the primary AFs in terms of the diminishing gradients, limited non-linearity, optimization difficulty, computational inefficiency and lack of adaptibility. It can be noticed that the Tanh function is computationally inefficient because it involves the computation of exponential multiple times [30]. However, in implementation it can be computed using single exponential with the help of Sigmoid function. These limitations in the existing AFs have been the driving factors for the development of recent AFs as surveyed in the further sections of this paper.

3. Logistic Sigmoid and Tanh Based AFs

The traditional AFs such as Logistic Sigmoid and Tanh were used very extensively in the early days of neural networks. However, these AFs had shown the hurdle to train the deep networks due to their saturated output. Several attempts have also been made to improve these AFs for different networks. Table 2 presents the comparison of Logistic Sigmoid and Tanh based AFs in terms of their properties including parametric, monotonic, smooth and bounded.

In order to tackle the limited output range and zero gradient problems of Tanh, a scaled Hyperbolic Tangent (sTanh) is used in [18] which is defined as,

$$sTanh(x) = A \times Tanh(B \times x) \tag{3}$$

with the output range in $[-A, A]$. A Parametric Sigmoid Function (PSF) is proposed as a continuous, differentiable, and bounded function as,

$$PSF(x) = \frac{1}{(1 + e^{-x})^m} \tag{4}$$

where m is a hyperparameter [31]. The gradient flow is improved for the higher value of m. The sum of shifted log-sigmoid is also explored as an AF [32] which retains the symmetry in the generated features. The Rectified Hyperbolic Secant (ReSech) AF is differentiable, symmetric, and bounded [19] which is given as,

$$ReSech(x) = x \times Sech(x) \tag{5}$$

with the output range in $[-1, 1]$. However, it exhibits the vanishing gradient problem due to saturating behavior for both large positive and large negative inputs. The training of deep networks become difficult due to the uniform slope of the Logistic Sigmoid and Tanh AFs near the origin [20]. To minimize this limitation, the Scaled Sigmoid (sSigmoid) is defined as,

$$sSigmoid(x) = (4 \times Sigmoid(x) - 2) \tag{6}$$

with the output range in $[-2, 2]$ and the Penalized Tanh (pTanh) is defined as,

$$pTanh(x) = \begin{cases} Tanh(x), & x \geq 0 \\ a \times Tanh(x), & x < 0 \end{cases} \tag{7}$$

with the output range in $[-a, 1]$ where $a \in (0, 1)$. However, sSigmoid and pTanh AFs also suffer from the vanishing gradient problem. It is noticed that the pTanh AF performs better for Natural Language Processing (NLP) tasks [33].

A noisy AF is defined to overcome the vanishing gradient problem [48]. Due to the added noise the gradients may flow easily even in the saturating regime. The vanishing gradient

Table 3: Summary of Rectified Linear Unit based activation functions.

Name	Parametric	Monotonic	Smooth	Bounded
Rectified Linear Unit (ReLU), 2010 [17]	No	Yes	No	For negative inputs
Leaky ReLU (LReLU), 2013 [34]	No	Yes	No	No
Parametric ReLU (PReLU), 2015 [35]	Yes	Yes	No	No
Randomized ReLU (RReLU), 2015 [35]	No	Yes	No	No
Concatenated ReLU (CReLU), 2016 [36]	No	Yes	No	For negative inputs
Bounded ReLU (BReLU), 2016 [37]	No	Yes	No	Yes
Parametric Tanh Linear Unit (PTELU), 2017 [38]	Yes	Yes	Yes	For negative inputs
Flexible ReLU (FReLU), 2018 [39]	Yes	Yes	No	For negative inputs
Elastic ReLU (EReLU), 2018 [40]	No	Yes	No	For negative inputs
Randomly Translational ReLU (RTReLU), 2018 [41]	No	Yes	No	For negative inputs
Dual ReLU (DualReLU), 2018 [42]	No	Yes	No	No
Paired ReLU (PairedReLU), 2018 [43]	Yes	Yes	No	No
Average Biased ReLU (ABReLU), 2018 [44]	No	Yes	No	For negative inputs
Natural-Logarithm (NLReLU), 2019 [45]	No	Yes	No	For negative inputs
Multi-bin Trainable Linear Units (MTLU), 2019 [46]	Yes	No	No	No
Lipschitz ReLU (L-ReLU), 2020 [47]	Yes	Depends upon ϕ and η	Depends upon ϕ and η	Depends upon ϕ and η

problem is minimized by the Hexpo function [21] which is similar to Tanh with a scaled gradient. It is given as,

$$Hexpo(x) = \begin{cases} -a \times (e^{-x/b} - 1), & x \geq 0 \\ c \times (e^{x/d} - 1), & x < 0 \end{cases} \quad (8)$$

in the output range of $[-c, a]$. The output of the sigmoid function is multiplied with its input in sigmoid-weighted linear unit (SiLU) AF [23] as

$$SiLU(x) = x \times Sigmoid(x) \quad (9)$$

in the output range of $(-0.5, \infty)$. At the same time an improved logistic Sigmoid (ISigmoid) AF [22] is proposed to solve the vanishing gradient problem of Sigmoid with the help of a piecewise combination of sigmoidal and linear functions. It is defined as,

$$ISigmoid(x) = \begin{cases} \alpha \times (x - a) + Sigmoid(a), & x \geq a \\ Sigmoid(x), & -a < x < a \\ \alpha \times (x + a) + Sigmoid(a), & x \leq -a \end{cases} \quad (10)$$

in the output range of $(-\infty, \infty)$. The Linearly scaled hyperbolic tangent (LiSHT) AF scales the Tanh in a linear fashion to overcome the vanishing gradient issue [24]. The LiSHT can be defined as,

$$LiSHT(x) = x \times Tanh(x) \quad (11)$$

in the output range of $[0, \infty)$. The LiSHT function is symmetric, but is has the shortcoming of including unbounded and non-negative outputs only. The Elliott AF [25] is similar to Sigmoid function in terms of the characteristics diagram and defined as,

$$Elliott(x) = \frac{0.5 \times x}{1 + |x|} + 0.5 \quad (12)$$

in the output range of $[0, 1]$. The Soft-Root-Sign (SRS) AF [26] is defined as,

$$SRS(x) = \frac{x}{\frac{x}{\alpha} + e^{-x/\beta}} \quad (13)$$

in the output range of $[\frac{\alpha \times \beta}{\beta - \alpha \times e}, \alpha]$ where α and β are the learnable parameters. The use of additional parameters increases the complexity of the SRS function. Most of the variants of Sigmoid/Tanh AFs have tried to overcome the vanishing gradient issue. However, this issue is still present in most of these AFs.

4. Rectified Activation Functions

A summary of rectified AFs is illustrated in Table 3. Rectified Linear Unit (ReLU) is a simple function which is the identity function for positive input and zero for negative input and given as,

$$ReLU(x) = max(0, x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}. \quad (14)$$

Hence, the range of ReLU is $[0, \infty)$. The gradient for positive and negative inputs is one and zero, respectively. The ReLU function solves the problem of computational complexity of the Logistic Sigmoid and Tanh functions. The downside of ReLU is with the vanishing gradient problem for the negative inputs. In spite of having the vanishing gradient problem, the ReLU AF has been used very extensively with the deep learning models. The advancements in ReLU based AFs are discussed in the rest of this section.

4.1. On the Non-utilization of Negative Values of ReLU

Vanishing gradient is the main problem with ReLU AF which is caused due to the non-utilization of negative values. A Leaky Rectified Linear Unit (LReLU) is the extension of ReLU by utilizing the negative values [34]. The LReLU is defined as,

$$LReLU(x) = \begin{cases} x, & x \geq 0 \\ 0.01 \times x, & x < 0 \end{cases} \quad (15)$$

in the output range of $(-\infty, \infty)$. The LReLU has been used in many applications with promising performance. One major problem associated with LReLU is the finding of the right slope in linear function for negative inputs. Different slopes might be suited for different problems and different networks.

Thus, it is extended to Parametric ReLU (PReLU) by considering the slope for negative input as a trainable parameter [35]. The PReLU is given as,

$$PReLU(x) = \begin{cases} x, & x \geq 0 \\ p \times x, & x < 0 \end{cases} \quad (16)$$

in the output range of $(-\infty, \infty)$ where p is the trainable parameter. However, it can lead to overfitting easily which is the downside of PReLU. The Maxout layer, which computes the maximum of several linear units, is also used as AF [49]. Both ReLU and Leaky ReLU can be seen as the special cases of Maxout. The randomized ReLU (RReLU) considers the slope of LReLU randomly during training sampled from an uniform distribution $U(l, u)$ [50]. The RReLU is defined as,

$$RReLU(x) = \begin{cases} x, & x \geq 0 \\ R \times x, & x < 0 \end{cases} \quad (17)$$

in the output range of $(-\infty, \infty)$ where $R \sim U(l, u)$, $l < u$ and $l, u \in [0, 1)$. It uses a deterministic value $x/\left(\frac{l+u}{2}\right)$ during test time.

The ReLU is not able to utilize the potential useful information from the negative values. In most of the networks, the feature map given as the input to AF is dense near zero. Thus, a small jitter in the rectification point can lead to difficulty in training. Concatenated ReLU (CReLU) [36] concatenates the ReLU's output over original input and negated input. The CReLU can be given as,

$$CReLU(x) = [ReLU(x), ReLU(-x)] \quad (18)$$

in the output range of $[0, \infty)$. The CReLU is derived from the fact that the lower layer kernels in CNN models form pairs with opposite phases. The shifting of the feature map with multiple biases is also performed before the ReLU layer [51]. However, it increases the model complexity as more ReLUs are required. A Parametric Tan Hyperbolic Linear Unit (P-TELU) is also used as an AF [38]. The P-TELU is defined as,

$$PTELU(x) = \begin{cases} x, & x \geq 0 \\ \alpha \times \mathrm{Tanh}(\beta \times x), & x < 0 \end{cases} \quad (19)$$

in the output range of $[-\alpha, \infty)$ where $\{\alpha, \beta\} \geq 0$ are the learnable parameters.

The Flexible ReLU (FReLU) [39] captures the negative values with a rectified point which is considered as trainable in the Shifted ReLU [39]. The FReLU is given as,

$$FReLU(x) = ReLU(x) + b \quad (20)$$

in the output range of $[b, \infty)$. A similar arrangement is also followed by Random Translation ReLU (RTReLU) [41] by utilizing an offset, sampled from a Gaussian distribution, given as,

$$RTReLU(x) = \begin{cases} x + a, & x + a > 0 \\ 0, & x + a \leq 0 \end{cases} \quad (21)$$

in the output range of $[0, \infty)$ where a is a random number. At test time, the offset is set to zero. A data dependent Average Biased ReLU (AB-ReLU) [44] is also investigated to tackle the negative values by a horizontal shifting based on the average of features. The ABReLU can be written as,

$$ABReLU(x) = \begin{cases} x - \beta, & x - \beta \geq 0 \\ 0, & x - \beta < 0 \end{cases} \quad (22)$$

having the output range in $[0, \infty)$ where β is computed as the average of input activation map to the activation function. The batch dependent threshold for the ReLU is used by the Dynamic ReLU (D-ReLU) [60]. The Dual ReLU (DualReLU) [42] is a two dimensional AF for recurrent neural networks. The DualReLU is given as,

$$DualReLU(a, b) = \max(0, a) - \max(0, b) \quad (23)$$

in the output range of $(-\infty, \infty)$ where a and b are the inputs in different dimensions. Similar to the CReLU, the PairedReLU AF is used for image super-resolution [43]. The PairedReLU is given as,

$$PairedReLU(x) = [\max(s \times x - \theta, 0), max(s_p \times x - \theta_p, 0)] \quad (24)$$

in the output range of $(-\infty, \infty)$. However, the computational complexity of PairedReLU is increased as compared to CReLU. In another attempt, V-shaped ReLU (vReLU) AF [61] is defined as,

$$vReLU(x) = \begin{cases} x, & x \geq 0 \\ -x, & x < 0 \end{cases} \quad (25)$$

having the output range in $[0, \infty]$. The vReLU activation function suffers from the non-symmetric output. The SignReLU AF utilizes the negative values using the Softsign function [62]. The positive part of SignReLU is the same as the ReLU.

A Displaced ReLU (DisReLU) [63] is designed as a generalization of Shifted ReLU [39]. The DisReLU displaces the rectification point to consider the negative values, given as,

$$DisReLU(x) = \begin{cases} x, & x \geq -\delta \\ -\delta, & x < -\delta \end{cases} \quad (26)$$

having the output range in $[-\delta, \infty]$. A Bendable Linear Unit (BLU) AF is investigated as,

$$BLU(x) = \beta \times (\sqrt{x^2 + 1} - 1) + x \quad (27)$$

where $-1 \leq \beta \leq 1$ is a learnable parameter to adapt the shape between the identity function and a rectifier function [64]. A Lipschitz ReLU (L-ReLU) AF uses the piecewise linear functions to model the degree of presence and the degree of absence of features [47]. The L-ReLU is defined as,

$$L\text{-}ReLU(x) = \begin{cases} \max(\phi(x), 0), & x \geq 0 \\ \min(\eta(x), 0), & x < 0 \end{cases} \quad (28)$$

where ϕ and η are non-linear functions. Moreover, the range of L-ReLU also depends upon the values of ϕ and η functions.

Table 4: Summary of Exponential Linear Unit based activation functions.

Name	Parametric	Monotonic	Smooth	Bounded
Exponential Linear Unit (ELU), 2016 [27]	Yes	Yes	Yes	For negative inputs
Scaled ELU (SELU), 2017 [52]	Yes	Yes	Yes	For negative inputs
Continuously Differentiable ELU (CELU), 2017 [53]	Yes	Yes	No	For negative inputs
Parametric ELU (PELU), 2017 [54]	Yes	Yes	No	For negative inputs
Multiple PELU (MPELU), 2018 [55]	Yes	Yes	No	For negative inputs
Fast ELU (FELU), 2019 [56]	Yes	Yes	No	For negative inputs
Parametric Rectified Exponential Unit (PREU), 2019 [57]	Yes	No	Yes	For negative inputs
Elastic ELU (EELU), 2020 [58]	Yes	Yes	No	For negative inputs
Parametric Deformable ELU (PDELU), 2020 [59]	Yes	Yes	Yes	For negative inputs

4.2. On the Limited Non-linearity of ReLU

S-shaped ReLU (SReLU) increases the non-linearity in ReLU by combining three linear functions with four learnable parameters [65]. On a similar line, Multi-bin Trainable Linear Unit (MTLU) [46] considers multiple bins to increase the non-linear capacity. The MTLU can be written as,

$$MTLU(x) = \begin{cases} a_0 \times x + b_0, & x \leq c_0 \\ a_k \times x + b_k, & c_{k-1} < x \leq c_k \\ ... \\ a_K \times x + b_K, & c_{K-1} < x \end{cases} \quad (29)$$

having the output range in $(-\infty, \infty)$. The number of bins and the range of bins are the hyperparameters, whereas the linear function of a bin is trainable (i.e., $a_0, ..., a_K$ $b_0, ..., b_K$ are the learnable parameters). The non-differentiable nature at multiple points is the drawback of the MTLU. An Elastic ReLU (EReLU) considers a slope randomly drawn from a uniform distribution during the training for the positive inputs to control the amount of non-linearity [40]. The EReLU is defined as,

$$EReLU(x) = max(R \times x, 0) \quad (30)$$

in the output range of $[0, \infty)$ where R is a random number. At the test time, the EReLU becomes the identity function for positive inputs. The Linearized Sigmoidal Activation (LiSHA) function considers three linear functions to increase the non-linearity characteristics [66]. It is also extended to adaptive linear sigmoidal AF by learning the slope of upper and lower linear functions. The ReLU is combined with Tanh as Rectified Linear Tanh (ReLTanh) [67] to increase the non-linearity of ReLU and to overcome the vanishing gradient problem of Tanh. However, the ReLTanh is unbounded in both the positive and negative directions. Natural-Logarithm ReLU (NLReLU) modifies the ReLU's output for positive inputs using the logarithm function to increase the degree of nonlinearity [45]. The NLReLU is defined as,

$$NLReLU(x) = \ln(\beta \times max(0, x) + 1.0) \quad (31)$$

having the output range in $[0, \infty)$ where β is a constant. The NLReLU does not affect the negative regime, thus suffers from vanishing gradient. The concept of Leaky ReLU (LReLU) is further improved to Dynamic ReLU [68] by considering a mean square error (MSE) based additional hyperparameter. Thus, it can control the slope of the Dynamic ReLU in every epoch based on the convergence. A Piecewise Linear Unit (PLU) [69] is defined as,

$$PLU(x) = max(\alpha \times (x + c) - c, min(\alpha \times (x - c) + c, x)) \quad (32)$$

having the output range in $[-\infty, +\infty]$, where α and c are the constants. Basically, the PLU activation function consists of three linear functions in pieces, but continuous. Hence, it avoids the saturation and leads to a good amount of gradient flow through the activation function during backpropagation in order to resolve the vanishing gradient problems of ReLU and Tanh. However, the PLU activation is unbounded in both positive and negative directions.

4.3. On the Unbounded Output of ReLU

The unbounded outputs of ReLU and many of its variants may lead to training instability. Moreover, the bounded AF is needed for the dedicated hardware based embedded system applications. ReLU is extended to Bounded ReLU (BReLU) [37] defined as,

$$BReLU(x) = min(max(0, x), A) \quad (33)$$

having the output range in $[0, A]$). The training stability is improved in BReLU due to two rectifications (i.e., at 0 and A). ReLU is a common choice in practice in deep learning. ReLU based AFs are generally efficient. The major drawbacks of ReLU, such as gradient diminishing for negative inputs, limited non-linearity and unboundedness, are improved in the different AFs. However, the ReLU variants are not able to resolve all the issues of ReLU.

5. Exponential Activation Functions

The exponential AFs tackle the gradient diminishing problem of ReLU. Table 4 lists the properties of the exponential AFs. The Exponential Linear Unit (ELU) [27] is given as,

$$ELU(x) = \begin{cases} x, & x > 0 \\ \alpha \times (e^x - 1), & x \leq 0 \end{cases} \quad (34)$$

having the output range in $[-1, \infty)$ where α is a learnable parameter. The ELU function exhibits all the benefits of the ReLU function. The ELU is differentiable, saturates for large negative inputs and reduces the bias shift. The negative saturation regime of ELU adds some robustness to noise as compared to the Leaky ReLU and Parametric ReLU. The ELU is extended

to Scaled ELU (SELU) [52] by using a scaling hyperparameter to make the slope larger than one for positive inputs. The SELU can be defined as,

$$SELU(x) = \lambda \times \begin{cases} x, & x > 0 \\ \alpha \times (e^x - 1), & x \leq 0 \end{cases} \quad (35)$$

having the output range in $[-\lambda, \infty)$ where α is a hyperparameter. Basically, the SELU induces self-normalization to automatically converge towards zero mean and unit variance. The Parametric ELU (PELU) [54] changes the saturation point and exponential decay and also regulates the slope of the linear function for the positive inputs for differentiability. The PELU AF can be written as,

$$PELU(x) = \lambda \times \begin{cases} \frac{a}{b} \times x, & x \geq 0 \\ a \times (e^{x/b} - 1), & x < 0 \end{cases} \quad (36)$$

having $[-a, \infty)$ output range, where a and b are the trainable parameters. The parametric ELU is also explored in Continuously differentiable ELU (CELU) [53] for the negative inputs. The CELU is given as,

$$CELU(x) = \begin{cases} x, & x \geq 0 \\ \alpha \times (e^{x/\alpha} - 1), & x < 0 \end{cases} \quad (37)$$

having the output range in $[-\alpha, \infty)$ where α is a learnable parameter. The PELU is also extended to multiple PELU (MPELU) [55] by using two learnable parameters to represent MPELU as either rectified, exponential or combined. The MPELU can be expressed as,

$$MPELU(x) = \begin{cases} x, & x > 0 \\ \alpha_c \times (e^{\beta_c \times x} - 1), & x \leq 0 \end{cases} \quad (38)$$

having the output range in $[-\alpha_c, \infty)$, where α_c and β_c are the trainable parameters.

A soft exponential AF interpolates between the exponential, linear and logarithmic functions using the trainable parameter [70]. A Shifted ELU (ShELU) AF is also explored as a locally optimal function [71]. A Parametric Rectified Exponential Unit (PREU) [57] is designed as,

$$PREU(x) = \begin{cases} \alpha \times x, & x > 0 \\ \alpha \times x \times e^{\beta \times x}, & x \leq 0 \end{cases} \quad (39)$$

having the output range in $[-1, \infty)$, where α and β are the trainable parameters. The PREU utilizes the negative information near to zero effectively. The efficiency of ELU is improved in Fast ELU (FELU) AF [56] with the help of the simple displacement bits and integer algebra operations. The FELU is defined as,

$$FELU(x) = \begin{cases} x, & x > 0 \\ \alpha \times (e^{x/\ln(2)} - 1), & x \leq 0 \end{cases} \quad (40)$$

having the output range in $[-\alpha, \infty)$ with α as a learnable parameter. Recently, the properties of ELU and RELU have been utilized to design an Elastic ELU (EELU) AF [58]. The EELU is defined as,

$$EELU(x) = \begin{cases} k \times x, & x > 0 \\ \alpha \times (e^{\beta \times x} - 1), & x \leq 0 \end{cases} \quad (41)$$

having the output range in $[-\alpha, \infty)$ where α and β are the trainable parameters. The EELU preserves a small non-zero gradient for the negative input and exhibits an elastic slope for the positive input. A Parametric Deformable ELU (PDELU) AF tries to shift the mean value of output closer to zero using the flexible map shape [59]. The PDELU is defined as,

$$PDELU(x) = \begin{cases} x, & x > 0 \\ \alpha \times ([1 + (1 - t) \times x]^{\frac{1}{1-t}} - 1), & x \leq 0 \end{cases} \quad (42)$$

having the output range in $[-1, \infty)$ where α is a learnable parameter. A ReLU-Memristor-like AF (RMAF) [72] uses two hyperparameters to have ReLU like shape for positive input and to give more importance to the negative values near to zero. An Exponential Linear Sigmoid SquasHing (ELiSH) is defined in [73] as,

$$ELiSH(x) = \begin{cases} x/(1 + e^{-x}), & x \geq 0 \\ (e^x - 1)/(1 + e^{-x}), & x < 0 \end{cases} \quad (43)$$

Moreover, it is also extended to HardELiSH which is a multiplication of HardSigmoid and Linear in the positive part and HardSigmoid and ELU in the negative part. Here, HardSigmoid is defined as,

$$HardELish(x) = max(0, min(1, (x + 1)/2)). \quad (44)$$

The ELU based AFs exploit the negative inputs without compromising with the non-linearity. Some ELU variants also modify the function for positive inputs to make it bounded.

6. Learning/Adaptive Activation Functions

Most of the aforementioned AFs are not adaptive and might not be able to adjust based on the dataset complexity. This problem is tackled using learning/adaptive AFs as summarized in Table 5. Some of the earlier mentioned AFs are also adaptive, such as PReLU [57], SReLU [65], PTELU [38], MTLU [46], PELU [54], MPELU [55], PREU [57], EELU [58], PDELU [59], SRS [26], etc.

The Adaptive Piecewise Linear (APL) is defined as a sum of hinge-shape functions [28]. It is given as,

$$APL(x) = max(0, x) + \sum_{s=1}^{S} a_s \times max(0, b_s - x), \quad (45)$$

where a and b are the trainable parameters and S is a hyperparameter representing the number of hinges. The output range of APL is $[0, \infty)$. Due to the trainable parameters, different neurons can learn different AFs.

Table 5: Summary of adaptive and learning based activation functions.

Name	Parametric	Monotonic	Smooth	Bounded
Adaptive Piecewise Linear Unit (APL), 2015 [28]	Yes	No	No	No
Spline AF (SAF), 2016 [74]	Yes	Yes	Yes	No
Bi-Modal Derivative Adaptive Activation (BDAA), 2017 [75]	Yes	Yes	Yes	Yes
Adaptive AF (AAF), 2018 [76]	Yes	Yes	No	No
Swish, 2018 [29]	Yes	No	Yes	No
ESwish, 2018 [77]	Yes	No	Yes	No
Trainable AF (TAF), 2018 [78]	Yes	No	Yes	No
Self-Learnable AF (SLAF), 2019 [79]	Yes	No	Yes	No
Mexican ReLU (MeLU), 2019 [80]	Yes	No	No	No

Ramachandran et al. [29] have performed an automatic search, which resulted in a Swish AF. It is defined as,

$$S wish(x) = x \times S igmoid(\beta \times x) \qquad (46)$$

where β is a learnable parameter. The output range of Swish is $(-\infty, \infty)$. Based on the learnt value of β the shape of the Swish AF is adjusted between the linear and ReLU functions. The smaller and higher values of β lead towards the linear and ReLU functions, respectively. Thus, it can control the amount of non-linearity based on the dataset and network complexity. Swish is also extended to E-Swish by multiplying the Swish with a learnable parameter to control the slope in the positive direction [77]. The E-Swish is defined as,

$$ES wish(x) = \beta \times x \times S igmoid(x) \qquad (47)$$

having the output the range in $(-\infty, \infty)$ and β is trainable parameter. A flatten-T Swish considers zero function for negative inputs similar to the ReLU [81]. The Adaptive Richard's Curve weighted Activation (ARiA) is also motivated from Swish and replaces the sigmoidal function with Richard's Curve [82]. The ARiA AF uses five hyper-parameters to control the shape of the non-linearity.

The basic AFs are combined with learnable weights in adaptive AFs [76]. The Adaptive AF (AAF) designed over PReLU [35] and PELU [54] is given as,

$$AAF(x) = \sigma(w \times x) \times PRELU(x) + (1 - \sigma(w \times x)) \times PELU(x) \qquad (48)$$

having the output range in $[0, 1]$, where σ is the sigmoidal function and w is a learnable parameter. In practice, AAF is costly as multiple AFs are involved. In [83], the AF for each neuron is selected from a library of AFs. In [84], different combinations of the identity function, ReLU, and Tanh are learnt automatically. In another attempt, an Adaptive Blending Unit (ABU) is defined to allow the networks to learn its preferred AFs [85]. The ABU combines a set of AFs with trainable weights. A Lookup Table Unit (LuTU) function [86] uses a single period cosine mask based smoothing and linear interpolation using a set of anchor points. Activation ensembles are used at each layer in [87] with the contribution of each AF controlled by the trainable weights. Similarly, the Self-Learnable AF (SLAF) computes the sum of the different functions in an ensemble with the learnt coefficients [79]. The SLAF can be expressed as,

$$SLAF(x) = \sum_{i=0}^{N-1} a_i \times x^i \qquad (49)$$

in the output range of $(-\infty, \infty)$, where a_i is the trainable parameter. A Mexican ReLU (MeLU) AF is proposed in [80] by using a "Mexican hat type" function and given as,

$$MeLU(x) = PReLU(x) + \sum_{j=1}^{k} c_j \times \max(\lambda_j - |x - a_j|, 0) \qquad (50)$$

in the output range of $(-\infty, \infty)$, where c_j is the trainable parameter and λ_j & a_j are the real numbers.

A cubic spline interpolation is also used to learn the AF from data [74] which is given as,

$$SAF(x) = \Phi(s; \mathbf{q}) \qquad (51)$$

having the output range in $(-\infty, \infty)$ where $\Phi(.)$ is parameterized by a vector $\mathbf{q}$ cubic in nature. Fourier series basis expansion is used for nonparametrically learning AFs (NPF) [88]. Hyperactivations utilize a hypernetwork on top of an activation network, which are used to explore the AFs search space [89]. A shallow neural network is used in the activation network to produce the output for each input, whereas a neural network is used in the hypernetwork to produce weights for another network. A bi-modal derivative adaptive activation (BDAA) function uses twin maxima derivative sigmoidal function [75] by controlling the maxima's position with an adaptive parameter. The BDAA is given as,

$$BDAA(x) = \frac{1}{2} \times \left(\frac{1}{1 + e^{-x}} - \frac{1}{1 + e^{-x-a}} \right) \qquad (52)$$

in the output range of $[0, 1]$ where a is a learnable parameter. The authors have exploited the Bi-modal derivatives on four AFs. Linear regression is used in [78] to train AF for each neuron which results in different AFs for the different neurons. The TAF is defined as,

$$TAF(x) = \sqrt{(x - a)^2 + b^2} \qquad (53)$$

in the output range of $[b, \infty)$, where a and b are the trainable parameters. Recently, a trainable parameter was used in different non-adaptive AFs such as Sigmoid, Tanh, and ReLU to make it adaptive [90].

The adaptive and trainable AFs are the recent trend to adjust the non-linearity based on the data and network complexity. However, the minimal burden is increased in terms of the increased number of parameters. Though the complexity of tunable AFs is relatively increased w.r.t. non-tunable AFs, it is

7. Miscellaneous Activation Functions

This section covers other attempts in AFs such as Softplus, Probabilistic, Polynomial, Subnetwork and Kernel.

7.1. Softplus Activation Functions

The softplus function [91] was proposed in 2001 as $\log(e^x+1)$ and mostly used in statistical applications. After the breakthrough of deep learning the softmax function is used as the AF [92]. Softmax function produces the categorical probability distribution equivalent output. Softplus unit based AF is also used in deep neural networks [93]. The smooth nature of the Softplus facilitates the differentiability. The noisy softplus AF [94] is suitable for the spiking neural networks (SNNs). A Softplus Linear Unit (SLU) is also proposed by considering softplus with rectified unit [95]. The SLU AF is defined as,

$$SLU(x) = \begin{cases} \alpha \times x, & x \geq 0 \\ \beta \times \log(e^x + 1) - \gamma, & x < 0 \end{cases} \quad (54)$$

where α, β and γ are the trainable parameters with α controlling the slope in the positive direction, β controlling the saturation points in the negative direction and γ controlling the offset in the negative direction w.r.t. the horizontal axis. The Rectified Softplus (ReSP) AF introduces the rectification for positive input in Softplus activation [96]. In order to make the softplus function to follow the zero mean, a shifting and scaling of the outputs is performed in [97]. A Rand Softplus (RSP) AF models the stochasticity-adaptability of biological neurons as,

$$RSP(x) = (1 - \rho) \times \max(0, x) + \rho \times \log(1 + e^x) \quad (55)$$

where ρ is a stochastic hyperparameter [98]. It improves the capability of the network towards the noise. The softplus function is also used with Tanh function in Mish activation function [99], which is given as,

$$Mish(x) = x \times Tanh(Softplus(x)). \quad (56)$$

The Mish is a non-monotonic and smooth AF. It has recently been used by the YOLOv4 model for object detection [100]. However, the increased complexity in Mish due to the multiple functions can be a limitation for the deep networks.

7.2. Probabilistic Activation Functions

So far, stochastic AFs have not been much explored due to expensive sampling processes. Few AFs exist in this category such as Randomized ReLU (RReLU) [50], Elastic ReLU (EReLU) [40], Randomly Translational ReLU (RTReLU) [41] and Gaussian Error Linear Unit (GELU) [101]. GELU [101] considers nonlinearity as the stochastic regularization driven transformation and defined as,

$$GELU(x) = x \times P(X \leq x). \quad (57)$$

where P is the probability. The complexity of GELU increases due to use of probabilistic nature. The GELU is also extended to the Symmetrical Gaussian Error Linear Unit (SGELU) [102] to enhance its ability of bidirectional convergence. Doubly truncated Gaussian distributions [103] is a family of nonlinearities which can generate different AFs such as Sigmoid, Tanh and ReLU by setting the appropriate truncation points. Probabilistic AF (ProbAct) introduces the adaptable and trainable variance in the ReLU's output [104]. It leads to the generalization of the models. However, all other drawbacks of ReLU exist with ProbAct also.

7.3. Polynomial Activation Functions

Smooth Adaptive AF (SAAF) is defined as the piecewise polynomial function [105]. Two power functions symmetric to the linear part of ReLU are combined in [106] to improve the performance of ReLU. A piecewise polynomial approximation based AF is also learnt from the data [107]. This activation leads to the light-weight models suitable for the FPGAs and microcontrollers. The AF is also treated as the cumulative distribution function [108]. The ReLU is also extended to a Rectified Power Unit (RePU) for positive inputs as,

$$RePU(x) = \begin{cases} x^s, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (58)$$

where s is a hyperparameter [109]. The RePU is suitable for smoother gradients near zero. However, vanishing gradient, unbounded and asymmetric nature are the downsides of RePU. The rational function of polynomials is better suited as compared to the polynomial functions in order to approximate the ReLU [110]. Recently, a Padé approximation is used to develop a non-smooth Padé Activation Unit (PAU) [111] as,

$$PAU(x) = \frac{P(x)}{Q(x)} \quad (59)$$

where $P(x)$ and $Q(x)$ are two polynomials of order m and n, respectively. The PAUs can approximate the commonly used hand-designed AFs. Moreover, it can also learn the new AFs with compact representations. Recently, a Rational AF (RAF) [112] was proposed to tackle the problem of non-smooth nature of the PAU function.

7.4. Activations as a Subnetwork

A Variable AF (VAF) is used as a subnetwork of ReLUs [113]. It uses the ensemble of ReLUs in a subnetwork using learnable parameters. In a very similar approach, the maximum of multiple linear functions is used in the Dynamic ReLU (DYReLU) [114]. In Wide Hidden Expansion (WHE) [115], each WHE intermediate channel is followed by one AF before connecting to the output channel to increase the non-linearity of the network. An AF Unit (AFU) [116] uses a small neural network to model the activation. All neurons in the original network share the weights in AFU. The advantage of the AFU is that different AFs can be learnt by different layers.

7.5. Kernel Activation Functions

A Kernel-based non-parametric AF (KAF) [117] uses an inexpensive kernel expansion to make the activation flexible. The KAF is further extended to multikernel AFs (multi-KAF) [118]. Several AFs are also introduced for complex valued neural networks [119], [120], [121].

8. Aspects of Activation Functions

This section summarizes the effect of weight initialization, understanding of AFs and suitability with different types of data. The learning of the network speeds up drastically by using the orthogonal weight initialization based on the dynamical isometry [122]. A set of conditions in parameter initialization also boosts the performance of networks with sigmoidal activations [123]. The symmetric probability distribution based weights and biases initialization leads the network to suffer with the dying ReLU problem. However, the asymmetric initialization resolves the dying ReLU problem [124]. The over-parameterization during initialization also benefits in the training [125]. The data-dependent weight initialization using a subset of data minimizes the issues of the ReLU [126], whereas an initial parameter sharing based initialization guarantees the dynamical isometry for the ReLU [127].

Several researchers have tried to understand the working and impact of AFs through different strategies. The lower and upper bounds are established for network complexity to realize that the ReLU in deep networks approximates the smooth functions more efficiently as compared to shallow networks [128]. A ReLU network with only one hidden layer is trained to reach the global optimum in polynomial time even with exponentially growing input dimension [129]. The ReLU type AF based neural networks produce the overconfident predictions far away from the training data [130]. However, this can be resolved by employing adversarial confidence enhanced training. A Gaussian margin driven time and accuracy tradeoff analysis is also done on the ReLU's learning [131]. The singular values for ReLU layers are analyzed to understand the interaction of ReLU with the linear components [132]. The approximation of Gaussian posterior distribution over the ReLU network weight's fixes the overconfidence problem [133].

Despite most of the AFs are tested over image data, there are few research papers dealing with the AFs over other types of data. Table 6 summarizes the insights and remarks of state-of-the-art AFs for various networks and datasets.

9. Performance Comparison and Analysis

This survey is compared with the existing survey/performance analysis and the experimental performance analysis of selected AFs is performed over Image, Text and Speech data.

9.1. Comparison with Existing Survey/Performance Analysis

A performance analysis of AFs was conducted using multilayer perceptron network in [134]. Among compared AFs, the Tanh has shown better performance. A comparative performance analysis of different AFs suggests an Elliott function as better suited for classification using LSTM networks [25]. The ELU outperforms the ReLU, LReLU, and SELU AFs over MNIST classification task using Deep Neural Networks [135]. As per [136], the ELU is reported in [135] to outperform the ReLU, LReLU, PReLU and PELU over sufficiently large datasets for speech recognition. However, for smaller datasets, the ReLU is preferred. A similar trend is also reported in [137] with a note that the ELU and SELU AFs exhibit faster convergence as compared to the ReLU and LReLU AFs. In [138], 21 AFs are listed without experimental results comparison. In contrast to [138], this paper presents a comprehensive survey of AFs. The ReLU based deep networks perform superior or mildly worse than the spline methods [139]. A review of adaptive functions is conducted in [140] by considering 9 functions, including Sigmoid, Tanh, PReLU, and adapt-Tanh. In [141], the comparison between ReLU and LReLU is performed using CNN on MNIST dataset. An empirical study is also done for the variations of ReLU activation by generalizing it with the help of parameters [142]. The comparison of AFs is also performed for generalized learning vector quantization [143]. The ReLU activation has performed better for object, face, and text datasets [144]. However, the SELU and Maxout have performed better for medical and sound datasets, respectively [144]. The piecewise AF is better suited for facial expression recognition in [145]. A survey of adaptive AFs is conducted in [146] without experimental comparison. The evaluation of seven AFs is conducted in [147] using a simple network over CIFAR10 dataset, whereas in our survey we cover different AFs and also perform the experimental comparison.

A summary of the comparison with existing surveys and performance analysis of AF is shown in Table 7. Following are the observations:

- This survey presents a detailed classification to cover the wide range of AFs as compared to the existing surveys and performance analysis.

- This survey covers exhaustive state-of-the-art AFs to date, whereas the existing survey/performance analysis covers either a limited number of AFs or only basic AFs.

- The performance analysis conducted in this paper considers a wide range of neural networks over different types of data for eighteen AFs, whereas the existing analysis is limited to a single type of data and network.

- This survey highlights the trends to help the researchers to further explore the better AFs and practitioners to choose based on the data and network types.

9.2. Experimental Performance Analysis

In order to compare the AFs, three experiments are conducted in this paper, including image classification, language

Table 6: Summary of the existing state-of-the-art activation functions.

Activation	Models	Datasets	Insights and Remarks
On Image Datasets			
Wide Hidden Expansion (WHE) - 2020 [115]	ResNet, SENet, and MobileNet	CIFAR100 and ImageNet classification, Pascal VOC 2007 and COCO detection	Upto 2% higher Top-1 accuracy than baseline models of recognition and detection
Soft-Root-Sign (SRS) - 2020 [26]	VGG and MobileNet	CIFAR10 and CIFAR100 classification	The SRS is better with MobileNet over both datasets and with VGG over CIFAR100. The LReLU is better with VGG over CIFAR10.
Relu-Memristor-Like AF (RMAF) - 2020 [72]	ResNet, AlexNet, SqueezeNet, and DenseNet	CIFAR10, CIFAR100, MNIST and ImageNet classification	The RMAF performs better than the ReLU, ELU, SELU, PReLU, Tanh and Swish.
Parametric Deformable ELU (PDELU) - 2020 [59]	NIN and ResNet	CIFAR10 and CIFAR100 classification	The PDELU performs better than the ReLU, ELU and FReLU.
Pade Activation Unit (PAU) - 2020 [111]	VGG8, MobileNetV2, ResNet and DenseNet	MNIST, Fashion-MNIST, CIFAR10 and ImageNet classification	The PAU encode AFs as rational functions and performs better than many existing AFs.
Elastic Exponential Linear Unit (EELU) - 2020 [58]	A simple CNN model and VGG16	CIFAR10, CIFAR100, ImageNet, and Tiny ImageNet classification	The EELU shows better results than the ReLU, ELU, EPReLU and Swish.
Dynamic ReLU (DY-ReLU) - 2020 [114]	MobileNetV2	ImageNet classification and COCO detection	The DY-ReLU is suitable for light-weight networks.
Variable AF (VAF) - 2019 [113]	Shallow CNN models	MNIST, Fashion MNIST and CIFAR10 classification	The VAF shows promising performance.
Multi-bin Trainable Linear Unit (MTLU) - 2019 [46]	FDnet and FSRnet	Image denoising and Super-resolution	The MTLU is significantly faster having comparable results with the state-of-the-arts.
Swish - 2018 [29]	MobileNet, ResNet, WRN and DenseNet	CIFAR10, CIFAR100 and ImageNet classification	The learnable parameter in Swish leads to improved performance than Softplus.
On Time Series Datasets			
Variable AF (VAF) - 2019 [113]	Multi-Layered Neural Network	Regression tasks (Kinematics, Energy Cooling, Yatch, etc.)	Better performance over Kinematics, Energy Cooling and Yatch datasets.
Self-Learnable AFs (SLAF) - 2019 [79]	Multi-Layered Neural Network	Boston Housing and Learning Sparse Polynomial regression	The newer parameter space makes the optimization easier.
On Text Datasets			
Soft-Root-Sign (SRS) - 2020 [26]	A 6 layer transformer network	IWSLT 2016 German-English translation	The SRS is better over tst2011 and tst2012 test sets, whereas the SELU and LReLU are better over tst2013 and tst2014 test sets, respectively.
Swish - 2018 [29]	A 12 layer transformer network	WMT 2014 English-German dataset	The performance of Swish is comparable to state-of-the-arts.
PenalizedTanh - 2018 [33]	MLP, CNN and RNN	Sentence classification, Document classification and Sentence tagging	The PenalizedTanh exhibits the stability across the different tasks in contrast to the Swish function.
On Signal Datasets			
Rectified Linear Tanh (ReLTanh) - 2019 [67]	Stacked autoencoder (SAE) based DNN	Vibration signals for rotating machinery fault diagnosis	The ReLTanh leads to larger gradients for faster learning and reduces the vanishing gradient.
On Game Datasets			
Sigmoid-weighted Linear Unit (SiLU) - 2018 [23]	Deep reinforcement learning algorithm	SZ-Tetris, 10×10 Tetris, and Atari 2600 games	The SiLU AF outperforms the ReLU function for reinforcement learning.

translation and speech recognition. Eighteen state-of-the-art AFs are considered for analysis, including Logistic Sigmoid, Tanh, Elliott [25], ReLU [8], LReLU [34] PReLU [35], ELU [27], SELU [52], GELU [101], CELU [53], Softplus [93], Swish [29], ABReLU [44], LiSHT [24], Soft-Root-Sign (SRS) [26], Mish [99], PAU [111] and PDELU [59]. Note that Swish, ABReLU, LiSHT, SRS, Mish, PAU and PDELU are the most recent functions. Google Colab based computational resource is used in most of the experiments. Few experiments are also performed over a desktop system consisting of 8 GB GPU. The PyTorch framework is used in all the experiments.

The CIFAR10 and CIFAR100 datasets[1] [148] are used for the image classification experiment in this paper. The CIFAR10 dataset contains $50,000$ training images and $10,000$ test images from 10 object categories. The CIFAR100 dataset contains $50,000$ training images and $10,000$ test images from 100 object categories. We also utilize the language translation and speech recognition datasets for the experiments. For the experiments over CIFAR-10 and CIFAR-100 datasets, training is performed for 100 Epochs. The batch size is 128 for CIFAR-10 and 64 for CIFAR-100. The learning rate is 0.001 for first 80 Epochs and 0.0001 for last 20 Epochs. Random crop and random horizontal flip are the data augmentation used during training. Data normalization is performed both during train and test times. Adam optimizer is used for the training with cross entropy loss. All existing activation functions except softmax are replaced with the corresponding activation function in different networks.

The test accuracy is reported in Tables 8 and 9 on CIFAR10 and CIFAR100 datasets, respectively. In these Tables, the mean and standard deviation of image classification accuracy over

---

[1]https://www.cs.toronto.edu/~kriz/cifar.html

Table 7: Comparison of this survey with the existing surveys and performance evaluations.

Method	Models	Activations	Datasets	Remarks
Karlik and Ol-gac [134]	Multilayer Perceptron (MLP)	5 AFs, including Bi-polar sigmoid, Uni-polar sigmoid, Tanh, etc.	Classification	The Tanh performs better compared to other traditional AFs.
Vydana and Vuppala (2017) [136]	Hidden Markov Model-Deep Neural Network (HMM-DNN)	5 AFs, including ReLU, LReLU, PReLU, ELU, and PELU	TIMIT and WSJ speech recognition	The ELU is better over sufficiently larger size datasets. However, the ReLU is preferred for smaller datasets.
Alcantara (2017) [135]	A neural network with 2 hidden layers having 100 neurons/layer	4 AFs, including ReLU, LReLU, ELU, and SELU	MNIST classification	The ELU AF outperforms others.
Pedamonti (2018) [137]	A neural network with 2 hidden layers having 100 neurons/layer	5 AFs, including Sigmoid, ReLU, LReLU, ELU, and SELU	MNIST classification	The ELU and SELU AFs exhibit the faster convergence as compared to the ReLU and LReLU AFs.
Lau and Lim (2018) [140]	Deep Neural Network (DNN)	ReLU and Adaptive ReLU	MNIST classification	The adaptive AFs improve the generalization of the network.
Farzad et al. (2019) [25]	Long Short Term Memory (LSTM)	23 AFs, including Elliott, Gaussian, Logarithmic, Loglog, etc.	IMDB, Movie Review, MNIST classification	Elliott function is better suited to the LSTM network.
Dubey and Jain (2019) [141]	Simple Convolutional Neural Network (CNN)	2 AFs, including ReLU and Leaky ReLU	MNIST classification	The ReLU performed better than Leaky ReLU (LReLU).
Banerjee et al. (2019) [142]	Convolutional Neural Network (CNN)	Generalized ReLU	MNIST classification	Network learns the parameters for different ReLU variations.
Villmann et al. (2019) [143]	Generalized learning vector quantization (GLVQ)	12 AFs, including Sigmoid, Swish, ReLU, Softplus, etc.	Tecator, Indian Pine and Wisconsin-Breast-Cancer classification	The Sigmoid, Swish and Softplus AFs are better suited with GLVQ.
Castaneda et al. (2019) [144]	6 different models for different applications	3 AFs, including ReLU, SELU and Maxout	Object, Face, Text, Medical and Sound datasets	The ReLU is better for object, face and text datasets, whereas SELU and Maxout are better for medical and sound datasets, respectively.
Wang et al. (2020) [145]	Inception-v3 model	6 AFs, including Sigmoid, Tanh, ReLu, etc.	JAFFE and FER2013 facial expression recognition	The combination of log, softdesign and ReLU AFs provides improved performance.
Szandala (2020) [147]	A simple network	7 AFs, including Sigmoid, Tanh, ReLU, LReLU, Swish, etc.	CIFAR10 classification	The LReLU performs better. The ReLU is efficient.
Our survey and performance analysis	MobileNet, VGG, GoogLeNet, ResNet, SENet, DenseNet, etc.	Exhaustive list of AFs, including performance analysis over 18 state-of-the-art activations	CIFAR10 classification, Language translation, Speech recognition	A classification to categorize and analyze the AFs and a performance comparison of the state-of-the-art activations.

Table 8: Experimental results comparison over CIFAR10 dataset.

Accuracy	CNN Models					
Activations	MobileNet	VGG16	GoogLeNet	ResNet50	SENet18	DenseNet121
Sigmoid	88.60 ± 0.17	87.69 ± 0.49	87.33 ± 2.48	80.13 ± 3.33	90.29 ± 0.29	89.92 ± 1.96
Tanh	87.21 ± 0.24	90.49 ± 0.11	90.16 ± 1.86	89.09 ± 1.47	90.44 ± 0.09	91.80 ± 0.69
Elliott [25]	88.48 ± 0.18	87.94 ± 0.49	89.84 ± 3.43	81.60 ± 3.91	90.25 ± 0.25	91.53 ± 1.04
ReLU [8]	90.10 ± 0.22	*92.84* ± 0.19	*93.43* ± 0.48	93.74 ± 0.34	93.70 ± 0.16	**93.96** ± 0.51
LReLU [17]	90.10 ± 0.19	91.09 ± 0.09	89.28 ± 0.82	*93.83* ± 0.42	93.66 ± 0.19	93.85 ± 0.48
PReLU [35]	90.43 ± 0.18	92.19 ± 0.08	92.85 ± 0.55	92.99 ± 0.62	92.76 ± 0.26	92.82 ± 0.63
ELU [27]	90.92 ± 0.25	88.55 ± 1.17	92.47 ± 0.76	93.53 ± 0.66	93.39 ± 0.20	92.89 ± 0.62
SELU [52]	90.11 ± 0.32	92.25 ± 0.28	91.87 ± 0.84	93.53 ± 0.52	89.96 ± 0.31	92.71 ± 0.73
GELU [101]	90.71 ± 0.20	92.42 ± 0.09	93.16 ± 0.61	93.81 ± 0.46	**93.72** ± 0.18	*93.90* ± 0.41
CELU [53]	*91.04* ± 0.17	88.11 ± 0.14	92.60 ± 0.60	**94.09** ± 0.17	91.63 ± 0.22	93.46 ± 0.35
Softplus [93]	**91.05** ± 0.22	92.69 ± 0.20	92.66 ± 0.66	93.34 ± 0.65	93.25 ± 0.11	93.07 ± 0.70
Swish [29]	90.66 ± 0.34	92.32 ± 0.20	92.68 ± 0.53	93.02 ± 0.85	93.24 ± 0.19	93.16 ± 0.51
ABReLU [44]	88.97 ± 0.47	92.36 ± 0.15	93.34 ± 0.23	93.29 ± 0.52	93.35 ± 0.14	93.26 ± 0.55
LiSHT [24]	86.53 ± 0.49	89.83 ± 0.28	90.27 ± 0.80	90.89 ± 0.66	90.25 ± 0.84	87.91 ± 0.93
SRS [26]	89.43 ± 0.81	92.06 ± 0.26	91.36 ± 1.19	92.28 ± 0.48	78.05 ± 1.37	90.64 ± 1.93
Mish [99]	90.82 ± 0.15	**92.85** ± 0.25	93.29 ± 0.61	93.69 ± 0.63	93.66 ± 0.12	93.62 ± 0.62
PAU [111]	90.67 ± 0.17	92.00 ± 0.26	92.80 ± 0.65	93.67 ± 0.52	93.08 ± 0.20	93.05 ± 0.53
PDELU [59]	90.18 ± 0.19	92.80 ± 0.13	**93.49** ± 0.30	93.42 ± 0.71	*93.71* ± 0.07	**93.96** ± 0.59

5 trials are reported for each AF. Moreover, the better results are highlighted. Different types of CNN models are used in this experiment, such as plain models (i.e., MobileNet [149] and VGG16 [150]), inception model (i.e., GoogLeNet [151]) and skip/residual connection based models (i.e., ResNet50 [152], SENet18 [153], and DenseNet121 [154]). The MobileNet, GoogLeNet and SENet18 are light models, whereas the VGG16, ResNet50 and DenseNet121 are heavy models in

Table 9: Experimental results comparison over CIFAR100 dataset.

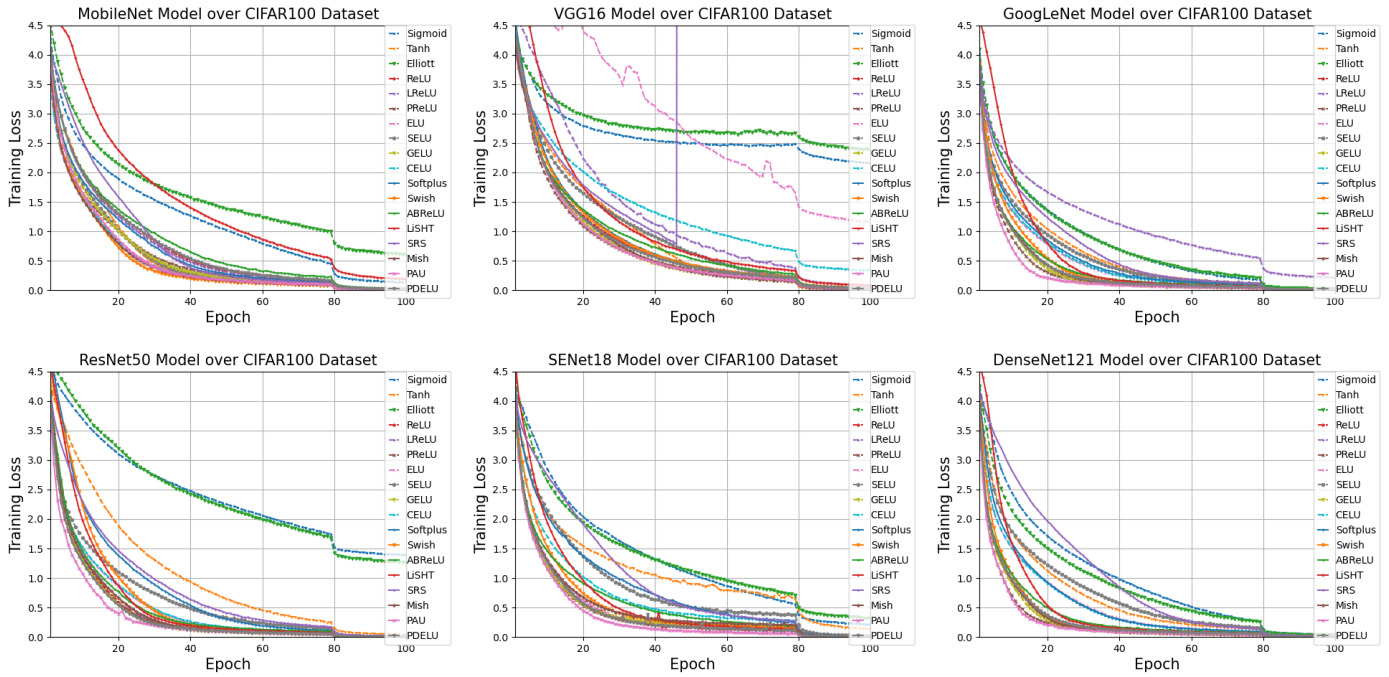| Accuracy | CNN Models | | | | | |
Activations	MobileNet	VGG16	GoogleNet	ResNet50	SENet18	DenseNet121
Sigmoid	61.88 ± 0.18	37.75 ± 0.59	70.31 ± 0.54	46.78 ± 5.42	66.17 ± 1.16	68.31 ± 2.41
Tanh	53.10 ± 0.51	58.43 ± 0.38	67.66 ± 2.32	64.32 ± 1.69	60.13 ± 1.86	69.53 ± 1.68
Elliott [25]	60.70 ± 0.34	33.20 ± 0.97	64.85 ± 6.28	49.88 ± 4.03	66.30 ± 0.28	69.58 ± 2.40
ReLU [8]	61.33 ± 0.34	67.47 ± 0.44	74.05 ± 1.69	71.96 ± 0.94	70.45 ± 0.73	72.99 ± 1.35
LReLU [17]	61.13 ± 0.41	65.72 ± 0.14	63.79 ± 2.38	**72.77** ± 0.49	70.58 ± 0.45	73.33 ± 1.25
PReLU [35]	59.86 ± 0.35	65.26 ± 0.40	69.57 ± 1.50	71.08 ± 1.70	69.77 ± 0.48	68.23 ± 1.55
ELU [27]	*61.97* ± 0.24	51.35 ± 3.01	72.57 ± 1.76	71.41 ± 1.63	**71.27** ± 0.58	72.06 ± 1.93
SELU [52]	59.62 ± 0.39	64.55 ± 0.43	71.47 ± 1.39	69.94 ± 1.92	55.01 ± 0.98	70.15 ± 1.04
GELU [101]	61.20 ± 0.61	67.25 ± 0.38	*74.27* ± 0.70	71.58 ± 0.87	71.14 ± 0.29	73.31 ± 1.70
CELU [53]	61.90 ± 0.21	55.78 ± 0.69	72.87 ± 1.52	70.95 ± 1.40	63.43 ± 0.81	72.68 ± 1.16
Softplus [93]	**62.59** ± 0.21	67.70 ± 0.19	73.08 ± 1.66	71.99 ± 2.03	*71.16* ± 0.46	72.54 ± 1.73
Swish [29]	59.40 ± 0.41	66.05 ± 0.82	71.56 ± 1.66	71.12 ± 2.08	68.42 ± 1.62	71.34 ± 1.10
ABReLU [44]	56.21 ± 0.53	66.95 ± 0.09	71.83 ± 2.26	71.96 ± 1.43	70.47 ± 0.91	**73.79** ± 1.45
LiSHT [24]	54.09 ± 1.54	58.87 ± 0.81	66.66 ± 2.50	65.28 ± 1.33	66.01 ± 1.04	65.61 ± 1.10
SRS [26]	54.93 ± 0.80	58.22 ± 1.09	70.39 ± 1.09	67.11 ± 1.46	36.95 ± 0.93	64.52 ± 1.39
Mish [99]	61.81 ± 0.54	**68.13** ± 0.40	73.76 ± 1.48	71.89 ± 1.12	70.80 ± 0.68	73.49 ± 1.39
PAU [111]	59.81 ± 0.61	64.14 ± 0.62	70.48 ± 1.53	68.59 ± 2.15	68.29 ± 0.77	67.83 ± 0.35
PDELU [59]	61.35 ± 0.56	*67.92* ± 0.32	**74.48** ± 1.23	*72.11* ± 1.60	70.81 ± 0.47	*73.71* ± 1.64

Figure 3: Convergence plots over CIFAR100 dataset.

terms of the number of trainable parameters. Overall, it is observed that the Softplus, ELU and CELU are better suited with MobileNet. The ReLU, Mish and PDELU exhibit good performance with VGG16, GoogleNet and DenseNet. The ReLU, LReLU, ELU, GELU, CELU, ABReLU, and PDELU activation functions are better for the networks having residual connections, such as ResNet50, SENet18 and DenseNet121. In order to demonstrate the convergence of different AFs, the training loss vs epochs is plotted in Fig. 3 on CIFAR100 dataset using different models. The PAU has emerged as a promising AF with fastest convergence in most of the cases. The PReLU, GELU and PDELU AFs are also consistent with good convergence. Note that the training diverges with SRS for the

SENet18 model. Sigmoid and Elliott AFs showed the poorest convergence. The time taken for the training is also computed for different AFs using different CNN models on CIFAR100 dataset and reported in Table 10. These results are computed using a desktop computer system having 32 GB RAM and 8 GB Nvidia GPU Card for 100 epochs of training. The time is represented in hh:mm:ss format. It is clear that PDELU AF is very inefficient. Moreover, SRS and Elliott also take more time for training. The activations such as ReLU, ELU, CELU, and Softplus depict a good tradeoff between the accuracy and training time.

The results for language translation and speech recognition for different AFs are illustrated in Table 11. The German to

Table 10: Training time (hh:mm:ss) comparison over CIFAR100 dataset.

Training Time	CNN Models					
Activations	MobileNet	VGG16	GoogleNet	ResNet50	SENet18	DenseNet121
Sigmoid	00:33:15	00:49:16	04:55:54	03:36:03	01:13:14	04:12:24
Tanh	00:33:18	00:49:55	04:58:02	03:33:03	01:13:18	04:09:24
Elliott [25]	00:49:52	00:59:13	06:53:55	05:38:49	01:41:38	07:46:55
ReLU [8]	00:31:22	00:47:19	04:55:10	03:32:30	01:15:33	04:15:06
LReLU [34]	00:31:48	00:49:03	05:01:30	03:33:00	01:18:38	04:14:09
PReLU [35]	00:44:24	00:49:01	05:42:18	03:55:57	01:27:05	04:55:47
ELU [27]	00:31:05	00:47:38	04:57:37	03:36:47	01:13:25	04:08:39
SELU [52]	00:29:40	00:47:31	04:54:57	03:33:47	01:13:27	04:09:17
GELU [101]	00:29:43	00:47:22	04:55:53	03:32:32	01:13:32	04:11:26
CELU [53]	00:29:36	00:46:47	05:00:44	03:31:40	01:14:08	04:18:11
Softplus [93]	00:29:44	00:47:06	04:58:55	03:32:03	01:14:02	04:12:08
Swish [29]	00:43:13	00:55:37	06:18:38	04:58:38	01:32:15	06:41:14
ABReLU [44]	00:38:51	00:53:49	05:43:59	04:27:02	01:25:30	05:42:53
LiSHT [24]	00:37:01	00:54:10	05:40:00	04:25:57	01:23:59	05:38:15
SRS [26]	01:06:38	01:11:36	08:43:09	07:35:35	02:05:33	11:10:27
Mish [99]	00:40:19	00:54:23	05:59:48	04:46:45	01:28:53	06:10:27
PAU [111]	00:41:59	00:54:10	05:54:22	04:12:31	01:25:37	05:39:57
PDELU [59]	05:23:38	04:01:55	34:22:00	36:48:48	08:32:40	50:23:00

Table 11: Experimental results for German to English language translation and speech recognition tasks.

	Language Translation	Speech Recognition	
Activations	Bleu Score	Average CER	Average WER
Sigmoid	14.59 ± 0.47	0.53 ± 0.18	1.19 ± 0.39
Tanh	**20.93** ± 0.91	0.26 ± 0	0.68 ± 0
Elliott [25]	14.49 ± 0.96	0.40 ± 0.01	0.93 ± 0.01
ReLU [8]	18.88 ± 0.86	**0.24** ± 0.01	*0.66* ± 0.01
LReLU [34]	18.89 ± 0.82	**0.24** ± 0	*0.66* ± 0.01
PReLU [35]	20.04 ± 0.98	**0.24** ± 0	**0.65** ± 0
ELU [27]	19.40 ± 1.33	*0.25* ± 0	0.67 ± 0
SELU [52]	*20.85* ± 0.64	0.26 ± 0	0.69 ± 0.01
GELU [101]	18.75 ± 1.83	**0.24** ± 0	**0.65** ± 0
CELU [53]	18.71 ± 0.55	*0.25* ± 0	0.67 ± 0
Softplus [93]	16.78 ± 0.84	0.30 ± 0.01	0.76 ± 0.02
Swish [29]	19.51 ± 0.97	**0.24** ± 0.01	**0.65** ± 0.01
ABReLU [44]	17.55 ± 0.93	*0.25* ± 0	0.68 ± 0
LiSHT [24]	20.39 ± 0.93	0.29 ± 0.01	0.74 ± 0.01
SRS [26]	20.66 ± 0.78	0.28 ± 0	0.72 ± 0
Mish [99]	19.56 ± 1.15	**0.24** ± 0	**0.65** ± 0
PAU [111]	20.11 ± 1.24	**0.24** ± 0	**0.65** ± 0.01
PDELU [59]	19.07 ± 0.95	*0.25* ± 0	0.67 ± 0.01

English translation is used to test the performance of the AFs over text data. Benchmark Seq2Seq model consisting of a Long Short Term Memory (LSTM) based autoencoder network is used for the experiment. The model and dataset are downloaded from Kaggle[2]. The AF is applied to the feature embedding before the dropout layer. For the language translation experiments, the number of Epochs is set to 50 with 0.001 learning rate and 256 batch size. The embedding size of encoder and decoder is 300. The dropout factor is 0.5 for both encoder and decoder. Adam optimizer is used for the training with cross entropy loss. The Bleu score [155] with 4-*gram* is reported in Table 11 in 2nd column for different AFs. The mean and standard deviation of Bleu score over 5 trials are reported for each AF. It is noticed that the Tanh and SELU AFs are better suitable for language translation. The PReLU, LiSHT, SRS and PAU AFs also perform better for language translation.

The speech recognition experiment is also performed to show the performance of the different AFs for time-series signal data. The end-to-end speech recognition based Deep Speech 2 framework available from assemblyai[3] is used. The model consists of 2 layers of residual convolution layers to learn the relevant audio features, and 2 layers of bidirectional gated recurrent units (GRUs) to use the learned residual convolutional audio features. The 100 hours of transcribed audio English data from LibriSpeech dataset is used for the experiment. For the speech recognition experiments, torchaudio 0.4.0 and torch 1.4.0 are used. The model consists of 2 CNN layers and 2 RNN layers. The dimension of a RNN layer is 512. Number of classes is 29 in the dataset. Dropout factor is 0.5. The learning rate is 0.0005, batch size is 10 and the number of Epochs is 10. The mean and standard deviation over 5 trials of character error rate (CER) and word error rate (WER) are reported in Table 11 for speech recognition. The recent AFs such as PReLU, GELU, Swish, Mish and PAU AFs are found as the most suitable for speech recognition in this experiment.

10. Conclusion and Recommendations

An extensive and up to date survey of activation functions is conducted in this paper. Different types of AFs are considered, including Logistic Sigmoid and Tanh based, ReLU based, ELU based, and Learning based. However, the main focus is given to the recent developments in AFs in view of the deep learning applications of neural networks. The overview of AFs presented in this paper focuses on the aspects including the detailed coverage of AFs, classification and performance comparison over image, text and speech data.

Following are the concluding remarks of the survey and performance analysis conducted through this paper:

- Most of the improvements in Logistic Sigmoid and Tanh targets to tackle the non zero-mean and zero-gradient problems. However, these improvements carry forward the drawback of increased complexity.

- The ReLU variants try to tackle the three major problems of ReLU, namely under-utilization of negative values, limited nonlinearity and unbounded output. These activations perform well for some applications, e.g. LReLU and ABReLU works better with residual networks. However, most of these activations fail to perform better than ReLU, e.g. LReLU, PReLU and ABReLU do not improve for MobileNet, VGG and GoogleNet models. Note that, the ReLU, Leaky ReLU and PReLU AFs are the most common choice among researchers due to its simplicity. Moreover, many networks consider the ReLU as a default choice for the AF.

- The exponential based AFs also focus over the better utilization of the negative values and to avoid the saturation for important features. However, most of the exponential activations suffer due to the non-smooth functions.

- The learning based adaptive AFs try to find the best parameters to represent the non-linearity needed for the given dataset. This category of AF has gained more popularity in recent years. However, the major problem associated with such AF is to find the better base function and number of trainable parameters. Some AFs diverge during the training if not initialized properly.

- In contrast to existing surveys, this survey covers an exhaustive list different types of AFs. Moreover, a performance analysis on different types of data using several AFs provides new insights for future research.

Following are the recommendations curated from this survey and performance analysis:

- In order to speed up the training, both negative & positive values should be used to ensure the near zero mean.

- The most important aspect in deep learning is to find the network having matching complexity as the dataset complexity. If the complexity of the model is high then it may lead to overfitting and if the complexity of the model is low then it may lead to under convergence. Thus, the AF should bridge this gap based on the model and dataset complexity during training automatically.

- The Logistic Sigmoid and Tanh AFs should be avoided for Convolutional Neural Networks as it leads to poor convergence. However, this type of AF is commonly used as gates in recurrent neural networks.

- Despite the ReLU being a popular choice, recently proposed AFs such as Swish, Mish, and PAU are also worth trying for different problems.

- The ReLU, Mish and PDELU activation functions have shown a good performance with VGG16 and GoogleNet. The ReLU, LReLU, ELU, GELU, CELU, and PDELU functions are better for the networks having residual connections for image classification.

- In general, the parametric AFs show better convergence as it can adapt the data faster by learning the parameter from the data. Specially, PAU, PReLU and PDELU have shown better convergence.

- Some AFs lead to increased training time complexity. PDELU and SRS are such examples. However, AFs such as ReLU, SELU, GELU, and Softplus depict a promising tradeoff between the accuracy and training time.

- The exponential AFs generally lead to the increased nonlinearity due to utilization of the negative values.

- The Tanh and SELU AFs are found better for language translation along with PReLU, LiSHT, SRS and PAU.

- It is suggested to use the PReLU, GELU, Swish, Mish and PAU AFs for speech recognition.

References

[1] F. Shao, L. Chen, J. Shao, W. Ji, S. Xiao, L. Ye, Y. Zhuang, J. Xiao, Deep learning for weakly-supervised object detection and localization: A survey, Neurocomputing (2022).

[2] Y. Mo, Y. Wu, X. Yang, F. Liu, Y. Liao, Review the state-of-the-art technologies of semantic segmentation based on deep learning, Neurocomputing (2022).

[3] Y. Guo, F. Feng, X. Hao, X. Chen, Jac-net: Joint learning with adaptive exploration and concise attention for unsupervised domain adaptive person re-identification, Neurocomputing (2022).

[4] S. R. Dubey, A decade survey of content based image retrieval using deep learning, IEEE Transactions on Circuits and Systems for Video Technology (2021).

[5] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, N. Ding, Gan-based anomaly detection: A review, Neurocomputing (2022).

[6] H. Li, Y. Pan, J. Zhao, L. Zhang, Skin disease diagnosis with deep learning: a review, Neurocomputing 464 (2021) 364–393.

[7] C. H. Dagli, Artificial neural networks for intelligent manufacturing, Springer Science & Business Media, 2012.

[8] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.

[9] A. Graves, A.-r. Mohamed, G. Hinton, Speech recognition with deep recurrent neural networks, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 6645–6649.

[10] K. K. Babu, S. R. Dubey, Pcsgan: Perceptual cyclic-synthesized generative adversarial networks for thermal and nir to visible image transformation, Neurocomputing (2020).

[11] J. Liu, Y. Liu, Q. Zhang, A weight initialization method based on neural network with asymmetric activation function, Neurocomputing (2022).

[12] Y. Srivastava, V. Murali, S. R. Dubey, A performance evaluation of loss functions for deep face recognition, in: National Conference on Computer Vision, Pattern Recognition, Image Processing, and Graphics, Springer, 2019, pp. 322–332.

[13] S. S. Basha, S. R. Dubey, V. Pulabaigari, S. Mukherjee, Impact of fully connected layers on performance of convolutional neural networks for image classification, Neurocomputing 378 (2020) 112–119.

[14] Q. Xu, M. Zhang, Z. Gu, G. Pan, Overfitting remedy by sparsifying regularization on fully-connected layers of cnns, Neurocomputing 328 (2019) 69–74.

[15] S. R. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, B. B. Chaudhuri, diffgrad: An optimization method for convolutional neural networks, IEEE transactions on neural networks and learning systems 31 (11) (2019) 4500–4511.

[16] W. Duch, N. Jankowski, Survey of neural transfer functions, Neural Computing Surveys 2 (1) (1999) 163–212.

[17] V. Nair, G. E. Hinton, Rectified linear units improve restricted boltzmann machines, in: International Conference on Machine Learning, 2010, pp. 807–814.

[18] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[19] A. N. S. Njikam, H. Zhao, A novel activation function for multilayer feed-forward neural networks, Applied Intelligence 45 (1) (2016) 75–82.

[20] B. Xu, R. Huang, M. Li, Revise saturated activation functions, International Conference on Learning Representations Workshop (2016).

[21] S. Kong, M. Takatsuka, Hexpo: A vanishing-proof activation function, in: International Joint Conference on Neural Networks, 2017, pp. 2562–2567.

[22] Y. Qin, X. Wang, J. Zou, The optimized deep belief networks with improved logistic sigmoid units and their application in fault diagnosis for planetary gearboxes of wind turbines, IEEE Transactions on Industrial Electronics 66 (5) (2018) 3814–3824.

[23] S. Elfwing, E. Uchibe, K. Doya, Sigmoid-weighted linear units for neural network function approximation in reinforcement learning, Neural Networks 107 (2018) 3–11.

[24] S. K. Roy, S. Manna, S. R. Dubey, B. B. Chaudhuri, Lisht: Non-parametric linearly scaled hyperbolic tangent activation function for neural networks, arXiv preprint arXiv:1901.05894 (2019).

[25] A. Farzad, H. Mashayekhi, H. Hassanpour, A comparative performance analysis of different activation functions in lstm networks for classification, Neural Computing and Applications 31 (7) (2019) 2507–2521.

[26] Y. Zhou, D. Li, S. Huo, S.-Y. Kung, Soft-root-sign activation function, arXiv preprint arXiv:2003.00547 (2020).

[27] D.-A. Clevert, T. Unterthiner, S. Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), in: International Conference on Learning Representations, 2016.

[28] F. Agostinelli, M. Hoffman, P. Sadowski, P. Baldi, Learning activation functions to improve deep neural networks, International Conference on Learning Representations Workshops (2015).

[29] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, International Conference on Learning Representations Workshops (2018).

[30] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

[31] P. Chandra, Y. Singh, An activation function adapting training algorithm for sigmoidal feedforward networks, Neurocomputing 61 (2004) 429–437.

[32] S. S. Sodhi, P. Chandra, Bi-modal derivative activation function for sigmoidal feedforward networks, Neurocomputing 143 (2014) 182–196.

[33] S. Eger, P. Youssef, I. Gurevych, Is it time to swish? comparing deep learning activation functions across nlp tasks, arXiv preprint arXiv:1901.02671 (2019).

[34] A. L. Maas, A. Y. Hannun, A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in: International Conference on Machine Learning, Vol. 30, 2013, p. 3.

[35] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: IEEE international conference on computer vision, 2015, pp. 1026–1034.

[36] W. Shang, K. Sohn, D. Almeida, H. Lee, Understanding and improving convolutional neural networks via concatenated rectified linear units, in: International Conference on Machine Learning, 2016, pp. 2217–2225.

[37] S. S. Liew, M. Khalil-Hani, R. Bakhteri, Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems, Neurocomputing 216 (2016) 718–734.

[38] R. Duggal, A. Gupta, P-telu: Parametric tan hyperbolic linear unit activation for deep neural networks, in: IEEE International Conference on Computer Vision Workshops, 2017, pp. 974–978.

[39] S. Qiu, X. Xu, B. Cai, Frelu: Flexible rectified linear units for improving convolutional neural networks, in: International Conference on Pattern Recognition, 2018, pp. 1223–1228.

[40] X. Jiang, Y. Pang, X. Li, J. Pan, Y. Xie, Deep neural networks with elastic rectified linear units for object recognition, Neurocomputing 275 (2018) 1132–1139.

[41] J. Cao, Y. Pang, X. Li, J. Liang, Randomly translational activation inspired by the input distributions of relu, Neurocomputing 275 (2018) 859–868.

[42] F. Godin, J. Degrave, J. Dambre, W. De Neve, Dual rectified linear units (drelus): A replacement for tanh activation functions in quasi-recurrent neural networks, Pattern Recognition Letters 116 (2018) 8–14.

[43] Z. Tang, L. Luo, H. Peng, S. Li, A joint residual network with paired relus activation for image super-resolution, Neurocomputing 273 (2018) 37–46.

[44] S. R. Dubey, S. Chakraborty, Average biased relu based cnn descriptor for improved face retrieval, arXiv preprint arXiv:1804.02051 (2018).

[45] Y. Liu, J. Zhang, C. Gao, J. Qu, L. Ji, Natural-logarithm-rectified activation function in convolutional neural networks, in: International Conference on Computer and Communications, 2019, pp. 2000–2008.

[46] S. Gu, W. Li, L. V. Gool, R. Timofte, Fast image restoration with multi-bin trainable linear units, in: IEEE International Conference on Computer Vision, 2019, pp. 4190–4199.

[47] M. Basirat, P. Roth, L* relu: Piece-wise linear activation functions for deep fine-grained visual categorization, in: IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 1218–1227.

[48] C. Gulcehre, M. Moczulski, M. Denil, Y. Bengio, Noisy activation functions, in: International Conference on Machine Learning, 2016, pp. 3059–3068.

[49] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, Y. Bengio, Maxout networks, arXiv preprint arXiv:1302.4389 (2013).

[50] B. Xu, N. Wang, T. Chen, M. Li, Empirical evaluation of rectified activations in convolutional network, arXiv preprint arXiv:1505.00853 (2015).

[51] H. Li, W. Ouyang, X. Wang, Multi-bias non-linear activation in deep neural networks, in: International Conference on Machine Learning, 2016, pp. 221–229.

[52] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 971–980.

[53] J. T. Barron, Continuously differentiable exponential linear units, arXiv (2017) arXiv–1704.

[54] L. Trottier, P. Gigu, B. Chaib-draa, et al., Parametric exponential linear unit for deep convolutional neural networks, in: IEEE International Conference on Machine Learning and Applications, 2017, pp. 207–214.

[55] Y. Li, C. Fan, Y. Li, Q. Wu, Y. Ming, Improving deep neural network with multiple parametric exponential linear units, Neurocomputing 301 (2018) 11–24.

[56] Z. Qiumei, T. Dan, W. Fenghua, Improved convolutional neural network based on fast exponentially linear unit activation function, IEEE Access 7 (2019) 151359–151367.

[57] Y. Ying, J. Su, P. Shan, L. Miao, X. Wang, S. Peng, Rectified exponential units for convolutional neural networks, IEEE Access 7 (2019) 101633–101640.

[58] D. Kim, J. Kim, J. Kim, Elastic exponential linear units for convolutional neural networks, Neurocomputing 406 (2020) 253–266.

[59] Q. Cheng, H. Li, Q. Wu, L. Ma, N. N. King, Parametric deformable exponential linear units for deep neural networks, Neural Networks 125 (2020) 281–289.

[60] J. Si, S. L. Harris, E. Yfantis, A dynamic relu on neural network, in: IEEE Dallas Circuits and Systems Conference, 2018, pp. 1–6.

[61] H. Hu, Vrelu activation functions for artificial neural networks, in: International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, 2018, pp. 856–860.

[62] G. Lin, W. Shen, Research on convolutional neural network based on improved relu piecewise activation function, Procedia Computer Science 131 (2018) 977–984.

[63] D. Macêdo, C. Zanchettin, A. L. Oliveira, T. Ludermir, Enhancing batch normalized convolutional networks using displaced rectifier linear units: A systematic comparative study, Expert Systems with Applications 124 (2019) 271–281.

[64] L. B. Godfrey, An evaluation of parametric activation functions for deep learning, in: IEEE International Conference on Systems, Man and Cybernetics, 2019, pp. 3006–3011.

[65] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, S. Yan, Deep learning with s-shaped rectified linear activation units, in: AAAI Conference on Artificial Intelligence, 2016.

[66] V. S. Bawa, V. Kumar, Linearized sigmoidal activation: A novel activation function with tractable non-linear characteristics to boost representation capability, Expert Systems with Applications 120 (2019) 346–356.

[67] X. Wang, Y. Qin, Y. Wang, S. Xiang, H. Chen, Reltanh: An activation function with vanishing gradient resistance for sae-based dnns and its application to rotating machinery fault diagnosis, Neurocomputing 363 (2019) 88–98.

[68] X. Hu, P. Niu, J. Wang, X. Zhang, A dynamic rectified linear activation units, IEEE Access 7 (2019) 180409–180416.

[69] A. Nicolae, Plu: The piecewise linear unit activation function, arXiv preprint arXiv:1809.09534 (2018).

[70] L. B. Godfrey, M. S. Gashler, A continuum among logarithmic, linear, and exponential functions, and its potential to improve generalization in neural networks, in: International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management, Vol. 1, 2015, pp. 481–486.

[71] B. Grelsson, M. Felsberg, Improved learning in convolutional neural networks with shifted exponential linear units (shelus), in: International Conference on Pattern Recognition, 2018, pp. 517–522.

[72] Y. Yu, K. Adu, N. Tashi, P. Anokye, X. Wang, M. A. Ayidzoe, Rmaf: Relu-memristor-like activation function for deep learning, IEEE Access 8 (2020) 72727–72741.

[73] M. Basirat, P. M. Roth, The quest for the golden activation function, arXiv preprint arXiv:1808.00783 (2018).

[74] S. Scardapane, M. Scarpiniti, D. Comminiello, A. Uncini, Learning activation functions from data using cubic spline interpolation, in: Italian Workshop on Neural Nets, 2017, pp. 73–83.

[75] A. Mishra, P. Chandra, U. Ghose, S. S. Sodhi, Bi-modal derivative adaptive activation function sigmoidal feedforward artificial neural networks, Applied Soft Computing 61 (2017) 983–994.

[76] S. Qian, H. Liu, C. Liu, S. Wu, H. San Wong, Adaptive activation functions in convolutional neural networks, Neurocomputing 272 (2018) 204–212.

[77] E. Alcaide, E-swish: Adjusting activations to different network depths, arXiv preprint arXiv:1801.07145 (2018).

[78] Ö. F. Ertuğrul, A novel type of activation function in artificial neural networks: Trained activation function, Neural Networks 99 (2018) 148–157.

[79] M. Goyal, R. Goyal, B. Lall, Learning activation functions: A new paradigm of understanding neural networks, arXiv preprint arXiv:1906.09529 (2019).

[80] G. Maguolo, L. Nanni, S. Ghidoni, Ensemble of convolutional neural networks trained with different activation functions, arXiv preprint arXiv:1905.02473 (2019).

[81] H. H. Chieng, N. Wahid, P. Ong, S. R. K. Perla, Flatten-t swish: a thresholded relu-swish-like activation function for deep learning, arXiv preprint arXiv:1812.06247 (2018).

[82] N. Patwardhan, M. Ingalhalikar, R. Walambe, Aria: Utilizing richard's curve for controlling the non-monotonicity of the activation function in deep neural nets, arXiv preprint arXiv:1805.08878 (2018).

[83] M. Dushkoff, R. Ptucha, Adaptive activation functions for deep networks, Electronic Imaging 2016 (19) (2016) 1–5.

[84] F. Manessi, A. Rozza, Learning combinations of activation functions, in: IEEE International Conference on Pattern Recognition, 2018, pp. 61–66.

[85] L. R. Sütfeld, F. Brieger, H. Finger, S. Füllhase, G. Pipa, Adaptive blending units: Trainable activation functions for deep neural networks, arXiv preprint arXiv:1806.10064 (2018).

[86] M. Wang, B. Liu, H. Foroosh, Look-up table unit activation function for deep convolutional neural networks, in: IEEE Winter Conference on Applications of Computer Vision, 2018, pp. 1225–1233.

[87] D. Klabjan, M. Harmon, Activation ensembles for deep neural networks, in: IEEE International Conference on Big Data, 2019, pp. 206–214.

[88] C. Eisenach, Z. Wang, H. Liu, Nonparametrically learning activation functions in deep neural nets, in: International Conference on Learning Representations Workshops, 2017.

[89] C. J. Vercellino, W. Y. Wang, Hyperactivations for activation function exploration, in: Conference on Neural Information Processing Systems Workshop on Meta-learning, 2017.

[90] A. D. Jagtap, K. Kawaguchi, G. E. Karniadakis, Adaptive activation functions accelerate convergence in deep and physics-informed neural networks, Journal of Computational Physics 404 (2020) 109136.

[91] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, R. Garcia, Incorporating second-order functional knowledge for better option pricing, in: Advances in Neural Information Processing Systems, 2001, pp. 472–478.

[92] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: International Conference on Artificial Intelligence and Statistics, 2011, pp. 315–323.

[93] H. Zheng, Z. Yang, W. Liu, J. Liang, Y. Li, Improving deep neural networks using softplus units, in: International Joint Conference on Neural Networks, 2015, pp. 1–4.

[94] Q. Liu, S. Furber, Noisy softplus: a biology inspired activation function, in: International Conference on Neural Information Processing, 2016, pp. 405–412.

[95] H. Zhao, F. Liu, L. Li, C. Luo, A novel softplus linear unit for deep convolutional neural networks, Applied Intelligence 48 (7) (2018) 1707–1720.

[96] C. Xu, J. Huang, S.-p. Wang, A.-q. Hu, A novel parameterized activation function in visual geometry group, in: International Conference on Data Science and Business Analytics, 2018, pp. 386–389.

[97] K. Sun, J. Yu, L. Zhang, Z. Dong, A convolutional neural network model based on improved softplus activation function, in: International Conference on Applications and Techniques in Cyber Security and Intelligence, 2019, pp. 1326–1335.

[98] Y. Chen, Y. Mai, J. Xiao, L. Zhang, Improving the antinoise ability of dnns via a bio-inspired noise adaptive activation function rand softplus, Neural Computation 31 (6) (2019) 1215–1233.

[99] D. Misra, Mish: A self regularized non-monotonic neural activation function, arXiv preprint arXiv:1908.08681 (2019).

[100] A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao, Yolov4: Optimal speed and accuracy of object detection, arXiv preprint arXiv:2004.10934 (2020).

[101] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), arXiv preprint arXiv:1606.08415 (2016).

[102] C. Yu, Z. Su, Symmetrical gaussian error linear units (sgelus), arXiv preprint arXiv:1911.03925 (2019).

[103] Q. Su, L. Carin, et al., A probabilistic framework for nonlinearities in stochastic neural networks, in: Advances in Neural Information Processing Systems, 2017, pp. 4486–4495.

[104] J. Lee, K. Shridhar, H. Hayashi, B. K. Iwana, S. Kang, S. Uchida, Probact: A probabilistic activation function for deep neural networks, arXiv preprint arXiv:1905.10761 (2019).

[105] L. Hou, D. Samaras, T. M. Kurc, Y. Gao, J. H. Saltz, Convnets with smooth adaptive activation functions for regression, Proceedings of Machine Learning Research 54 (2017) 430.

[106] Y. Berradi, Symmetric power activation functions for deep neural networks, in: International Conference on Learning and Optimization Algorithms: Theory and Applications, 2018, pp. 1–6.

[107] E. López-Rubio, F. Ortega-Zamorano, E. Domínguez, J. Muñoz-Pérez, Piecewise polynomial activation functions for feedforward neural networks, Neural Processing Letters 50 (1) (2019) 121–147.

[108] F. Farhadi, V. P. Nia, A. Lodi, Activation adaptation in neural networks, arXiv preprint arXiv:1901.09849 (2019).

[109] B. Li, S. Tang, H. Yu, Powernet: Efficient representations of polynomials and smooth functions by deep neural networks with rectified power units, arXiv preprint arXiv:1909.05136 (2019).

[110] M. Telgarsky, Neural networks and rational functions, in: International Conference on Machine Learning, 2017, pp. 3387–3393.

[111] A. Molina, P. Schramowski, K. Kersting, Padé activation units: End-to-end learning of flexible activation functions in deep networks, International Conference on Learning Representations (2020).

[112] A. T. Nicolas Boullé, Yuji Nakatsukasa, Rational neural networks, arXiv preprint arXiv:2004.01902 (2020).

[113] A. Apicella, F. Isgrò, R. Prevete, A simple and efficient architecture for trainable activation functions, Neurocomputing 370 (2019) 1–15.

[114] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, Z. Liu, Dynamic relu, arXiv preprint arXiv:2003.10027 (2020).

[115] M. Wang, B. Liu, H. Foroosh, Wide hidden expansion layer for deep convolutional neural networks, in: IEEE Winter Conference on Applications of Computer Vision, 2020, pp. 934–942.

[116] A. Asif, et al., Learning neural activations, arXiv preprint arXiv:1912.12187 (2019).

[117] S. Scardapane, S. Van Vaerenbergh, S. Totaro, A. Uncini, Kafnets: Kernel-based non-parametric activation functions for neural networks, Neural Networks 110 (2019) 19–32.

[118] S. Scardapane, E. Nieddu, D. Firmani, P. Merialdo, Multikernel activation functions: formulation and a case study, in: INNS Big Data and Deep Learning conference, 2019, pp. 320–329.

[119] S. Scardapane, S. Van Vaerenbergh, A. Hussain, A. Uncini, Complex-valued neural networks with nonparametric activation functions, IEEE Transactions on Emerging Topics in Computational Intelligence (2018).

[120] S. Scardapane, S. Van Vaerenbergh, D. Comminiello, A. Uncini, Widely linear kernels for complex-valued kernel activation functions, in: IEEE International Conference on Acoustics, Speech and Signal Processing, 2019, pp. 8528–8532.

[121] M. Kobayashi, Singularities of three-layered complex-valued neural networks with split activation function, IEEE Transactions on Neural Networks and Learning Systems 29 (5) (2017) 1900–1907.

[122] J. Pennington, S. Schoenholz, S. Ganguli, Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice, in: Advances in Neural Information Processing Systems, 2017, pp. 4785–4795.

[123] E. Sansone, F. G. De Natale, Training feedforward neural networks with standard logistic activations is feasible, arXiv preprint arXiv:1710.01013 (2017).

[124] L. Lu, Y. Shin, Y. Su, G. E. Karniadakis, Dying relu and initialization: Theory and numerical examples, arXiv preprint arXiv:1903.06733 (2019).

[125] D. Arpit, Y. Bengio, The benefits of over-parameterization at initialization in deep relu networks, arXiv preprint arXiv:1901.03611 (2019).

[126] D. Aguirre, O. Fuentes, Improving weight initialization of relu and output layers, in: International Conference on Artificial Neural Networks, 2019, pp. 170–184.

[127] R. Burkholz, A. Dubatovka, Initialization of relus for dynamical isometry, in: Advances in Neural Information Processing Systems, 2019, pp. 2382–2392.

[128] D. Yarotsky, Error bounds for approximations with deep relu networks, Neural Networks 94 (2017) 103–114.

[129] R. Arora, A. Basu, P. Mianjy, A. Mukherjee, Understanding deep neural networks with rectified linear units, arXiv preprint arXiv:1611.01491 (2016).

[130] M. Hein, M. Andriushchenko, J. Bitterwolf, Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 41–50.

[131] S. Goel, S. Karmalkar, A. Klivans, Time/accuracy tradeoffs for learning a relu with respect to gaussian marginals, in: Advances in Neural Information Processing Systems, 2019, pp. 8582–8591.

[132] S. Dittmer, J. Emily, P. Maass, Singular values for relu layers, IEEE Transactions on Neural Networks and Learning Systems (2019).

[133] A. Kristiadi, M. Hein, P. Hennig, Being bayesian, even just a bit, fixes overconfidence in relu networks, arXiv preprint arXiv:2002.10118 (2020).

[134] B. Karlik, A. V. Olgac, Performance analysis of various activation functions in generalized mlp architectures of neural networks, International Journal of Artificial Intelligence and Expert Systems 1 (4) (2011) 111–122.

[135] G. Alcantara, Empirical analysis of non-linear activation functions for deep neural networks in classification tasks, arXiv preprint arXiv:1710.11272 (2017).

[136] H. K. Vydana, A. K. Vuppala, Investigative study of various activation functions for speech recognition, in: National Conference on Communications, 2017, pp. 1–5.

[137] D. Pedamonti, Comparison of non-linear activation functions for deep neural networks on mnist classification task, arXiv preprint arXiv:1804.02763 (2018).

[138] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall, Activation functions: Comparison of trends in practice and research for deep learning, arXiv preprint arXiv:1811.03378 (2018).

[139] K. Eckle, J. Schmidt-Hieber, A comparison of deep networks with relu activation function and linear spline-type methods, Neural Networks 110 (2019) 232–242.

[140] M. M. Lau, K. H. Lim, Review of adaptive activation function in deep neural network, in: IEEE-EMBS Conference on Biomedical Engineering and Sciences, 2018, pp. 686–690.

[141] A. K. Dubey, V. Jain, Comparative study of convolution neural network's relu and leaky-relu activation functions, in: Applications of Computing, Automation and Wireless Systems in Electrical Engineering, Springer, 2019, pp. 873–880.

[142] C. Banerjee, T. Mukherjee, E. Pasiliao Jr, An empirical study on generalizations of the relu activation function, in: ACM Southeast Conference, 2019, pp. 164–167.

[143] T. Villmann, J. Ravichandran, A. Villmann, D. Nebel, M. Kaden, Activation functions for generalized learning vector quantization-a performance comparison, arXiv preprint arXiv:1901.05995 (2019).

[144] G. Castaneda, P. Morris, T. M. Khoshgoftaar, Evaluation of maxout activations in deep learning across several big data domains, Journal of Big Data 6 (1) (2019) 72.

[145] Y. Wang, Y. Li, Y. Song, X. Rong, The influence of the activation function in a convolution neural network model of facial expression recognition, Applied Sciences 10 (5) (2020) 1897.

[146] A. Apicella, F. Donnarumma, F. Isgrò, R. Prevete, A survey on modern trainable activation functions, arXiv preprint arXiv:2005.00817 (2020).

[147] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in: Bio-inspired Neurocomputing, 2020, pp. 203–224.

[148] A. Krizhevsky, Learning multiple layers of features from tiny images, Tech Report, Univ. of Toronto (2009).

[149] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).

[150] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).

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

[152] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[153] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.

[154] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[155] K. Papineni, S. Roukos, T. Ward, W.-J. Zhu, Bleu: a method for automatic evaluation of machine translation, in: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, 2002, pp. 311–318.