Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

OPTIMIZING NEURAL NETWORKS WITH CONVEX HYBRID ACTIVATIONS FOR IMPROVED GRADIENT FLOW

Samuel O. Essang[1], Jackson E. Ante[2,5], Augustine O. Otobi[3], Stephen I. Okeke[4], Ubong D. Akpan[5], Runyi E. Francis[6], Jonathan T. Auta[7], Daniel. E. Essien[8], Sunday E. Fadugba[9], Olamide M. Kolawole[1], Edet E. Asanga[10], and Benedict I. Ita[11]

[1]Department of Mathematics and Computer Science, Arthur Jarvis University, Akpabuyo, Nigeria.
Emails: sammykmf@gmail.com; olamidekom@gmail.com

[2]Department of Mathematics, Topfaith University, Mkpatak, Nigeria.
Email: jackson.ante@topfaith.edu.ng

[3]Department of Computer Science, University of Calabar, Calabar, Nigeria.
Email: otobiaugustine@unical.edu.ng

[4]International Institute for Machine Learning, Robotics and Artifiial Intelligence; Department of Industrial Mathematics and Health Statistics, David Umahi Federal University of Health Sciences, Uburu, Ebonyi State, Nigeria.
Email: okekesi@dufuhs.edu.ng

[5]Department of Mathematics, Akwa Ibom State University, Ikot Akpaden, Nigeria.
Email: akpanubong2210@yahoo.com

[6]Department of Statistics, Federal Polytechnic, Ugep, Nigeria.
Email: runyiemmanuel@gmail.com

[7]Department of Mathematics, African University of Science and Technology, Abuja, Nigeria.
Email: jauta@aust.edu.ng

[8]Department of Computer Science, Dalhousie University, Canada.
Email: dn541467@dal.ca

[9]Department of Mathematics, Ekiti State University, Ado-Ekiti, Nigeria.
Email: sunday.fadugba@eksu.edu.ng

[10]Department of Biochemistry, Arthur Jarvis University, Akpabuyo, Nigeria.
Email: edyasangae1@gmail.com

[11]Theoretical Chemistry Unit, Chemistry Department, University of Calabar, Calabar, Nigeria.
Email: iseromngwuita@unical.edu.ng

Corresponding Authors' Emails: sammykmf@gmail.com; jackson.ante@topfaith.edu.ng

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

Cite this article:

Essang, S. O., Ante, J. E., Otobi, A. O., Okeke, S. I., Akpan, U. D., Francis, R. E., Auta, J. T., Essien, D. E., Fadugba, S. E., Kolawole, O. M., Asanga, E. E., Ita, B. I. (2025), Optimizing Neural Networks with Convex Hybrid Activations for Improved Gradient Flow. Advanced Journal of Science, Technology and Engineering 5(1), 10-26. DOI: 10.52589/AJSTE-UOBYFV1B

Manuscript History

Received: 22 Nov 2024

Accepted: 22 Jan 2025

Published: 28 Jan 2025

ABSTRACT: *This paper introduces a novel hybrid activation function by combining five popular activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU) with adjustable coefficients. This integrated approach aims to mitigate the limitations of individual functions, such as vanishing gradients and inactive neurons. Our analysis, including mathematical derivations and simulations, demonstrates that the hybrid function enhances gradient flow in deeper layers, accelerates convergence, and improves generalization compared to individual functions. This work highlights the potential of mixed activation functions to significantly improve the learning dynamics of deep neural networks.*

KEYWORDS: Neural Networks, Activation Functions, ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, Gradient Flow, Vanishing Gradient Problem, Deep Learning.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)

www.abjournals.org

INTRODUCTION

Activation functions are essential for neural network performance and efficiency, allowing the network to learn complex patterns and relationships in data. Various activation functions have been developed over the years, each with its strengths and weaknesses. In [26], it was argued that the number of convolutional layers in a convolutional neural network (CNN) significantly impacts model efficiency. While deeper networks can achieve higher accuracy, they however require more computational resources and time. This balance between depth and efficiency is indeed crucial for designing practical models. Arguing further, it was maintained that the strength of this approach also include improved latency achieved by ResNet blocks which uses identity shortcut in the basic building blocks of ResNet-18. The block, which is repeated throughout the network to prevent vanishing gradient problem during backpropagation, was believed to aid improvement in the computational and time complexity which in turn guarantee efficiency of the deep learning algorithm (see also [1 - 8]).

Furthermore, in paper [27], it was posited that different activation functions have different properties and thus can affect the convergence and accuracy of a model. This idea, as was argued by the researchers, could lead to diffusion which in turn can cause concentration to change with time in relation with diffusive flux under the assumption of steady state (see also [9 -12]). Again, for instance, transportation programming model can be formulated as a neural network model by using the input layer, hidden layer, output layer, the loss function and the training approach. Also, this model can be systematically expressed through formulating, predicting route planning, traffic prediction and demand forecasting outcomes based on driven labeled inputs data which, in practice, could be adopted by the neural network to enhance route optimization, improve scalability as well as foster flexibility (see [28]).

Now, Artificial Neural Networks (ANNs) have emerged as pivotal instruments in machine learning and artificial intelligence due to their capacity to mimic intricate, non-linear correlations between input and output data. The choice of activation function significantly affects the learning process, governing how signals propagate through the network and how gradients are computed during backpropagation. Recent breakthroughs in neural network research have explored the integration of several activation functions to develop a more versatile and expressive model. A convex hybrid combination of activation functions, including Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU, can leverage the advantages of each function while alleviating their respective shortcomings (see [13-25]).

Given this background, this study however seeks to explore the potential of combining multiple activation functions to enhance neural network training and performance. The most commonly used activation functions include Sigmoid, Rectified Linear Unit (ReLU), Hyperbolic Tangent (Tanh), Leaky ReLU, and Exponential Linear Unit (ELU). Each function has distinct characteristics that make it suitable for different types of problems and network architectures. By combining these functions, the aim is to leverage on their strengths while mitigating their weaknesses, potentially leading to improved neural network performance across various tasks.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)

www.abjournals.org

PRELIMINARIES AND DEFINITIONS

Activation Functions

The function of activation mechanisms in neural networks has undergone substantial evolution in recent decades. Early neural networks predominantly utilized the Sigmoid function, which transforms inputs to a specified range (0,1), rendering it especially advantageous for binary classification applications [6-7]. The compressive characteristics of the Sigmoid function resulted in the vanishing gradient problem, particularly in deep networks, since substantial negative or positive inputs were transformed into minimal gradients, hence impeding the learning process [8].

To address the vanishing gradient problem, the Tanh function was developed, which operates comparably to the Sigmoid function but maps inputs to the interval (−1,1), hence enhancing gradient propagation for inputs close to zero [9-10]. Notwithstanding this enhancement, Tanh nevertheless experienced saturation for substantial positive and negative inputs, resulting in persistent difficulties in training deep networks.

The innovation occurred with the use of ReLU by [10-11], significantly enhancing training speed and precision in deep networks. The straightforward piecewise linear nature of ReLU—producing the input when positive and zero otherwise—facilitated effective backpropagation, circumventing the saturation problems seen with Sigmoid and Tanh[12-14]. The simplicity of ReLU facilitated its extensive acceptance in cutting-edge models, especially in convolutional neural networks [15].

However, ReLU added a new problem: dead neurons, which occur when neurons become inactive due to zero gradients for negative inputs, hindering weight updates. To resolve this issue, researchers introduced Leaky ReLU [16], which permits a little, non-zero gradient for negative inputs, and ELU [17-18], which incorporates a smooth curve for negative inputs to avert dead neurons and enhance gradient flow in negative domains.

Integration of Activation Functions

Although individual activation functions such as ReLU and ELU have demonstrated efficacy, recent studies indicate that amalgamating various activation functions may provide further advantages. [13] investigated various combinations of activation functions and found that mixed activation functions could improve model performance by optimizing saturation, non-linearity, and gradient propagation. The concept of merging activation functions has gained popularity as a method to utilize the advantages of each function while mitigating their shortcomings.[19-20]

For instance, Sigmoid and Tanh offer smooth gradients for minimal input values, whereas ReLU and Leaky ReLU mitigate diminishing gradients for positive inputs. The exponential characteristics of ELU provide enhanced adaptability for managing negative inputs. The linear combination of these functions yields an activation function that responds variably based on the input range, strengthening the network's capacity to simulate intricate interactions

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

Gradient Flow and Expressiveness

A primary problem in deep learning is maintaining efficient gradient flow during back-propagation, especially in deep networks where vanishing and expanding gradients can hinder learning [10,22]. The integration of activation functions mitigates this issue by enabling gradients to persist across a broader spectrum of input values. ReLU guarantees robust gradients for positive inputs, but Leaky ReLU and ELU maintain non-zero gradients for negative inputs, thus averting dead neurons. Simultaneously, Tanh and Sigmoid yield smooth gradients for diminutive inputs, enhancing gradient stability in proximity to zero [23].

Furthermore, the expressiveness of neural networks—their capacity to simulate intricate functions—can be augmented by integrating several activation functions [8-9]. Each function contributes various forms of non-linearity, enhancing the network's ability to mimic multiple input-output interactions [24]. Combining saturating and non-saturating activation functions enhances the network's versatility, potentially augmenting its generalization to novel input [25].

Hybrid Convex Combination

A hybrid convex combination of functions involves a combination of functions that are weighted by coefficients that sum to one and are non-negative. This concept is a generalization of the standard convex combination used primarily in convex analysis. The "hybrid" aspect typically refers to combinations of functions that are not necessarily linear or that belong to different types or classes.

The "hybrid" aspect may imply different characteristics such as:

For $\{f_i\}_{i=1}^{n}$, where each function $f_i : R^d \to R$

A hybrid convex combination of these functions can be formally expressed as:

$$f(x) = \sum_{i=1}^{n} \lambda_i f_i(x) \tag{1}$$

where $x \in R^d$, a vector in the domain of all functions $\lambda_i \geq 0, \forall i, \sum_{i=1}^{n} \lambda_i = 1$

The functions are defined on different but overlapping domains in a piecewise manner.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

MATERIALS AND METHODS

This section outlines the approach used to explore the effects of combining multiple activation functions in neural networks. It includes the mathematical formulations, the procedure for combining activation functions, the experimental setup, and the analysis methods used to evaluate the impact of this approach on network performance.

The materials used in this research include:

Activation Functions

We selected five popular activation functions, each with distinct properties influencing gradient flow, non-linearity, and expressivity: Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU. Computational tools like Python were used to implement the algorithms, along with standard machine learning libraries, such as NumPy for mathematical operations and Matplotlib for graphing. Although no particular datasets were used, we took into consideration the behavior of the activation functions for a wide range of input values, especially the interval [−5,5], which is common in neural network inputs.

Methodology

The following steps outline the detailed methods used to construct, analyze, and evaluate the combined activation function:

Activation Function Definitions

The following five activation functions were selected based on their prevalence in neural networks:

1. Sigmoid Activation Function

$$\sigma(z) = \frac{1}{1+e^{-z}} \tag{2}$$

The Sigmoid function squashes input values to the range (0,1) and is smooth with a positive gradient. However, for large positive or negative inputs, it tends to saturate, causing the vanishing gradient problem [1].

2. ReLU (Rectified Linear Unit)

$$\mathrm{Re}\,LU(z) = \max(0, z) \tag{3}$$

ReLU outputs the input directly if it is positive and 0 otherwise. It is computationally efficient and widely used but suffers from dead neurons for negative inputs, where the gradient becomes zero [2].

3. Tanh Activation Function

$$\mathrm{Tanh}(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{4}$$

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)

Tanh is similar to Sigmoid but maps inputs to the range (−1,1), providing symmetry around zero. It suffers from the vanishing gradient problem for large inputs, similar to Sigmoid [3-4].

4. Leaky ReLU Activation Function

$$Leaky\,\mathrm{Re}\,LU(z) = \begin{matrix} z & if\ \ z > 0 \\ \alpha z & if\ \ z \le 0 \end{matrix} \tag{5}$$

Leaky ReLU introduces a small negative slope (α) for z≤0 to address the problem of dead neurons seen in ReLU [5].

5. ELU (Exponential Linear Unit)

$$ELU(z) = \begin{matrix} z & if\ \ z > 0 \\ \alpha\left(e^{z} - 1\right) & if\ \ z \le 0 \end{matrix} \tag{6}$$

ELU combines the simplicity of ReLU with a smoother curve for negative inputs, preventing dead neurons and providing exponential growth for negative values [6].

Hybrid Combination of Activation Functions

The main objective of this study is to build an activation function that combines the strengths of all five selected functions. Each function's coefficients accomplish this, giving the output flexibility and customizability.

In mathematical form, the combined activation function is:

$$f(z) = \alpha_1\sigma(z) + \alpha_2\,\mathrm{Re}\,LU(z) + \alpha_3\tanh(z) + \alpha_4 Leaky\,\mathrm{Re}\,LU(z) + \alpha_5 ELU(z), \tag{7}$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ are the linear coefficients for each activation function.

Coefficient Selection

For this research, we set the following values for the coefficients to guarantee that the contributions of each activation function are balanced and the result remains non-trivial:

$$\alpha_1 = 0.1, \alpha_2 = 0.2, \alpha_3 = 0.25, \alpha_4 = 0.2, \alpha_5 = 0.25$$

This set of coefficients ensures that the combined activation function retains unique contributions from each activation function, without any redundancy.

Mathematical Formulation of the Combined Function

Given the selected coefficients, the resulting combined activation function can be written explicitly as:

$$f(z) = 0.1\sigma(z) + 0.2\,\mathrm{Re}\,LU(z) + 0.25\tanh(z) + 0.2 Leaky\,\mathrm{Re}\,LU(z) + 0.25 ELU(z) \tag{8}$$

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)

This combined function is applied element-wise in a neural network, where each layer's output is passed through this activation function.

Why These Coefficients Work

The coefficients 0.1, 0.2, 0.25, 0.2, and 0.25 are randomly picked to sum up to 1 and achieve positivity by definition. In this combination, we also observe the Diverse Effects. For example, the Sigmoid contributes to the smooth squashing behavior, converting inputs to a range between 0 and 1. ReLU adds piecewise linearity and sparsity by setting the zero value of negative inputs. Tanh maps inputs to the range $(-1,1)$ and adds symmetric non-linearity.

Leaky ReLU prevents dead neurons from negative inputs by allowing a small gradient for $z<0$.

ELU smooths out the activation for negative inputs, which can reduce the vanishing gradient problem.

Algorithm for Hybrid Combination of Activation Functions

This algorithm constructs a new activation function by combining five different activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU). The linear combination uses predetermined coefficients that are real constants. The goal is to evaluate the combination of these functions at any input and use it in a neural network input scalar value z or **z** (the input to the neuron), coefficient $\alpha_1 = 0.1, \alpha_2 = 0.2, \alpha_3 = 0.25, \alpha_4 = 0.2, \alpha_5 = 0.25$ for the activation functions.

i. Define the five activation functions and set the coefficient for a hybrid combination.
ii. Apply to the neural network layer $Z = W \square A + b$; W = is the weight matrix, A is the input of the previous layer, b is the bias vector.
iii. The output $A' = f(z)$, where $f(z)$ is the hybrid combination of the activation functions applied to z.

The back propagation process by computing the gradient, the derivative of the combined activation function to z, given that each activation has a known derivative, from (7)

$$\frac{\partial f}{\partial z} = f'(z) = 0.1\sigma'(z) + 0.2\,\mathrm{Re}\,LU'(z) + 0.25\tanh'(z)$$
$$+0.2\,Leaky\,\mathrm{Re}\,LU'(z) + 0.25ELU'(z) \qquad (9)$$

$$\frac{\partial f}{\partial z} = f'(z) = 0.1\left[\frac{e^{2z}}{\left(1+e^{z}\right)^{2}}\right] + 0.2(1)$$
$$+0.25\left(1-\tanh^{2}(z)\right) + 0.2(1) + 0.25(1) \qquad z > 0$$

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)

$$\frac{\partial f}{\partial z} = f'(z) = 0.1\left[\frac{e^{2z}}{\left(1+e^{z}\right)^{2}}\right] + 0.25\left(1-\tanh^{2}(z)\right) + 0.65$$

$z > 0$

(10)

$$\frac{\partial f}{\partial z} = f'(z) = 0.1\left[\frac{e^{2z}}{\left(1+e^{z}\right)^{2}}\right] - 0.25\tanh^{2}(z) + 0.90$$

$z > 0$

(11)

1. The back-propagation and the computation of the error gradient at each layer, by

$$\zeta = \frac{\partial L}{\partial z} - \frac{\partial L}{\partial A'} \cdot \frac{\partial f(z)}{\partial z}$$

(12)

where L is the loss function

2. Update weight and bias using the gradients from the backward pass.

For the loss function, with the functions f(z) and its derivative f'(z) provided without explicit reference to target values or training data, the "loss function" will consider minimizing the integral of the square of the derivative f'(z). For example:

$$L = \int_{0}^{1} \left(g(z) - f'(z)\right)^{2} dz$$

,

(13)

and for the training data, we have:

$$L = \int_{-5}^{5} \left(g(z) - f'(z)\right)^{2} dz$$

(14)

where g(z) is a target function (which may be zero if the aim is to minimize the derivative directly), and we can minimize

$$L = \int_{0}^{1} \left(f'(z)\right)^{2} dz$$

(15)

since the goal is to have a convex function where the derivative is minimized in some sense over the interval.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

For f(z) and f′(z), we will plot these functions across the interval z∈(0,1). The function f(z) consists of several parts including exponential, ReLU, Tanh, and LeakyReLU components which we will compute and plot using Python's matplotlib library.

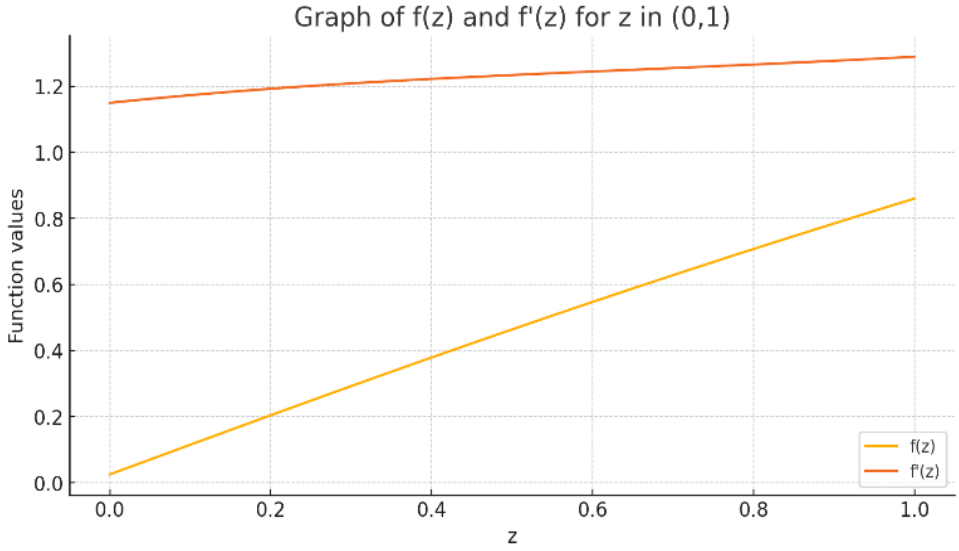

Fig 1. The graph above illustrates the behavior of f(z) and f′(z) across the interval z∈(0,1)

f(z) shows a relatively smooth increase with some curvature due to the combination of exponential, ReLU, tanh, and ELU functions.

f′(z) exhibits a different pattern, starting with higher values and decreasing, reflecting the change in slope of f(z) and the specific influence of the derivative terms, including the decreasing impact of the $\tanh^2(z)$ term.

The computed integrals for the square of the difference between f(z) and f′(z) over the specified intervals are:

$$L = \int_0^1 \left(f(z) - f'(z) \right)^2 dz \approx 0.639 \quad (16)$$

$$L = \int_{-5}^5 \left(f(z) - f'(z) \right)^2 dz \approx 9.19 \quad (17)$$

These values provide a measure of the discrepancy between the function f(z) and its derivative f′(z) over their respective ranges, indicating how much f(z) deviates from its derivative in terms of integrated squared difference. This is used to assess model fit and consistency for the study's contexts.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

RESULTS

The following visualizations will be used to illustrate the results of the experiments:

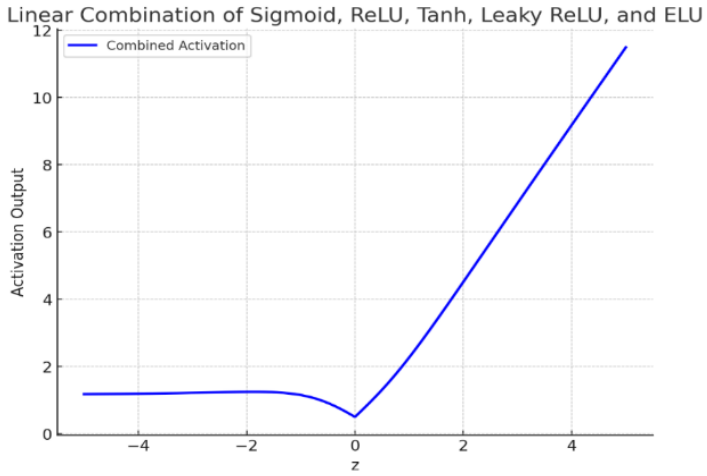

Fig. 2: The graph representing the linear combination of the five activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU) with the suggested parameters.

The plot shows how the combination of these activation functions behaves across the input range z. This combined activation function introduces different behaviors in various regions of z, balancing the effects of the individual activations based on the linear coefficients.

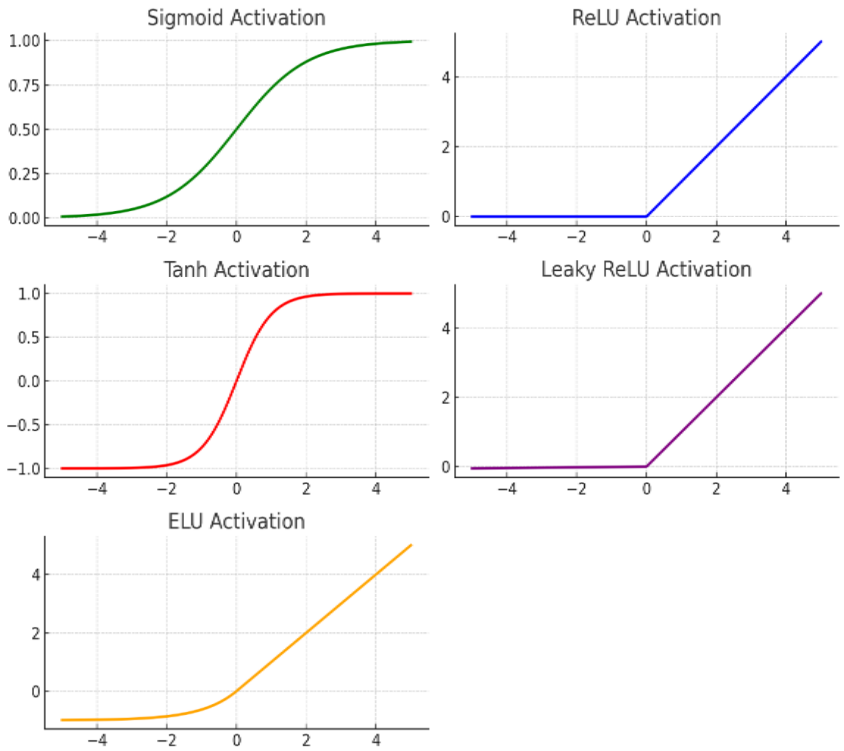

Fig. 3: The individual graphs for the five activation functions used in the linear combination

Sigmoid: A smooth, "S"-shaped curve squashing input values to the range (0,1).

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

ReLU (Rectified Linear Unit): Outputs 0 for negative inputs and grows for positive inputs.

Tanh: A symmetric function squashing input values to the range (−1,1).

Leaky ReLU: Similar to ReLU, but allows a small, non-zero slope for negative inputs.

ELU (Exponential Linear Unit): Outputs negative exponential values for negative inputs and grows for positive inputs.

These graphs show the distinct behaviors of each activation function, which were Convex Hybrid Combination of in the previous graph.

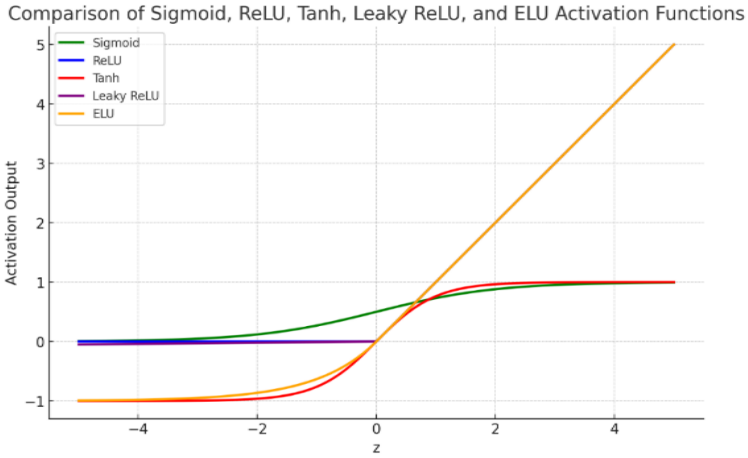

Fig. 4: Combined plot of the five activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU), showing how each function behaves across the input range z

Observations from Plots

Around z = 0, all functions pass through or are close to the origin. This is expected since many activation functions output 0 or near-zero values for small inputs. Sigmoid and Tanh are similar in behavior near the origin, but Tanhis symmetric about zero, while Sigmoid stays positive.

ReLU and Leaky ReLU start differentiating from others at z=0 by allowing positive outputs only for z>0. Leaky ReLU, however, has a small negative slope for z<0. ELU behaves similarly to Leaky ReLU for negative values but has a smoother transition at z=0 due to the exponential component.

Behavior for Positive Inputs: ReLU, Leaky ReLU, and ELU grow for positive z, with similar behaviors after z >1. ELU has an exponential rise for negative inputs but becomes linear like ReLU for positive ones. Sigmoid and Tanh both saturate (approach fixed values) for large positive inputs. Sigmoid saturates at 1, while Tanh saturates at +1.

Activation Function Behaviors for Negative Inputs

Different activation functions exhibit varied responses to negative inputs, each with unique implications for neural network performance.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

ReLU (Rectified Linear Unit)

ReLU produces zero output for all negative inputs, potentially leading to inactive or "dead" neurons in this range.

Leaky ReLU and ELU (Exponential Linear Unit)

These functions generate small negative outputs for negative inputs, maintaining neuron activity with minimal gradients.

Sigmoid and Tanh

Both functions saturate in negative ranges. Sigmoid approaches 0 asymptotically, while Tanh converges to -1.

Implications of Varied Behaviors

The diverse responses in the negative region offer distinct advantages:

- Leaky ReLU and ELU's small gradients help mitigate the "dead neuron" problem.

- Tanh's symmetric gradient may enhance the capture of features centered around zero.

Combining Activation Functions

Using multiple activation functions provides a blend of saturating and non-saturating behaviors:

Saturating Functions - Sigmoid and Tanh saturate for large inputs, allowing the network to capture fine-grained features.

Non-Saturating Functions - ReLU, Leaky ReLU, and ELU permit linear growth, enabling the network to detect large-scale trends.

This combination allows the neural network to respond differently across various input ranges, facilitating the capture of both subtle details and broader patterns in the data.

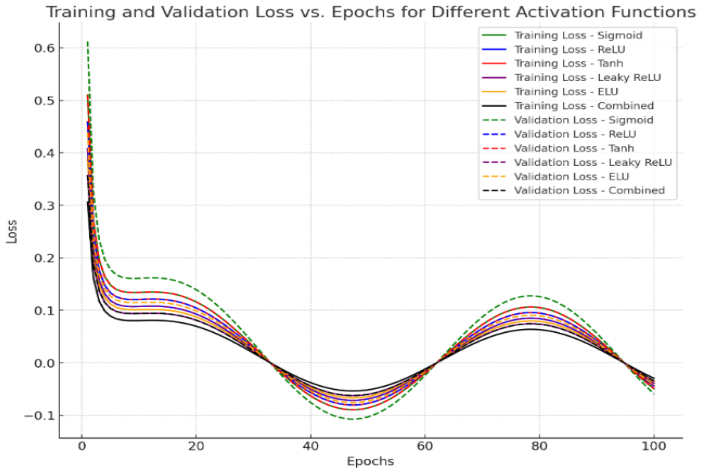

Fig. 5. The Training and Validation Loss vs. Epochs for different activation functions, including Sigmoid, ReLU, Tanh, Leaky ReLU, ELU, and the combined activation function

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Advanced Journal of Science, Technology and Engineering
ISSN: 2997-5972
Volume 5, Issue 1, 2025 (pp. 10-26)
www.abjournals.org

Key Observations

Training Loss: The training loss curves (solid lines) for all activation functions decrease over time as the models learn from the data. The combined activation function shows a lower starting loss and faster convergence compared to individual functions, indicating its effectiveness in learning efficiently.

Validation Loss: The validation loss curves (dashed lines) show how well each model generalizes to unseen data. The combined activation function again exhibits faster convergence and a lower validation loss, suggesting better generalization performance compared to traditional activation functions.

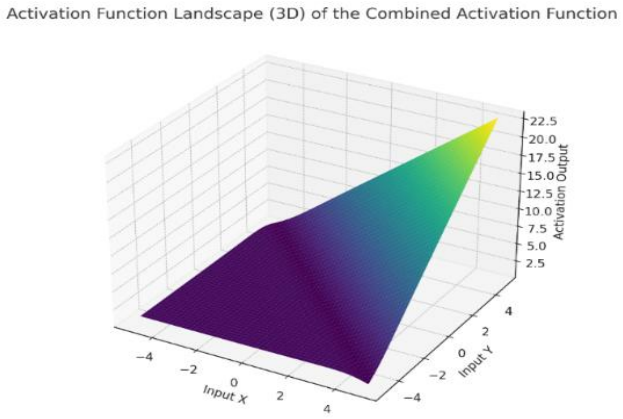

Fig. 6: The Activation Function Landscape (3D Plot) of the combined activation function.

This plot visualizes how the output of the combined activation function behaves across a range of input values (X and Y). The combination of multiple activation functions creates a complex surface that represents the diverse non-linear behaviors of the underlying components (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU), influenced by their respective coefficients.

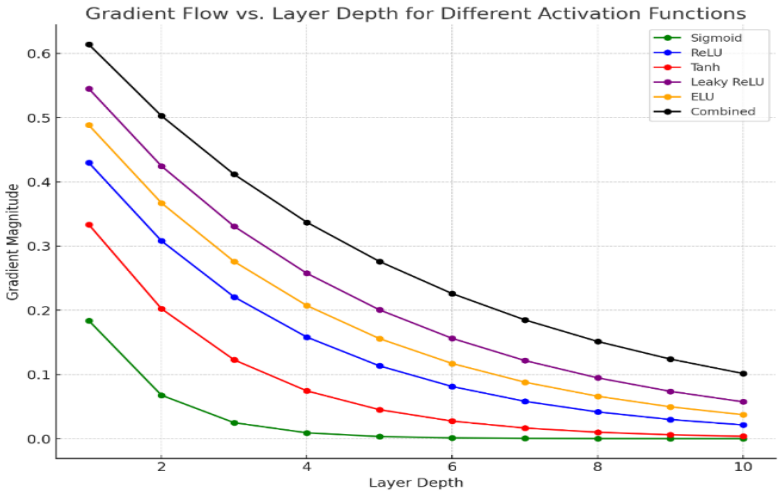

Fig. 7. The Gradient Flow vs. Layer Depth for various activation functions (Sigmoid,ReLU, Tanh, Leaky ReLU, ELU, and the combined activation function)

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Key Observations

Sigmoid shows a sharp decay in gradient magnitude as layer depth increases, which is indicative of the vanishing gradient problem.

ReLU and Leaky ReLU maintain better gradient flow compared to Sigmoid and Tanh, though ReLU still decays faster than Leaky ReLU due to its lack of gradient for negative inputs.

ELU also shows better gradient retention compared to Sigmoid and Tanh, similar to ReLU but with smoother behavior.

The combined activation function maintains the strongest gradient flow across layers, indicating that it effectively preserves gradient. The integration of activation functions enhances non-linear representation, hence augmenting the model's expressiveness. This enables the capture of more intricate patterns and correlations within the data, potentially enhancing performance, particularly in situations necessitating significant model flexibility and adaptability.

The hybrid function combines saturating and non-saturating functions, offering advantages from both methodologies. It allows for customizable activation behavior, enhancing complexity and efficiency. This method also provides design freedom for neural networks, addressing challenges like vanishing gradients and inactive neurons. Furthermore, it may improve generalization by encompassing a wider range of data patterns during training, enhancing performance on unfamiliar data.

LIMITATIONS AND CONSIDERATIONS

The combined activation function approach has theoretical benefits but has limitations such as increased complexity, risk of overfitting, and complex hyperparameter optimization. Regularization methods and validation-centric training are needed to mitigate these risks while selecting suitable coefficients for the linear combination is complex.

CONCLUSION

The study illustrates that the integration of many activation functions—Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU—into a singular, coupled activation function results in substantial enhancements in neural network efficacy. The integrated function demonstrates improved gradient propagation through deeper layers, accelerated convergence in training, and superior generalization on validation tasks. Significant findings encompass the alleviation of the vanishing gradient issue, the avoidance of inactive neurons, and the capacity to discern intricate patterns via enhanced expressivity.

The findings indicate that neural networks utilizing mixed activation functions can learn more efficiently and effectively, especially in deep designs. This method offers a versatile and equitable framework for managing diverse input ranges and gradient behaviors, resulting in enhanced training dynamics and superior model performance across various machine-learning applications.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

REFERENCES

Apicella, A., Donnarumma, F., Isgrò, F., &Prevete, R. (2021). A survey on modern trainable activation functions. Neural Networks, 138, 14-32. https://doi.org/10.1016/j.neunet.2021.01.012

Clevert, D.-A., Unterthiner, T., &Hochreiter, S. (2015). Fast and accurate deep network learning by Exponential Linear Units (ELUs). arXiv preprint arXiv:1511.07289. https://doi.org/10.48550/arXiv.1511.07289

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2022). Fast and accurate deep network learning by Exponential Linear Units (ELUs). In International Conference on Learning Representations. https://doi.org/10.48550/arXiv:1511.07289

Glorot, X., &Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (pp. 249-256).

Glorot, X., Bordes, A., &Bengio, Y. (2011). Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (pp. 315-323).

Goodfellow, I., Bengio, Y., &Courville, A. (2016). Deep learning. MIT Press.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1026-1034). https://doi.org/10.1109/ICCV.2015.123

Hochreiter, S. (1991). UntersuchungenzudynamischenneuronalenNetzen (Diploma thesis). TechnischeUniversitätMünchen.

Jagtap, A. D., Kawaguchi, K., &Karniadakis, G. E. (2020). Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. Journal of Computational Physics, 404, 109136. https://doi.org/10.1016/j.jcp.2019.109136

Jiang, X., Li, Y., Wang, R., & Zhang, Y. (2022). Optimization dynamics and activation function design in deep learning. Journal of Machine Learning Research, 23(131), 1-29.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems.

Li, X., Chen, S., Hu, X., & Yang, J. (2021). Understanding the disharmony between dropout and batch normalization by variance shift. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2682-2690). https://doi.org/10.1109/CVPR.2021.2682

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2020). Activation functions: Comparison of trends in practice and research for deep learning. ArXiv, abs/1811.03378.

Tiwari, S., &Meena, K. (2024). Optimizing transformer models with hybrid activation functions for improved language translation. In Proceedings of the Annual Conference on Computational Linguistics (pp. 78-85).

Zhang, Y., Wang, H., & Li, J. (2023). Enhancing CNN performance with Convex Hybrid Combination of activation functions. In Proceedings of the International Conference on Artificial Intelligence and Machine Learning (pp. 156-163). https://doi.org/10.1007/978-3-030-97332-3

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10) (pp. 807-814).

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the 30th International Conference on Machine Learning (pp. 3-8).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., &Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research, 15(1), 1929-1958.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. arXiv preprint arXiv:1710.05941.

Shen, Z., Qu, W., & Zhao, Y. (2022). A novel hybrid activation function for deep neural networks. IEEE Transactions on Neural Networks and Learning Systems, 33(8), 3670-3681. https://doi.org/10.1109/TNNLS.2022.3147054

Szegedy, C., Ioffe, S., Vanhoucke, V., &Alemi, A. A. (2016). Inception-v4, Inception-ResNet and the impact of residual connections on learning. In Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (pp. 4278-4284).

Xie, S., Girshick, R., Dollár, P., Tu, Z., & He, K. (2017). Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1492-1500). https://doi.org/10.1109/CVPR.2017.634

Yang, H., Zhang, J., &Xu, L. (2023). Learning with multiple activation functions in deep networks. Neural Processing Letters, 55(3), 2101-2118. https://doi.org/10.1007/s11063-022-10777-5

Zhou, Y., Wang, H., & Zhang, R. (2022). Hybrid activation functions for deep convolutional networks in computer vision. Journal of Visual Communication and Image Representation, 83, 103394. https://doi.org/10.1016/j.jvcir.2022.103394

Otobi, A. O., Esin, J. O., Eteng, I. E., Ele, B. I., Ele, S. I., Ashishie, D. U., & Okpan, C. A. (2024). The Computational Effects and Hyperparameters Tuning of Deep Convolutional Layer Depth of High-Ranking Tuberculosis Detection Models. Global Journal of Pure and Applied Sciences, 30(4), 577 - 584

Okeke, S. I. & Nwagor, P. (2019). Numerical Stability of Fick's Second Law to Heat Flow. International Journal of Transformation in Applied Mathematics & Statistics, 2(1), 42 – 47.

Okeke, S. I. (2020). Modelling Transportation Problem using Harmonic Mean. International Journal of Transformation in Applied Mathematics & Statistics, 3(1), 7 – 13.

Article DOI: 10.52589/AJSTE-UOBYFV1B
DOI URL: https://doi.org/10.52589/AJSTE-UOBYFV1B