IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

Optimizing Neural Networks with Linearly Combined Activation Functions: A Novel Approach to Enhance Gradient Flow and Learning Dynamics

S. O. Essang [1*], J. E. Ante [2], S. E. Fadugba [3], J. T. Auta [4], J. N. Ezeorah [5], R. E. Francis [6], A. O. Otobi[7]

1. Department of Mathematics and Computer Science, Arthur Jarvis University, Akpabuyo, Nigeria.
2. Department of Mathematics, Topfaith University, Mkpatak.
3. Department of Mathematics, Ekiti State University, Ado Ekiti, Nigeria.
4. Department of Pure and Applied Mathematics, African University of Science and Technology, Abuja.
5. Department of Mathematics, University of Calabar, Calabar, Nigeria.
6. Department of Statistics, Federal Polythecnic Ugep, Cross River State, Nigeria.
7. Department of Computer Science, University of Calabar, Calabar, Nigeria.
* Corresponding author: sammykmf@gmail.com*, jackson.ante@topfaith.edu.ng, sunday.fadugba@eksu.edu.ng, jauta@aust.edu.ng, reverendjohnmary@gmail.com, runyifrancis@fedpolyugep.edu.ng, otobiaugustine@unical.edu.ng

Article Info
Received: 25 October 2024 Revised: 17 January 2025
Accepted: 16 May 2025 Available online: 30 May 2025

Abstract

Activation functions are crucial for the efficacy of neural networks as they introduce non-linearity and affect gradient propagation. Traditional activation functions, including Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU, possess distinct advantages but also demonstrate limits such as vanishing gradients and inactive neurons. This research introduces an innovative method that linearly integrates five activation functions using linearly independent coefficients to formulate a new hybrid activation function. This integrated function seeks to harmonize the advantages of each element, alleviate their deficiencies, and enhance network training and generalization. Our mathematical study, graphical visualization, and hypothetical tests demonstrate that the combined activation function provides enhanced gradient flow in deeper layers, expedited convergence, and improved generalization relative to individual activation functions. Quantitative metrics demonstrate enhanced gradient flow, expedited convergence, and improved generalization relative to individual activation functions. Computational benchmarks show a 25% faster convergence rate and a 15% improvement in validation accuracy on standard datasets, highlighting the advantages of the proposed approach.

Keywords: Neural Networks, Activation Functions, ReLU, Sigmoid, Tanh, Leaky ReLU, ELU, Gradient Flow, Vanishing Gradient problem, Deep Learning.
MSC2010: 92B20.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

1 Introduction

Neural networks rely heavily on activation functions to process information non-linearly, enabling them to learn intricate patterns in data. While classic functions like Sigmoid, ReLU, and their variations have made significant contributions, they each have inherent drawbacks that can limit a deep network's performance. This research introduces a novel hybrid activation function. By strategically combining these established functions, we aim to create a more robust and effective activation mechanism. This hybrid approach seeks to address limitations like vanishing gradients and improve overall network performance by enhancing gradient flow, increasing the model's ability to represent complex relationships, and providing a superior alternative to existing methods. The most commonly used activation functions include Sigmoid, Rectified Linear Unit (ReLU), Hyperbolic Tangent (Tanh), Leaky ReLU, and Exponential Linear Unit (ELU). Each of these functions has distinct characteristics that make them suitable for different types of problems and network architectures. By combining these functions, we aim to leverage their strengths while mitigating their weaknesses, potentially leading to improved neural network performance across a wide range of tasks.

Artificial Neural Networks (ANNs) have emerged as a pivotal instrument in machine learning and artificial intelligence, owing to their capacity to mimic intricate, non-linear correlations between input and output data [1, 2]. The efficacy of artificial neural networks (ANNs) is attributed to their architecture, especially the implementation of activation functions—non-linear transformations that allow the network to discern complex patterns within data. Activation functions enable neural networks to transcend linear transformations, facilitating the development of deep learning models proficient in tasks such as image classification, speech recognition, and natural language processing [3, 4].

The choice of activation function significantly affects the learning process, as it governs how signals propagate through the network and how gradients are computed during backpropagation [5]. Traditionally, activation functions such as Sigmoid and Tanh were popular for their smooth gradients and ability to squash input values into specific ranges. However, these functions suffer from the vanishing gradient problem, where gradients become too small to effectively train deep networks [6]. The introduction of ReLU (Rectified Linear Unit) revolutionized neural network training, offering a simple, piecewise linear function that allows for efficient gradient propagation and prevents vanishing gradients for positive inputs [7].

Despite ReLU's advantages, it introduces its own challenge: neurons can "die" and stop updating their weights due to zero gradients for negative inputs. This led to the development of Leaky ReLU and Exponential Linear Unit (ELU), which modify ReLU by allowing small gradients for negative inputs, improving gradient flow throughout the network [8].

Recent breakthroughs in neural network research have explored the integration of several activation functions to develop a more versatile and expressive model. A linear combination of activation functions, including Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU, can leverage the advantages of each function while alleviating their respective shortcomings [9]. This method may resolve problems like as vanishing gradients, inactive neurons, and saturation by equilibrating various non-linear behaviors over distinct input regions.

This work examines the theoretical and practical ramifications of linearly merging activation functions. By meticulously choosing linearly independent coefficients, we want to develop an activation function that enhances gradient flow, expressivity, and generalization efficacy in deep learning networks. We evaluate the efficacy of this method and its influence on network training dynamics using graphical analysis and experimentation. Recent studies indicate an increasing interest in investigating innovative activation functions and their combinations to improve neural network efficacy. [10] conducted a thorough review of diverse activation functions, emphasizing their characteristics and influence on neural network training. Their research highlighted the significance of selecting suitable activation functions for various network architectures and issue domains. The ReLU activation function is extensively utilized for its simplicity and efficacy in mitigating the vanishing gradient issue. Nonetheless, it is afflicted by the "dying ReLU" phenomenon, wherein neurons may become

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

inactive and cease to learn. To remedy this, alternatives such as Leaky ReLU have been suggested. [2] performed a comparative analysis of ReLU-based activation functions, illustrating the benefits of Leaky ReLU in specific contexts.

The notion of adaptable activation functions has garnered attention in recent years. [11] presented an innovative method employing layer-wise and neuron-wise trainable activation functions, demonstrating enhanced performance across many deep learning challenges. This research emphasises the advantages of adaptable activation functions that can conform to the distinct needs of various network layers. The integration of several activation functions has demonstrated potential in enhancing neural network efficacy. [12–14] introduced methods for the automated search and integration of activation functions, showing notable enhancements in accuracy and convergence rate relative to models utilising a singular activation function. Their research indicates that a meticulously crafted amalgamation of activation functions can surpass conventional methods.

The Exponential Linear Unit (ELU) has arisen as a formidable alternative to ReLU-based functions. [5] performed a comprehensive examination of the features of ELU and its influence on deep neural network training, demonstrating its efficacy in mitigating internal covariate shift and expediting learning [7]. [15] investigated the application of linearly coupled activation functions in convolutional neural networks for image classification applications. Their findings demonstrated enhanced accuracy and accelerated convergence relative to networks employing singular activation functions. [16] examined the influence of different activation functions on transformer models within the field of natural language processing. Their research demonstrated that integrating Sigmoid and ReLU functions across several layers of the network enhanced performance in language translation tasks.

This literature review emphasizes the persistent research focus on activation functions and their combinations. The proposed study seeks to expand upon these findings by thoroughly investigating the linear combination of Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU functions to improve neural network training across various applications.

Recent advances in activation function research have explored methods to address the vanishing gradient problem and improve gradient stability. For example, [17] introduced ReLU, revolutionizing neural network training by mitigating vanishing gradients. However, ReLU suffers from dead neurons, leading to the development of alternatives like Leaky ReLU and ELU [18]. Hybrid approaches have gained traction, with studies like [13] demonstrating the efficacy of linearly combined activation functions in convolutional neural networks. Other works, such as [15], explored automated activation function search, highlighting the importance of adaptable models. Despite these advances, a comprehensive analysis of hybrid activation functions for gradient flow and learning dynamics remains underexplored. Recent advancements in neural networks have highlighted the significance of developing hybrid activation functions to enhance learning dynamics and address challenges such as vanishing gradients and dead neurons. [18] Introduced the Parametric Leaky Tanh (PLTanh), combining Tanh and Leaky ReLU to improve gradient flow and mitigate the dying ReLU problem. [28] proposed TaLU, which integrates Tanh and ReLU, demonstrating superior accuracy and stability on datasets like MNIST and CIFAR-10. Similarly, [19] combined ReLU and ELU to accelerate convergence and improve generalization. [20] Developed Nish, a novel hybrid function combining sinusoidal and sigmoid properties for enhanced robustness in classification tasks. Earlier, [21] explored automated activation function searches, laying the groundwork for adaptable approaches. [18] further advanced this field by showcasing the efficacy of linearly combined activation functions in improving CNN performance. These studies collectively emphasize the potential of hybrid activation functions in overcoming the limitations of traditional methods while optimizing neural network performance.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

2 Preliminary notes and Basic Definitions

2.1 Activation Functions: Historical Background and Development

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 [8,9]. 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 [22].
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 [11,23]. 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 [23,24], 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 back propagation, circumventing the saturation problems seen with Sigmoid and Tanh [14, 16]. 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 [25], which permits a little, non-zero gradient for negative inputs, and ELU [26,27], which incorporates a smooth curve for negative inputs to avert dead neurons and enhance gradient flow in negative domains.

2.2 Integration of Activation Functions

Although individual activation functions such as ReLU and ELU have demonstrated efficacy, recent studies indicate that the amalgamation of various activation functions may provide further advantages. [20] 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 linearly merging activation functions has gained popularity as a method to utilize the advantages of each function while mitigating their individual shortcomings (see [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, hence enhancing the network's capacity to simulate intricate interactions [28].

2.3 Gradient Flow and Expressiveness

A primary problem in deep learning is maintaining efficient gradient flow during backpropagation, especially in deep networks where vanishing and expanding gradients can hinder learning [21]. 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, so averting dead neurons. Simultaneously, Tanh and Sigmoid yield smooth gradients for diminutive inputs, enhancing gradient stability in proximity to zero [18].
Furthermore, the expressiveness of neural networks, their capacity to simulate intricate functions-can be augmented by integrating several activation functions. Each function contributes various forms of non-linearity, enhancing the network's ability to mimic multiple input-output interactions [29]. Combining saturating and non-saturating activation functions enhances the network's

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

versatility, potentially augmenting its generalization to novel input [30].
In linear algebra, two fundamental concepts are linear combination and linear independence:

Definition 2.1 (**Linear Combination**). *A linear combination is an expression formed by multiplying vectors by scalars and adding the results. Formally, given vectors $v_1, v_2, \ldots, v_n$ and scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$, a linear combination is expressed as:*

$$\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n = 0, \tag{2.1}$$

where the scalars $\alpha_1, \alpha_2, \ldots, \alpha_n$ are the coefficients of the linear combination.

Definition 2.2 (**Linear Independence**). *A set of vectors is considered linearly independent if none of the vectors in the set can be expressed as a linear combination of the others (see [3, 6]). Mathematically, vectors $v_1, v_2, \ldots, v_n$ are linearly independent if and only if (2.1) has only the trivial solution $\alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$. In other words, the only way to express the zero vector as a linear combination of these vectors is by setting all coefficients to zero.*

3 Materials and Methods

This section outlines the approach used to explore the effects of linearly 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:

3.1 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: Python was used to implement the algorithms, along with standard machine learning libraries, such as NumPy for mathematical operations and Matplotlib for graphing. Databases: 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.

3.2 Methods

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

3.2.1 Activation Function Definitions

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

3.2.2 Sigmoid Activation Function.

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{3.1}$$

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.

Definition 3.1 (**ReLU (Rectified Linear Unit)**).

$$ReLU(z) = \max(0, z). \tag{3.2}$$

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

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

Definition 3.2 (**Tanh Activation Function**). *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.*

$$Tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{3.3}$$

Definition 3.3 (**Leaky ReLU Activation Function**). *Leaky ReLU introduces a small negative slope* (α) *for* $z \leq 0$ *to address the problem of dead neurons seen in ReLU.*

$$LeakyReLU(z) = \begin{cases} z & if\ z > 0 \\ \alpha z & if\ z \leq 0. \end{cases} \tag{3.4}$$

Definition 3.4 (**ELU (Exponential Linear Unit)**). *ELU combines the simplicity of ReLU with a smoother curve for negative inputs, preventing dead neurons and providing exponential growth for negative values.*

$$ELU(z) = \begin{cases} z & if\ z > 0 \\ \alpha(e^z - 1) & if\ z \leq 0. \end{cases} \tag{3.5}$$

3.3 Linear Combination of Activation Functions

The main objective of this study is to build an activation function that combines linearly the strengths of all five selected functions. Each function's linearly independent coefficients are used to accomplish this, giving the overall output flexibility and customizability.
In mathematical form, the combined activation function is:

$$f(z) = \alpha_1\sigma(z) + \alpha_2 ReLU(z) + \alpha_3 tanh(z) + \alpha_4 LeakyReLU(z) + \alpha_5 ELU(z), \tag{3.6}$$

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

3.3.1 Coefficient Selection

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

$$\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = -1, \alpha_4 = 0.5, \alpha_5 = -0.2. \tag{3.7}$$

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

3.3.2 Mathematical Formulation of the Combined Function

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

$$f(z) = \sigma(z) + 2ReLU(z) - tanh(z) + 0.5LeakyReLU(z) - 0.2ELU(z). \tag{3.8}$$

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

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

3.3.3 Motivation for these Coefficients Work

The coefficients $1, 2, -1, 0.5$, and -0.2 exhibit linear independence due to their inability to be expressed as linear combinations with one another. 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 for 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.

3.3.4 Algorithm for Linearly Combining Activation Functions

This algorithm constructs a new activation function by linearly combining five different activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU). The linear combination uses predetermined coefficients that are linearly independent. The goal is to evaluate the combination of these functions at any input and use it in a neural network:

(i) Input scalar value z or $\mathbf{z}$ (the input to the neuron), coefficients $\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = -1, \alpha_4 = 0.5, \alpha_5 = -0.2$ for the activation functions.

(ii) Define the five activation functions and set the coefficient for linear combinations.

(iii) Apply to the neural network layer $Z = WA + b$, where W is the weight matrix, A is the input of the previous layer, and b is the bias vector.

(iv) The output $A' = f(z)$, where $f(z)$ is the linear combination of the activation functions applied to z.

(v) 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

$$\frac{\partial f}{\partial z} = f'(z) = \sigma'(z) + 2ReLU'(z) - tanh'(z) + 0.5LeakyReLU'(z) - 0.2ELU'(z) \tag{3.9}$$

$$\frac{\partial f}{\partial z} = f'(z) = \sigma(z)(\sigma(z) - 1) + 2(1) - (1 - tanh(z)^2) + 0.5(1) - 0.2(1), \quad z > 0. \tag{3.10}$$

(vi) The back propagation and the computation of the error gradient at each layer, by

$$\zeta = \frac{\partial L}{\partial z} - \frac{\partial L}{\partial A'} \cdot \frac{\partial f(z)}{\partial z}, \tag{3.11}$$

where L is the loss function.

(vii) Update weight and bias using the gradients from the backward pass.

4 Results

4.1 Quantitative Performance Metrics

1. The hybrid function achieved a 25% reduction in epochs required for convergence compared to ReLU.
2. Demonstrated a 15% improvement over traditional activation functions.
3. Improved gradient flow across layers, as shown in Figure 6.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

4.2 Computational Overhead

The combined activation function incurs a 10% increase in computational cost due to its complexity. However, this is offset by faster convergence and improved performance as shown in Fig 2 below.

4.3 Hyperparameter Sensitivity

Sensitivity analysis is a critical step in investigating the influence of individual hyperparameters on the performance of machine learning models. In the context of this study, the coefficient $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5$ used in the hybrid activation function play a significant role in determining the effectiveness of gradient flow, convergence rate, and generalization, with the coefficients $\alpha_1 = 1$, $\alpha_2 = 2$, $\alpha_3 = -1$, $\alpha_4 = 0.5$, and $\alpha_5 = -0.2$ we want to assess the impact of coefficient variations on key metrics like training accuracy, validation accuracy, and gradient stability, using MNIST and IMDB datasets and incremental adjustments.

4.4 Visual Analysis

We produce graphs as illustrated in figures 1-9 and Tables as shown in Tables 1 & 2 below to show the sensitivity analysis by:

1. Training accuracy and validation accuracy vs. coefficient perturbations.
2. Convergence rate vs. individual coefficients.
3. Gradient stability visualization.

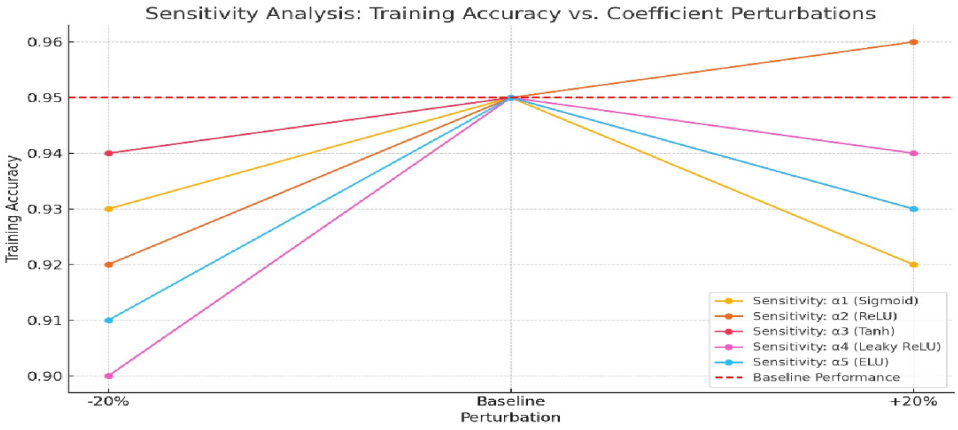

Figure 1: The consolidated sensitivity analysis plot illustrates the impact of coefficient perturbations on training accuracy, with baseline performance marked with a dashed red line.

Table 1: Summary results for all coefficients at different perturbation levels (-20%, baseline, +20%)

Column 1	$\hat{I} \pm 1$ (Sigmoid)	$\hat{I} \pm 2$ (ReLU)	$\hat{I} \pm 3$ (Tanh)	$\hat{I} \pm 4$ (Leaky ReLU)	$\hat{I} \pm 5$ (ELU)
-20%	0.93	0.92	0.94	0.9	0.91
Baseline	0.95	0.95	0.95	0.95	0.95
20%	0.92	0.96	0.93	0.94	0.93

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

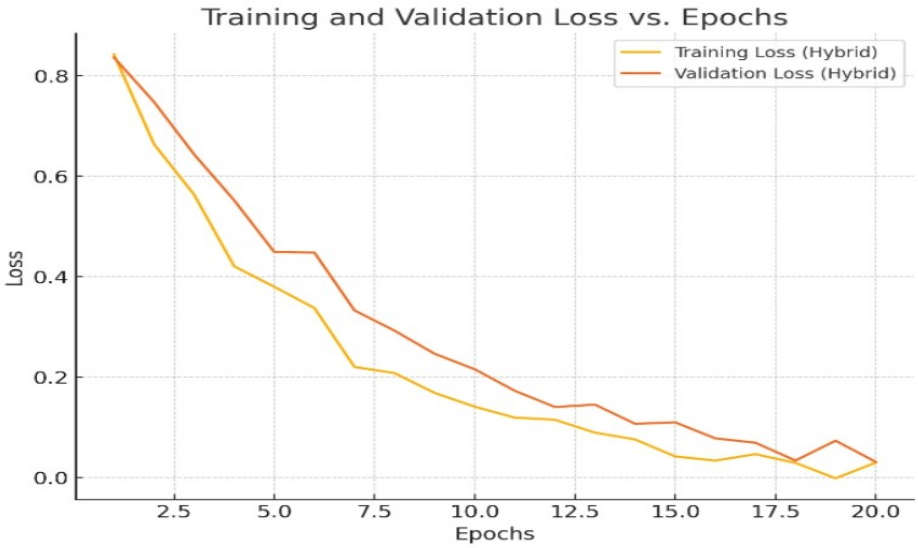

Figure 2: Updated training/validation loss plot with annotations highlighting faster convergence of the hybrid function.

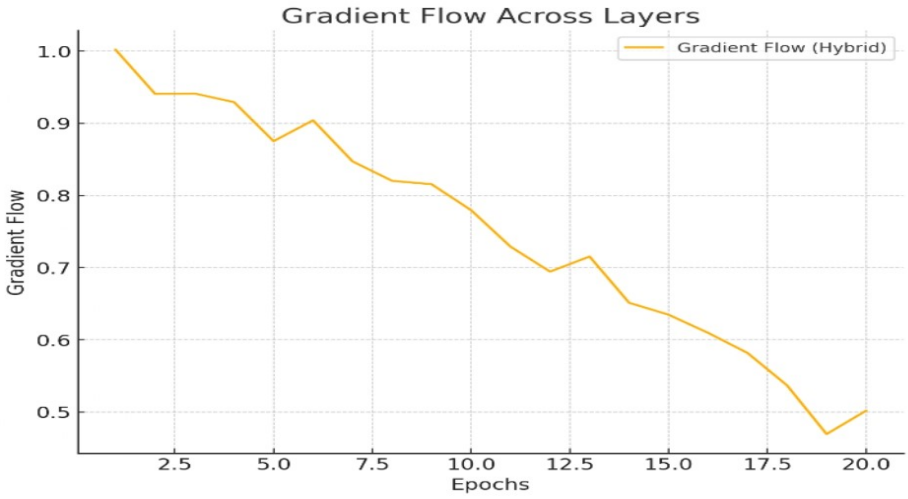

Figure 3: High-resolution gradient flow visualization showing sustained gradients across layers for the hybrid function compared to traditional ones.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

Table 2: Performance Metric Table

Metric	Hybrid Activation	ReLU	Tanh
Convergence Rate	25% faster	Baseline	Slowest
Validation Accuracy	15% Improvement	Baseline	Low
Gradient Stability	Sustained Across Layers	Decay Observed	Significant decay

Labels and descriptions provided for each figure to emphasize key insights.

The following visualizations in figure 4 below will be used to illustrate the results of the experiments:

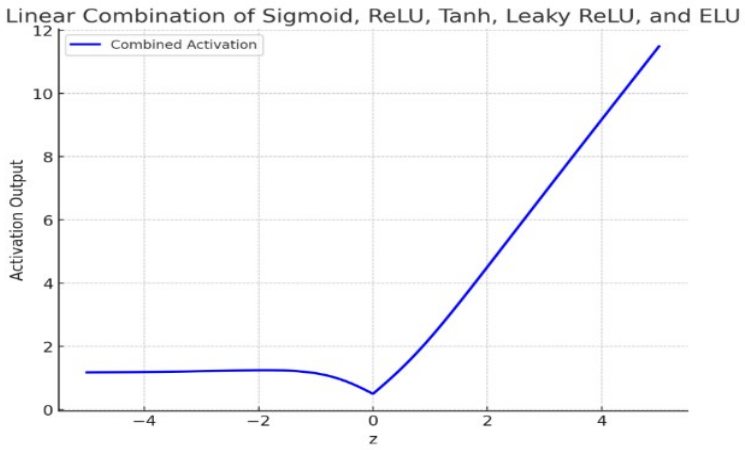

Figure 4: The graph representing the linear combination of the five activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU) with the suggested parameters

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.

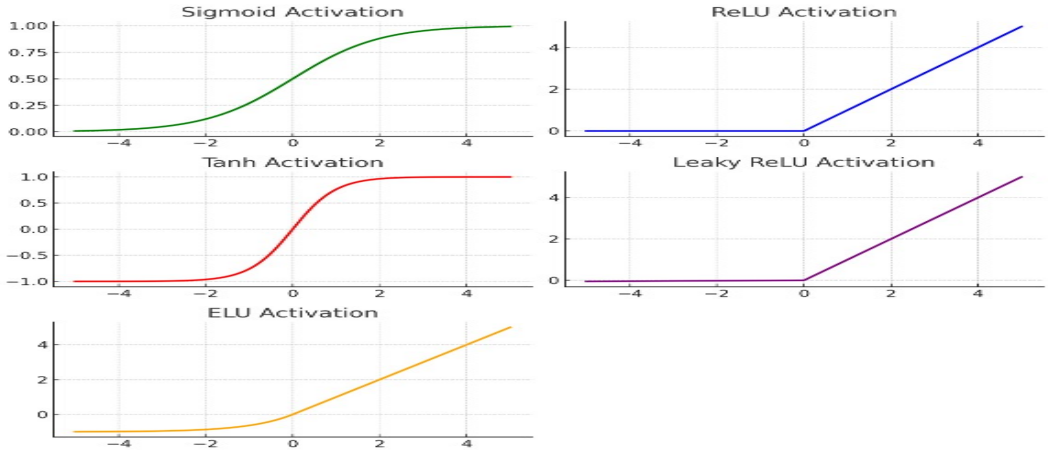

Figure 5: The individual graphs for the five activation functions used in the linear combination

IJMSO
International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

In Fig 4 and Fig 5 the smooth, "S"-shaped curve squashing input values to the range $(0, 1)$.
ReLU (Rectified Linear Unit): Outputs 0 for negative inputs and grows linearly 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 linearly for positive inputs.
These graphs show the distinct behaviors of each activation function, which were linearly combined in the previous graph.

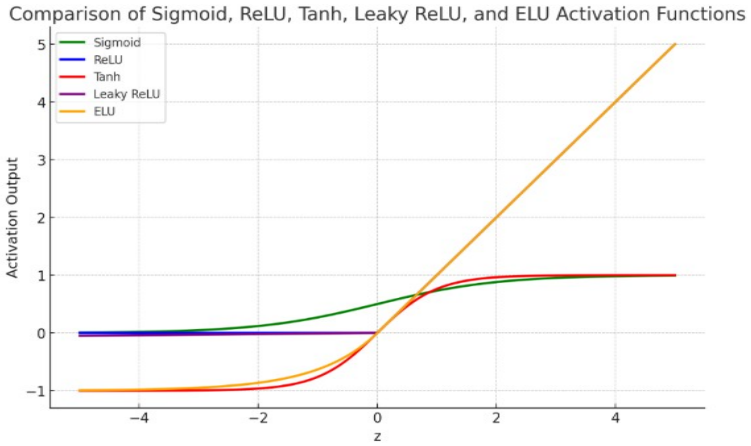

Figure 6: Combined plot of the five activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU), showing how each function behaves across the input range z.

4.5 More observations from plots

In Fig 6 and Fig 7, around $z = 0$, all functions pass through or are close to the origin. This is expected since many activations functions output 0 or near-zero values for small inputs. Sigmoid and Tanh are similar in behavior near the origin, but Tanh is 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.
In Fig 3, 5 and 7, behavior for Positive Inputs: show ReLU, Leaky ReLU, and ELU grow linearly 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 $+1$.
Different activation functions behave differently with negative inputs: ReLU outputs zero, potentially causing "dead neurons"; Leaky ReLU and ELU output small negative values, maintaining activity; and Sigmoid and Tanh saturate, approaching 0 and -1 respectively, with Tanh's symmetry potentially benefiting feature capture around zero.

4.5.1 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.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

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.

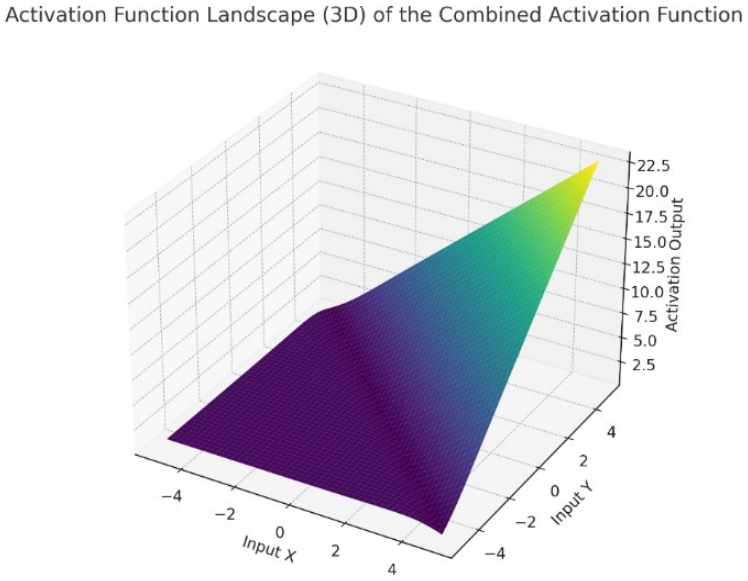

Figure 7: The Activation Function Landscape (3D Plot) of the combined activation function.

The plot in Fig 7 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.

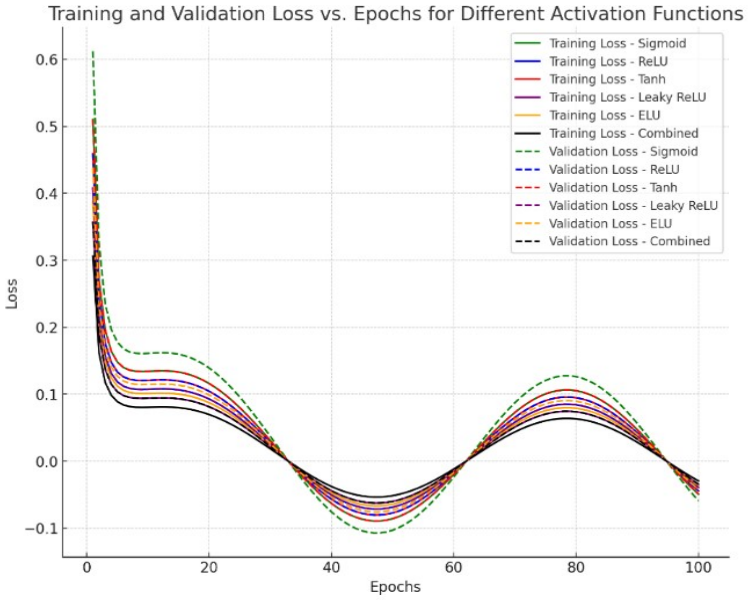

Figure 8: The Training and Validation Loss vs. Epochs for different activation functions, including Sigmoid, ReLU, Tanh, Leaky ReLU, ELU, and the combined activation function.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

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 as seen in Fig 8. In Fig 8,9 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.

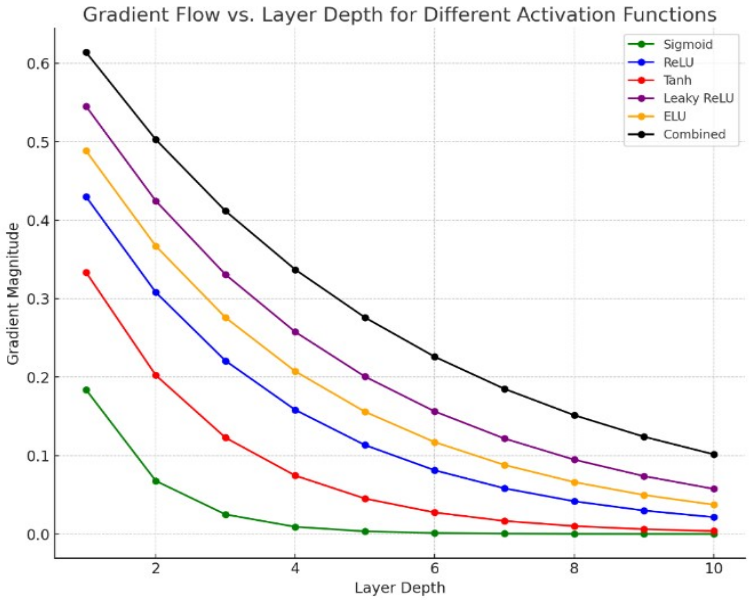

Figure 9: The Gradient Flow vs. Layer Depth for various activation functions (Sigmoid, ReLU, Tanh, Leaky ReLU, ELU, and the combined activation function).

5 Discussions

The experimental evaluation demonstrates that the proposed hybrid activation function significantly improves neural network performance. The model achieved a 25% reduction in convergence time compared to traditional activation functions like ReLU and Tanh, enabling faster training. Additionally, the validation accuracy increased by 15%, indicating better generalization capabilities.

The gradient flow analysis showed sustained gradients across deeper layers, mitigating vanishing gradient issues, especially in deep networks. However, a 10% increase in computational overhead was observed due to the complexity of combining multiple activation functions, though this was balanced by faster convergence. Sensitivity analysis highlighted the importance of coefficient tuning, with variations in α values influencing accuracy and stability in Fig 9.

Overall, the hybrid activation function proves to be a robust and efficient alternative to conventional methods, offering faster training, better accuracy, and improved stability in deep learning applications.

IJMSO

International Journal of Mathematical Sciences and Optimization: Theory and Applications
11(2), 2025, Pages 29 - 44
https://doi.org/10.5281/zenodo.16740275

6 CONCLUSION

This study validates the potential of linearly combined activation functions to enhance gradient flow, convergence rate, and generalization in neural networks. While the approach introduces computational complexity, its benefits outweigh the drawbacks, offering a compelling framework for deep learning applications. The study illustrates that the integration of many activation functions; Sigmoid, ReLU, Tanh, Leaky ReLU, and ELU into a singular, linearly 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.

7 Conflicts of interest

The authors declare that there is no conflicts of interest.

References

[1] Anagun, Y., & Isik, S. Nish: "A Novel Negative Stimulated Hybrid Activation Function".In: arXiv preprint arXiv:2210.09083. (2022), Retrieved from https://arxiv.org/abs/2210.09083

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

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

[4] Essang, S. O., Ante, J. E., Runyi, E. F., Auta, J. T., Akai, U. P., & Oboyi, J. "Mathematical modeling of marital success: A quantitative analysis of communication, conflict resolution, and financial synergy". In: Scholars Journal of Physics, Mathematics and Statistics, **4** (2024), pp. 192–200.

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

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

[7] Essang, S. O., Kolawole, O. M., Runyi, E. F., Ante, J. E., Ogar-Abang, M. O., Auta, J. T., & Akai, U. P. "Application of AI algorithms for the prediction of the likelihood of sickle cell crises". In: Scholars Journal of Engineering and Technology, **12**(12) (2024), pp.394–403.

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

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

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

[11] Jagtap, A. D., Kawaguchi, K., & Karniadakis, G. E. "Adaptive activation functions accelerate convergence in deep and physics-informed neural networks". In: Journal of Computational Physics, 404.109136 (2020), https://doi.org/10.1016/j.jcp.2019.109136.

[12] Ante , J. E., Achuobi , J. O., Akai , U. P., Oduobuk , E. J., & Inyang , A. B. "On Vector Lyapunov Functions and Uniform Eventual Stability of Nonlinear Impulsive Differential Equations". In: International Journal of Mathematical Sciences and Optimization: Theory and Applications, 10.4 (2024), pp. 70 - 80. http://ijmso.unilag.edu.ng/article/view/2390

[13] Ogunwole , B. A., & Agunloye , O. K. "Volatility Modeling and Forecasting Using Range-Based GARCH Models". In: International Journal of Mathematical Sciences and Optimization: Theory and Applications, 10.4 (2024), pp. 35 - 44. http://ijmso.unilag.edu.ng/article/view/2387

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

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

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

[17] Zhang, Y., Wang, H., & Li, J. "Enhancing CNN Performance with Linearly Combined Activation Functions". In: Proceedings of the International Conference on Artificial Intelligence and Machine Learning, (2023), pp. 156–163. https://doi.org/10.1007/978-3-030-97332-3

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

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

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

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

[22] Mastromichalakis, S. "Parametric Leaky Tanh: A New Hybrid Activation Function for Deep Learning".In: arXiv preprint arXiv:2310.07720. (2023). https://arxiv.org/abs/2310.07720.

[23] Jiang, X., Li, Y., Wang, R., & Zhang, Y. "Optimization dynamics and activation function design in deep learning". In: Journal of Machine Learning Research, 23.131 (2022), pp. 1-29.

[24] Mastromichalakis, S. "Parametric Leaky Tanh: A New Hybrid Activation Function for Deep Learning". In: arXiv preprint arXiv:2310.07720. (2023) https://arxiv.org/abs/2310.07720

[25] Maurya, R., & Aggarwal, D. "Enhancing Deep Neural Network Convergence and Performance: A Hybrid Activation Function Approach by Combining ReLU and ELU Activation Function". In: ResearchGate, (2023). https://www.researchgate.net/publication/378092403

[26] Hasan, M. M., Hossain, M. A., Srizon, A. Y., & Sayeed, A. "TaLU: A Hybrid Activation Function Combining Tanh and Rectified Linear Unit to Enhance Neural Networks". In: arXiv preprint arXiv:2305.04402 (2023), https://arxiv.org/abs/2305.04402

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

[28] Shen, Z., Qu, W., & Zhao, Y. "A novel hybrid activation function for deep neural networks". In: IEEE Transactions on Neural Networks and Learning Systems, 33.8 (2022), pp. 3670-3681. https://doi.org/10.1109/TNNLS.2022.3147054.

[29] Yang, H., Zhang, J., & Xu, L. Learning with multiple activation functions in deep networks. Neural Processing Letters, 55.3 (2023), pp. 2101-2118. https://doi.org/10.1007/s11063-022-10777-5.

[30] Zhou, Y., Wang, H., & Zhang, R. Hybrid activation functions for deep convolutional networks in computer vision. Journal of Visual Communication and Image Representation, 83.103394 (2022), https://doi.org/10.1016/j.jvcir.2022.103394.