Punjab University Journal of Mathematics (2025), 57(05), 591-611
https://doi.org/10.52280/pujm.2025.57(05)06

Adaptive-Tanh ReLU (AHReLU): A Novel Activation Function for Enhanced Convolutional Neural Network Performance

*Abaid Ullah
Department of Mathematics, Government College University, Faisalabad, Pakistan

M. Imran
Department of Mathematics, Government College University, Faisalabad, Pakistan

Saima Akram
Department of Mathematics, Government College Women University, Faisalabad, Pakistan

*Corresponding Author
abaidullah@gcuf.edu.pk

Received
10 February, 2025

Accepted
02 October, 2025

Published Online
13 November, 2025

Abstract. In this work, we propose a novel activation function, AHReLU, which demonstrates superior performance compared to traditional activation functions such as ReLU, GELU, and Mish. Replacing ReLU with AHReLU in the VGG-16 model improved Top-1 classification accuracy by $0.79\%, reaching 97.57\%$. In the CIFAR-100 dataset, AHReLU outperformed ReLU by 0.32%, achieving a Top-1 accuracy of 59.82%. For the SVHN dataset, AHReLU achieved a mean accuracy of 95.38%, slightly surpassing ReLU's performance of 95.36%. In machine translation tasks, specifically on the WMT 2014 dataset, AHReLU achieved a BLEU score of 27.5, which is 0.2 points higher than ReLU and 1.2 points higher than GELU. These results highlight that AHReLU, with its learnable parameters, outperforms traditional activation functions, leading to better model performance across various datasets and tasks. The introduction of learnable parameters into the activation function is key to the observed improvements, making AHReLU a promising candidate to replace widely used activation functions such as ReLU, GELU, and Mish in deep learning models.

AMS (MOS) Subject Classification Code: 76A05
Key Words: Activation function, Deep learning, Loss optimization, Computational effi-

ciency.

1. Introduction

Artificial neural networks (ANNs) are a type of computer model that mimics the functioning of neurons in the human brain. These models form the basis for deep learning, a vital component of modern artificial intelligence technologies. ANNs are made up of layers, each layer containing several neurons. These neurons take an input, perform some mathematical operations on it, such as an affine linear map, and pass the result to the next neuron. The activation function, a nonlinear function applied to each neuron, determines the output, acting like an ”on” or ”off” switch. Activation functions are a critical component of artificial neural networks. When the network is trained, the linear map is optimized to make the model's predictions more accurate. However, the activation function is fixed before training and is not changed. Recently, researchers have been working on developing new activation functions. Their focus is on understanding which mathematical properties make an activation function effective in enhancing learning performance[25].

A good activation function is one that works effectively on all types of data and completes training quickly and easily, improving the neural network's performance and accuracy. In the early days of deep learning, people used to build simple, small layer networks, during which activation functions like hyperbolic tangent and sigmoid were commonly used. As research progressed and people started building larger and more complex networks, ReLU[23], [6] (Rectified Linear Unit) became the most popular activation function. The biggest feature of ReLU is that it uses the positive part of the input. With ReLU, deep neural networks solve very complex and advanced problems while achieving higher accuracy. The ReLU activation function has some limitations, such as its output not being around zero, its tendency to ignore negative values, and its unbounded output[42]. To address these limitations and improve accuracy, new activation functions have been developed. The majority of these new functions are enhancements or variations of the original ReLU, such as: Leaky ReLU [20], which slightly processes negative values, ELU [3] (Exponential Linear Unit), which smoothens the output, PReLU [7] (Parametric ReLU), which is customizable and uses hyperparameters, RReLU [38](Randomized ReLU), which adds randomness, ISRLU [1](Inverse Square Root Linear Unit), designed for faster training,FReLU [26](Flexible ReLU), which offers more flexibility and adjustability. Nowadays, hyperbolic and sigmoid-based activation functions also show good performance, such as Swish, which has become a strong alternative to ReLU. The concept of hyperparameters is that by setting specific variables within an activation function, its performance can be significantly improved. Essentially, each function has a special parameter that makes it even more efficient.

In this article, a new activation function called AHReLU has been introduced. This function adjusts its parameter α during training using the backpropagation technique. This feature makes the function more flexible and powerful. After testing, it was found that AHReLU outperforms popular activation functions like ReLU and others. This function prevents gradient vanishing and achieves good accuracy even on complex datasets. Table 1 highlights the differences between AHReLU and older activation functions.

TABLE 1. Comparison of the proposed activation functions with previously established widely used activation functions.

Activation Function	Smooth	Zero-Centered	Negative Output Behavior	Trainable	Continuous	Non-Zero Negative	Non-Monotonic
ReLU	No	Yes	Zero outcome and bounded	No	Yes	No	No
Swish	Yes	Yes	Negative outcome and bounded	Yes	Yes	Yes	Yes
Leaky ReLU	No	Yes	Negative outcome and bounded	No	Yes	Yes	No
ELU	Yes	Yes	Negative outcome and bounded	No	Yes	Yes	No
Softplus	Yes	No	Positive outcome and bounded	No	Yes	No	No
Mish	Yes	Yes	Negative outcome and bounded	No	Yes	Yes	Yes
GELU	Yes	Yes	Negative outcome and bounded	No	Yes	Yes	Yes
AHReLU	Yes	Yes	Negative outcome and bounded	Yes	Yes	Yes	No

1.1. **MOTIVATION.** Motivation is very important when we look towards creating a new activation function, and the idea of designing AHReLU came to us from specific challenges and observations. We often felt that popular activation functions like ReLU, Swish, and Mish have certain practical limitations. ReLU sets negative values to zero, which hinders gradient flow and reduces neuron utilization. Swish and Mish are powerful but have slightly higher computational costs, and the control over bounded negative values is not manual. The integration of trainable parameters was missing, which could make activations context-specific.

During our experiments, we realized that an activation function that provides smooth bounded negative values and is trainable could perform better on diverse datasets and problems. Tanh and ReLU both offer very different and useful properties. We felt that if these two properties could be smartly fused using the smoothness of tanh functions and the linear behavior of ReLU a balanced and effective activation function could be designed. Another idea was to make it adaptively adjustable to different datasets and model architectures. For this reason, we introduced α as a trainable parameter, which optimizes itself for the model. When we tested this concept, the results seemed very promising. On problems like image classification, it performed better than ReLU and Swish, and we felt that this idea was worth exploring further.

2. RELATED WORKS

An activation function that can enhance the performance of neural network models is a significant area of research. Identifying the optimal activation function is often a difficult task. In the early days of neural networks, functions like Tanh and Sigmoid were used, which were simple and basic but had certain limitations. Therefore, new functions were developed. In 2010, ReLU [23] was introduced, which provides the direct value for positive inputs and zero for negative inputs. It was simple and effective, which made it the most popular. However, it had an issue known as the ”dying ReLU” problem [20], meaning neurons become inactive for negative values, and the gradient flow stops. Researchers developed new versions to improve ReLU, such as Leaky ReLU [20], which gives a small linear output for negative values, and improved it further by making it trainable in PReLU [7], allowing it to adjust during network training. Subsequently, new functions like ELU [3], RReLU [38], ISReLU [1], FReLU [26], PELU [33], SiLU [5], GELU [10] and Swish [27] were developed, which perform better for different problems. Mish [21] was recently introduced and has shown significant improvement compared to Swish and ReLU. The majority of these activation functions mentioned earlier, such as PReLU, PELU, and FReLU, are non-trainable. A trainable activation function relies on trainable parameters that are tuned during backpropagation. In the 1990s and early 2000s, during Pre-ReLU era, some new trainable activation functions were developed, such as Adjustable Generalized Hyperbolic Tangent [2], Sigmoid, Sector [29] etc. However, later, functions like Leaky ReLU, ELU, and ReLU were modified into PReLU, PELU, and FReLU, making them trainable. Recently, in 2017, Swish [27] was discovered during exhaustive research as an activation function. It gained popularity in the deep learning community for its simplicity and efficiency after being explored through reinforcement learning techniques.

3. PROPOSED ACTIVATION FUNCTION AND THEIR PROPERTIES

A standard process in artificial neural network training involves adjusting the weights of the network’s linear part. However, designing activation functions tailored to the problem at hand has its own advantages. Real world datasets are frequently noisy or complex, making it challenging to design an optimal activation function that can generalize effectively. Anticipating whether an activation function will perform well on such datasets is often unpredictable. While designing custom activation functions for specific problems can be beneficial, identifying activation functions that can often generalize on real world datasets simplifies implementation. For this reason, we focus on an activation function AHReLU, establishing its generalizability and effectiveness compared to convolutional activation functions. We discuss the properties of such activation functions, experiments on complex models, and comparisons with some well known and widely used activation functions. AHReLU defined as

$$f(x) \quad = \begin{cases} x, & x \geq 0, \\ \frac{\alpha \tanh(x)}{1+x^2}, & x < 0, \end{cases} \tag{3. 1}$$

$$f'(x) \quad = \begin{cases} 1, & x \geq 0, \\ \frac{\alpha\left((1-\tanh^2(x))(1+x^2)-\tanh(x)(2x)\right)}{(1+x^2)^2}, & x < 0. \end{cases} \tag{3. 2}$$

3.1. **Comparison with Parametric Variants.** Several parametric activations, such as PReLU [7], PELU [27], and FReLU, extend ReLU by introducing learnable parameters in the negative region. For example,

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

$$\text{PELU:} \qquad f(x) = \begin{cases} \frac{a}{b}x, & x \geq 0, \\ a\left(\exp\left(\frac{x}{b}\right) - 1\right), & x < 0. \end{cases} \tag{3. 4}$$

These functions apply the learned parameter as a linear or simple exponential scaling of negative inputs. In contrast, AHReLU (Eq. (3. 1)) applies the parameter α to a nonlinear bounded mapping $\frac{\tanh(x)}{1+x^2}$. This results in the following.

- Smooth and bounded negative outputs, unlike the unbounded linear tail of PReLU.
- Non-constant derivatives (Eq. (3. 2)), providing richer gradient information for optimization.
- A nested structure, where $\alpha = 0$ recovers the standard ReLU, but non zero α yields a qualitatively different negative response rather than a simple slope adjustment.

Figure 1 shows the graph of the AHReLU activation function for different values of alpha, while Figure 2 shows the graph of the first derivative of AHReLU for various values of alpha. Figures 3 and 4 illustrate the comparison of AHReLU with different activation functions and the comparison of their derivatives, respectively.

An important distinction is that the derivative of AHReLU in the negative region remains bounded and continuously varying, since $|\tanh(x)| \leq 1$ and the denominator $(1 + x^2)^2$ increases with $|x|$. This property avoids abrupt derivative jumps and ensures gradient stability. In contrast, parametric variants such as PReLU have constant derivatives for $x < 0$, which can limit the expressivity. This difference shows that AHReLU's improvements arise not only from having a trainable parameter, but also from the nonlinear fusion of $\tanh$ and ReLU. Therefore, AHReLU nests ReLU as a limiting case when $\alpha = 0$, while preserving a fundamentally distinct nonlinear behavior for $\alpha \neq 0$. This makes it structurally different from parametric ReLU variants, which modify only the slope of the negative branch.

AHReLU becomes the ReLU function for $\alpha = 0$. For example,

$$\lim_{\alpha \to 0} f(x; \alpha) = \text{ReLU}(x), \quad \forall x \in \mathbb{R} \tag{3. 5}$$

The neural networks employing AHReLU activation functions are dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$, and $C(K)$ denotes the space of all continuous functions in K (refer to [22]). This result is established based on the following proposition, as the proposed activation function is non-polynomial. **Proposition (Theorem 1.1 in Kidger and Lyons, 2019** [13]**):** Let $\rho : \mathbb{R} \to \mathbb{R}$ be a continuous function. Define N_ρ^n as the class of neural networks with ρ as the activation function, n neurons in the input layer, one neuron

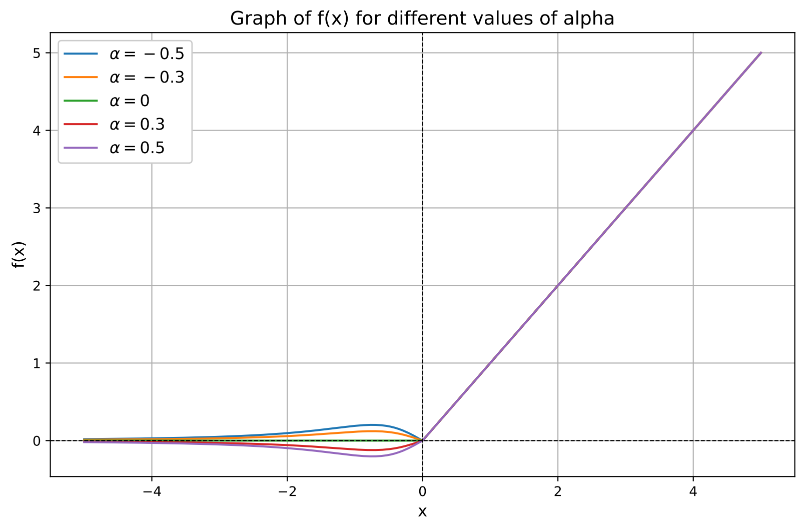

FIGURE 1. Plots of AHReLU for different values of α

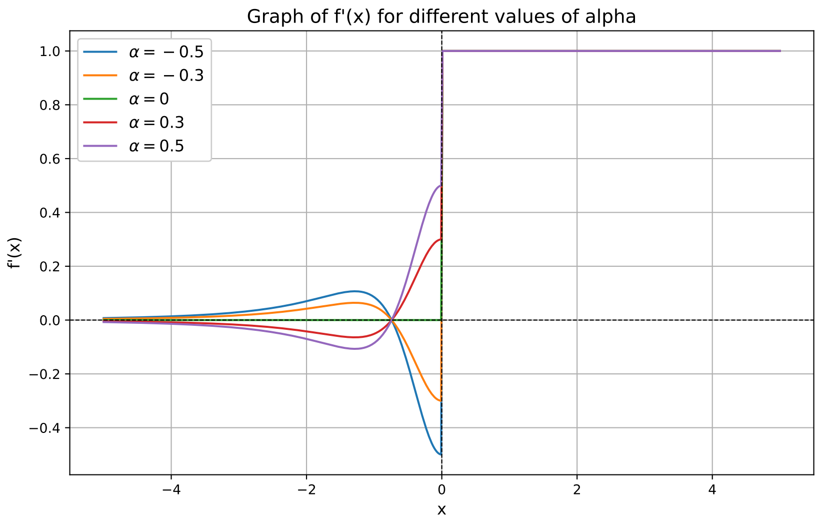

FIGURE 2. Plots of first derivative AHReLU for different values of α

in the output layer and a hidden layer containing an arbitrary number of neurons. For any compact subset $K \subset \mathbb{R}^n$, N_ρ^n is dense in $C(K)$ if and only if ρ is non-polynomial.

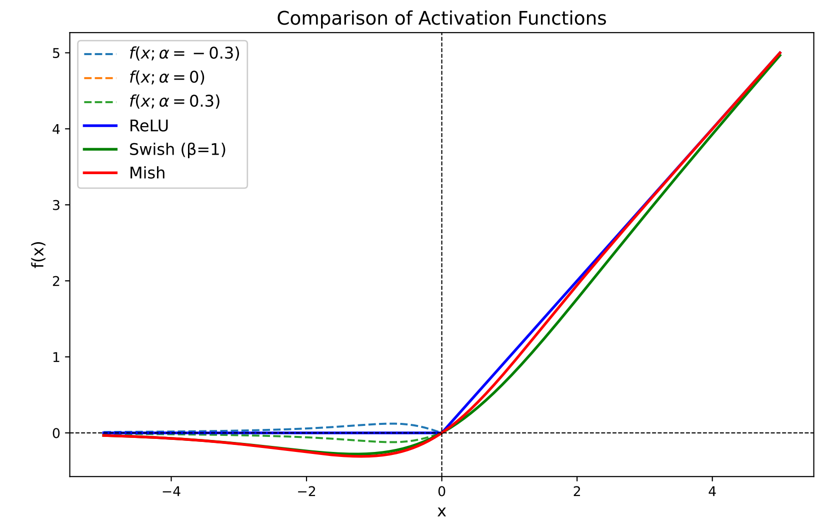

FIGURE 3. Plots of ReLU, Swish, Mish and AHReLU for different values of α

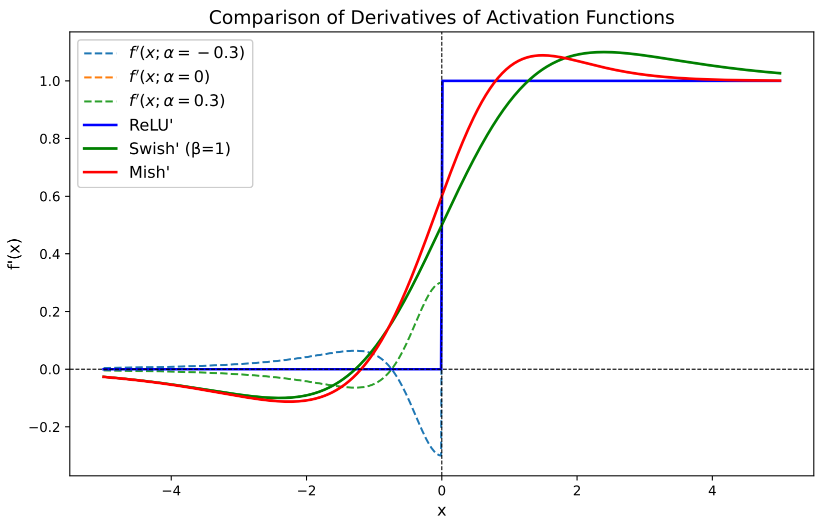

FIGURE 4. Plots of first derivative of ReLU, Swish, Mish and AHReLU for different values of α

4. RESULTS AND DISSCUSSION

In this work, we used the backpropagation algorithm [17] to set the initial values of the hyperparameters for our AHReLU function and then update them. For a single layer, the hyperparameter gradient α is given as:

$$\frac{\partial E}{\partial \alpha} = \sum \frac{\partial E}{\partial f(x)} \cdot \frac{\partial f(x)}{\partial \alpha} \tag{4. 6}$$

Here E is the objective function,α is the hyperparameter defined in the activation function , $f(x)$ is our activation function. We evaluated the performance of activation function using different models and datasets and compared them with seven popular activation functions. A brief explanation of these baseline functions is provided in the following.

- **The Rectified Linear Unit (ReLU)** activation function was introduced by Nair and Hinton [23], and Hahnloser et al. [6]. It is a very popular and widely used activation function due to its simplicity and computational efficiency. ReLU faces a challenge referred to as the "Dying ReLU" problem, where neurons can stop functioning during training as a result of gradients becoming zero.

  ReLU is mathematically defined as:

$$f(x) = \max(0, x) \tag{4. 7}$$

  Here, $f(x)$ is the output of the activation function and x is the input.
- **Leaky ReLU** activation function was introduced by Mass *et al.* [20] in 2013. It is a modified version of ReLU that addresses the "Dying ReLU" problem. In Leaky ReLU, instead of assigning zero for negative inputs, a small slope is given, which helps propagate the gradient. Mathematically, Leaky ReLU is represented as:

$$f(x) = \max(\alpha x, x) \tag{4. 8}$$

  Here, α is a small positive constant, usually set to 0.01, and x is the input.
- **The Exponential Linear Unit (ELU)** activation function was introduced by Clevert *et al.* in 2015 [3]. The ELU activation function uses an exponential curve for negative values, which brings smoothness and efficiency to model training.

  Mathematically, ELU is defined as:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(\exp(x) - 1), & \text{if } x \leq 0 \end{cases} \tag{4. 9}$$

  Here, α is a hyperparameter.
- **Swish** activation function was introduced by Ramachandran *et al.* [27] in 2017. It is continuous and differentiable, making the model's learning more effective. Swish is mathematically defined as:

$$f(x) = x \cdot \sigma(x) \tag{4. 10}$$

  where $\sigma(x)$ is the sigmoid function, given by:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4. 11}$$

- **SoftPlus** activation function was introduced by Zheng *et al.* [41], Dugas *et al.* [4]. It is a smooth version of ReLU, which generates a non zero gradient even for negative inputs. It is mathematically defined as:

$$f(x) = \ln(1 + e^x) \tag{4. 12}$$

- **Gaussian Error Linear Unit (GELU)** activation function was introduced by Hendrycks and Gimpel [10] in 2016. It is defined as:

$$f(x) = 0.5x \left(1 + \tanh\left(\frac{\sqrt{2}}{\pi}\left(x + 0.044715x^3\right)\right)\right) \tag{4. 13}$$

- **Mish** activation function was introduced by Diganta Mishra [21] in 2019. It combines the best properties of ReLU and Swish and is known for its performance in deep learning tasks. Mish is mathematically defined as:

$$f(x) = x \cdot \tanh\left(\log\left(1 + \exp(x)\right)\right) \tag{4. 14}$$

In subsequent sections, the proposed activation function is compared with several other activation functions, including ReLU, ELU, Leaky ReLU, Softplus, Swish, GELU, and Mish, across various deep learning tasks such as image classification and machine translation. For the AHReLU activation function, we set the value of α to 0.5, as mentioned in [7], and updated this hyperparameter value during training using backpropagation, in equation 4. 6 . In the following sections, we outline the experimental setup, framework, and results, all performed on an NVIDIA A100 GPU featuring 40GB of VRAM.

4.1. **IMAGE CLASSIFICATION.** We have used some well-known and standard datasets to solve the image classification problem, such as MNIST [18], Fashion MNIST [36], SVHN [24], CIFAR-10 [15], CIFAR-100 [15] and Tiny imagenet datset [16]. We have provided our results on these datasets.

4.1.1. *MNIST.* The MNIST [18] dataset contains images of hand written digits from zero to nine. This data set has 60,000 training images and 10,000 testing images, all of which are 28x28 grayscale images. A custom CNN (Convolutional Neural Network) [34] with a single layer has been created, where 3x3 kernels are used in the CNN layers and 2x2 kernels are used in the pooling layers. The channel depths are set to 128 (twice), 64 (thrice), 32 (twice),a dense layer of size 128, Max-pooling layer(thrice), batch normalization [12] on the custom CNN architecture, and dropout [30] techniques are used, while data augmentation was not applied. The results obtained are shown in Table 2. The confusion matrix heatmap for the MNIST dataset, when using the AHReLU activation function, provides valuable insights into the model's performance across all digit classes (0-9). The heatmap visually highlights the accuracy of the model in classifying each digit, and the diagonal elements represent the correct predictions for each class shown on the heat map as shown in Figure 5.

4.1.2. *FASHION MNIST.* Fashion MNIST [36] is a well-known dataset in computer vision, containing 28x28 pixel grayscale images of fashion products, categorized into 10 distinct classes. It comprises 60,000 images for training and 10,000 images for testing. Compared to MNIST, Fashion MNIST offers a more complex classification challenge. No data augmentation was applied to the dataset. The same CNN architecture, previously used

TABLE 2. Results of Experiments on the MNIST Dataset

Activation Function	**Mean accuracy (%) on MNIST dataset**
Mish	99.49
ReLU	99.49
GELU	99.47
Leaky ReLU	99.44
Softplus	99.52
Swish	99.49
ELU	99.48
AHReLU	**99.53**

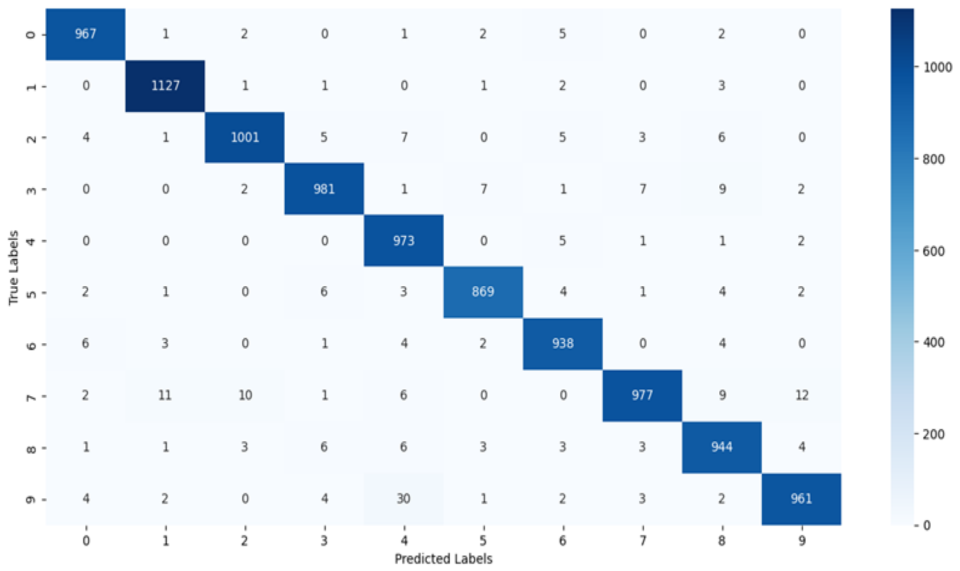

FIGURE 5. The heat map of confusion matrix in case of MNIST dataset.

for the MNIST dataset, was utilized for both training and testing on Fashion MNIST, and the results are presented in Table 3.

4.1.3. *SVHN (STREET VIEW HOUSE NUMBERS).* SVHN [36] is a widely used image dataset derived from house numbers captured in Google Street View imagery. It consists of RGB images with a resolution of 32x32 pixels. The database includes 73,200 training images, 26,032 testing images, and 10 different categories. For this database, we used the same CNN architecture that was used for the MNIST data set. In this experiment, data augmentation techniques were also applied. The results obtained are shown in Table 4.

4.1.4. *Tiny ImageNet Dataset.* The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) has long been recognized as one of the most influential benchmarks for image

TABLE 3. Results of Experiments on the Fashion MNIST Dataset

Activation Function	**Mean accuracy ($\%$) on Fashion MNIST dataset**
ReLU	93.40
Swish	93.35
Leaky ReLU	92.75
ELU	92.81
Softplus	92.97
Mish	92.68
GELU	93.42
AHReLU	**93.52**

TABLE 4. Results of Experiments on the SVHN Dataset

Activation Function	**Mean accuracy ($\%$) on SVHN dataset**
ReLU	95.36
ELU	94.71
Swish	95.34
Softplus	95.02
Leaky ReLU	95.16
Mish	94.14
GELU	94.96
AHReLU	**95.38**

classification. The Tiny ImageNet Challenge provides a scaled down variant of ILSVRC, making it particularly suitable for rapid experimentation and benchmarking of novel approaches. The dataset [16] consists of images of size 64×64 in 200 object categories, comprising a total of 120,000 images. It is divided into 100,000 training images, 10,000 validation images, and 10,000 test images, with each class containing 500 training, 50 validation and 50 test samples. To assess the effectiveness of the proposed AHReLU activation function, we compared it against several widely adopted activation functions using the WideResNet-28-10 (WRN-28-10) [40] architecture. The network was initialized using the Normal initializer [7] and trained with a batch size of 32, the Adam optimizer [14], a dropout rate of 0.2 [30], and an initial learning rate of 0.01. The learning rate was reduced by a factor of 10 after every 50 epochs, with training conducted for a total of 250 epochs. To enhance generalization, standard data augmentation techniques were employed during training. The top-1 classification accuracy (averaged over 5 independent runs) is summarized in Table 5. The results demonstrate that the proposed AHReLU achieves the highest accuracy of 62.26%, outperforming other popular activation functions.

4.1.5. *CIFAR.* CIFAR [15] is a widely recognized dataset in computer vision, comprising colored images with dimensions of 32x32 pixels. This data set has a total of 70,000 images, with 60,000 for training and 10,000 for testing. There are two types of CIFAR: CIFAR-10 and CIFAR-100. CIFAR-10 contains 10 classes, each with 6,000 images, while CIFAR-100 has 100 classes, each containing 600 images. We have reported the results of various

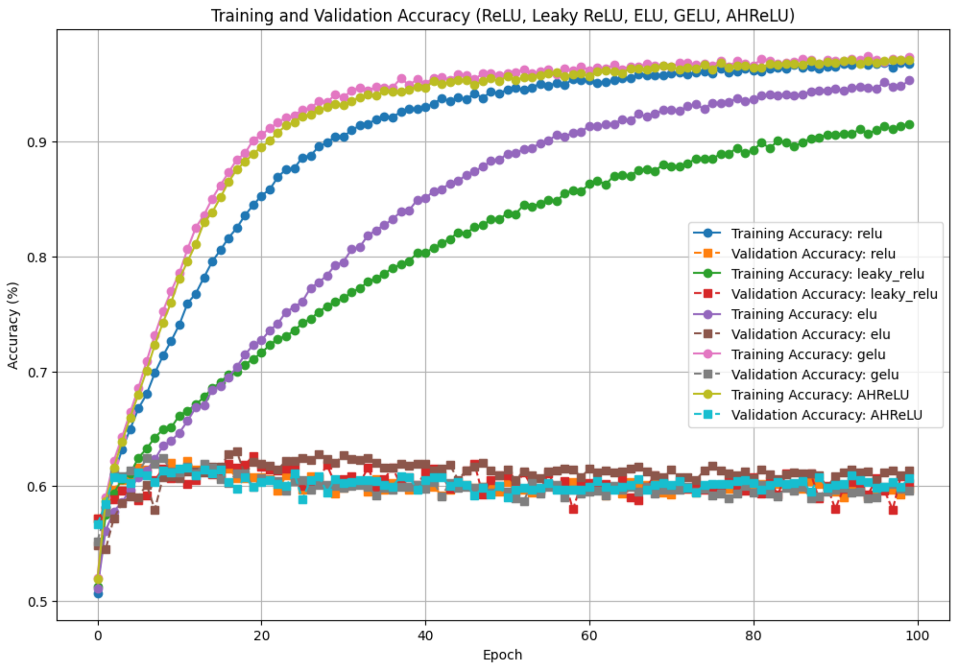

FIGURE 6. Training and test Accuracy (Higher is Better) on CIFAR100 Dataset Using VGG16 Model for ReLU, Leaky ReLU, ELU, GELU and AHReLU

TABLE 5. Top-1 classification accuracy (%) of different activation functions on Tiny-ImageNet using WRN-28-10 (mean of 5 runs).

Activation Function	**Accuracy (%)**
ReLU	60.85
Leaky ReLU	60.88
ELU	60.51
Softplus	60.01
GELU	61.38
Swish	61.23
Mish	61.83
AHReLU (Proposed)	**62.26**

models on this dataset, such as (PA-ResNet-34) [8], VGG-16 (with Batch-normalization) [28], Densenet-121 (DN-121) [11], WideResNet 28-10 (WRN 28-10) [40], ResNeXt-50 [37], Deep Layer Aggregation (DLA) [39], Resnet in Resnet (RIR) [32], EfficientNet B0

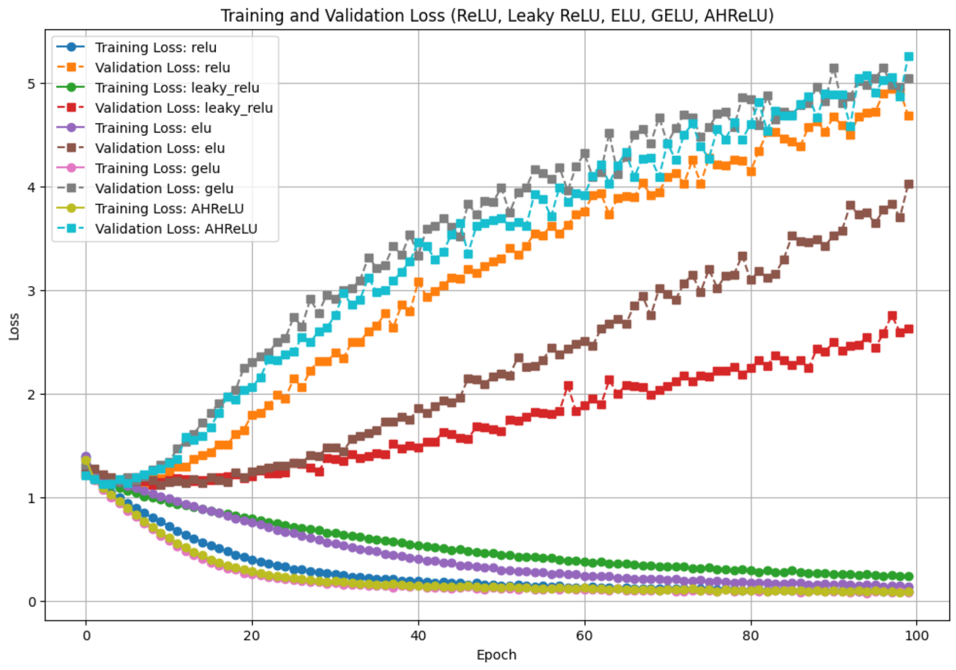

FIGURE 7. Training and test Loss (Lower is Better) on CIFAR100 Dataset Using VGG16 Model for ReLU, Leaky ReLU, ELU, GELU and AHReLU

(EN-B0) [31], Le-Net [19] and ResNet-34 [9] models. The results can be found in Tables 6 and 9. These results clearly demonstrate that the AHReLU activation function outperformed ReLU and Swish in most cases. AHReLU improved accuracy by upto 5% over ReLU across tested models. Each model was trained with a batch size of 128 and the Adam optimizer with a learning rate of 0.001 for up to 100 epochs. Data augmentation was also used for both datasets. The learning curves shown in Figures 6 to 9 for the VGG16 and LeNet models indicate faster convergence, higher accuracy, and lower loss, outperforming other models.

4.2. **MACHINE TRANSLATION.** Machine translation refers to the automatic translation of text from one language to another. For this problem, we used the WMT 2014 English-to-German translation dataset, which contains 4.5 million sentences. We evaluated the performance of the model using the BLEU score. A transformer model with eight heads was employed, which operates in a specific way known as an attention-based multi-head model [35]. In this model, a dropout rate [30] of 0.1 was used, which means that some information is deliberately omitted to improve model generalization. The Adam optimizer [14] was utilized and the model was trained to a satisfactory level of performance. Finally, the model was run five different times and its average performance was reported.

TABLE 6. Performance Evaluation on CIFAR10 Dataset: Top-1 Accuracy (%) Averaged Over 5 Independent Runs

Activation Function	**DN-121**	**RESNET-34**	**PA-RESNET-34**	**VGG-16**	**WRN 28-10**
ReLU	96.78	90.31	90.12	96.78	91.77
Swish	95.81	90.75	90.81	97.16	92.18
Leaky ReLU	95.91	91.06	90.91	91.52	91.32
ELU	95.90	90.16	90.78	95.30	91.02
Softplus	95.99	98.37	90.41	97.83	91.01
Mish	96.79	90.87	91.72	97.06	92.98
GELU	96.62	90.78	90.97	97.36	92.31
AHReLU	**96.98**	**92.10**	**92.21**	**97.57**	**94.02**

TABLE 7. Performance Evaluation on CIFAR10 Dataset: Top-1 Accuracy (%) Averaged Over 5 Independent Runs

Activation Function	**Le-Net**	**EN-B0**	**RIR**	**DLA**	**ResNeXt-50**
ReLU	97.01	87.36	90.91	90.71	92.52
Swish	97.59	86.82	90.65	89.23	91.92
Leaky ReLU	94.72	86.17	90.73	90.47	91.87
ELU	96.74	86.12	91.03	90.46	91.72
Softplus	95.85	85.81	90.99	91.26	92.32
Mish	97.69	86.71	91.99	91.86	92.04
GELU	97.83	86.91	91.13	90.51	91.90
AHReLU	**97.52**	**88.01**	**92.78**	**92.09**	**93.97**

TABLE 8. Performance Evaluation on CIFAR100 Dataset: Top-1 Accuracy (%) Averaged Over 5 Independent Runs

Activation Function	**RESNET-34**	**WRN 28-10**	**DN-121**	**PA-RESNET-34**	**VGG-16**
ReLU	63.58	67.98	67.83	61.99	59.50
Swish	63.87	68.13	68.39	63.74	59.67
Leaky ReLU	63.69	68.09	67.11	63.14	59.53
ELU	63.21	67.54	67.89	62.87	60.68
Softplus	64.36	66.71	67.07	62.81	59.96
Mish	64.87	69.02	69.01	64.12	59.72
GELU	64.91	68.89	67.79	63.71	58.71
AHReLU	**65.14**	**69.59**	**69.38**	**64.47**	**59.82**

4.3. **COMPUTATIONAL TIME COMPARISON.** We have reported the computational time comparison between the AHReLU activation function and the baseline activation functions, specifically for forward and backward passes on a 32×32 RGB image using the ResNet-50 model. Performance is evaluated for the mean of 100 runs, all of which are conducted on a NVIDIA A100 GPU with 40GB of VRAM. The computational times

TABLE 9. Performance Evaluation on CIFAR100 Dataset: Top-1 Accuracy (%) Averaged Over 5 Independent Runs

Activation Function	Le-Net	EN-B0	RIR	DLA	ResNeXt-50
ReLU	58.59	52.91	63.02	63.67	70.20
Swish	54.40	53.38	63.95	64.08	70.71
Leaky ReLU	60.10	52.94	63.21	63.95	69.78
ELU	54.17	53.70	63.84	63.15	69.23
Softplus	49.50	52.78	62.61	62.78	69.41
Mish	55.94	53.38	63.72	63.36	69.89
GELU	55.96	53.79	63.52	63.91	70.52
AHReLU	**60.69**	**53.82**	**64.99**	**64.77**	**71.09**

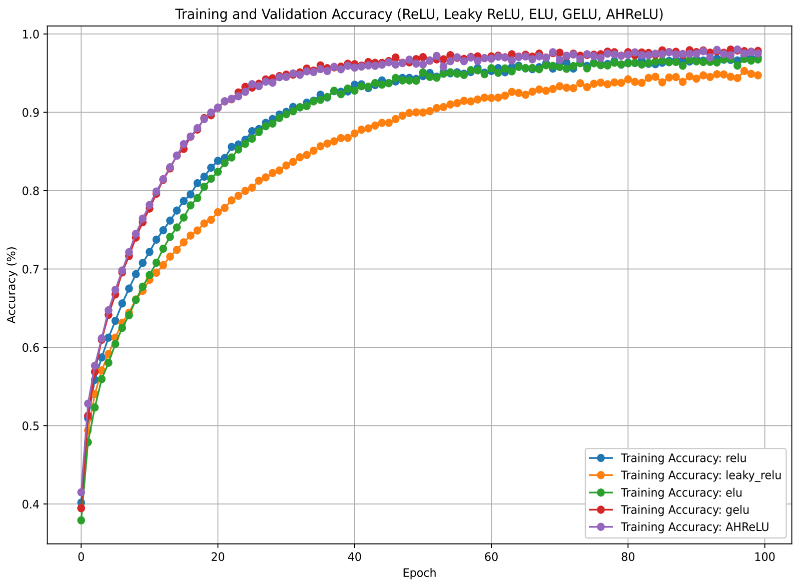

FIGURE 8. Training Accuracy (Higher Values Indicate Better Performance) on the CIFAR10 Dataset Using the LeNet Model for Activation Functions: ReLU, Leaky ReLU, ELU, GELU, and AHReLU.

for both forward and backward passes are reported in seconds (s) and are prensented in table 11. Based on the results, AHReLU performs competitively compared to other activation functions, especially when considering its computational efficiency. In terms of the forward pass, AHReLU has a relatively fast processing time of 2.85 seconds, which is comparable to or slightly better than activations like Mish (2.88 seconds), GELU (2.88 seconds), Softplus (2.92 seconds), and Swish (2.99 seconds). This suggests that AHReLU

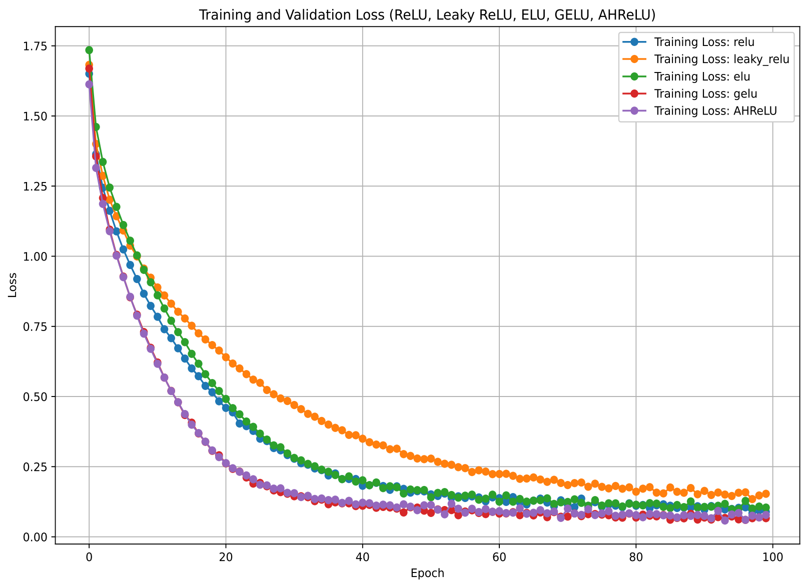

FIGURE 9. Training Loss (Lower Values Indicate Better Performance) on the CIFAR10 Dataset Evaluated with the LeNet Model for Activation Functions: ReLU, Leaky ReLU, ELU, GELU, and AHReLU.

TABLE 10. Results of Machine Translation Using Transformer Model on the WMT-2014 Dataset

Activation Function	**BLEU Score on the newstest2014 Dataset**
ReLU	27.3
Swish	27.2
Leaky ReLU	27.1
ELU	25.7
Softplus	24.9
Mish	27.1
GELU	26.3
AHReLU	**27.5**

is quite efficient in processing input through the network layers, making it a good option for scenarios where forward pass speed is crucial.

In the backward pass, AHReLU performs slightly better than Mish and Leaky ReLU, with a backward pass time of 46.61 seconds. However, this is somewhat higher than Softplus (48.00 seconds) and GELU (48.10 seconds), it is still competitive and indicates that

AHReLU provides a favorable balance between speed and accuracy in gradient computation during backpropagation.

TABLE 11. Comparative Results of Activation Functions (in seconds)

Activation	**Forward Pass Time (s)**	**Backward Pass Time (s)**
Mish	2.883642	46.622219
Swish	2.994929	48.055957
ReLU	4.830654	51.482345
GELU	2.879802	48.105213
Leaky ReLU	3.691619	46.185336
Softplus	2.915094	48.000142
ELU	5.151064	48.455081
AHReLU	2.851984	46.613886

4.4. **Convergence and Gradient Stability Analysis.** To strengthen the theoretical foundation of AHReLU, we analyzed its training convergence and gradient stability compared to ReLU, Mish, and Swish. We conducted experiments on the Fashion-MNIST data set using the WideResNet 28-10 (WRN 28-10) [40] model, where all methods were trained for 30 epochs with identical hyperparameters. For each activation, the results were averaged over five independent runs.

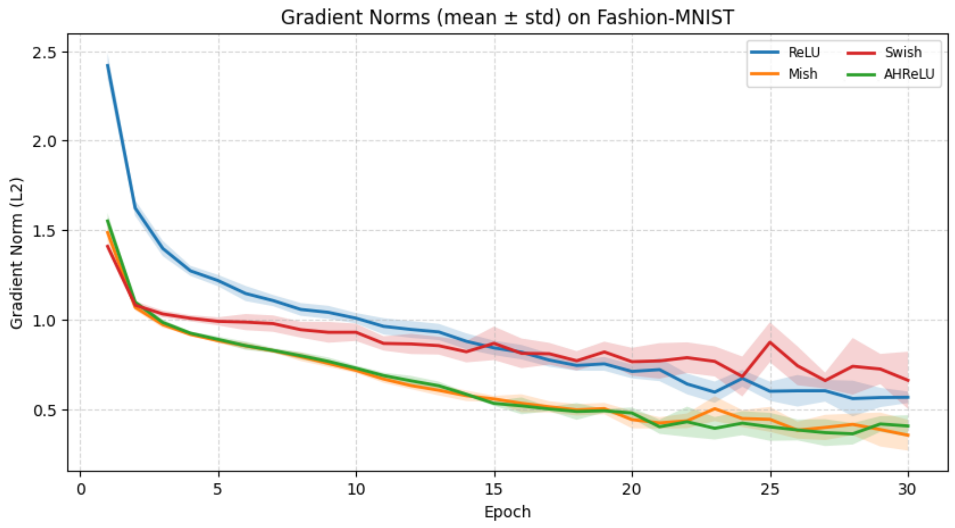

FIGURE 10. Gradient norm curves (mean ± std over 5 runs) comparing AHReLU with ReLU, Mish, and Swish on Fashion-MNIST.

The results demonstrate that AHReLU achieves smooth and stable gradient flow, avoiding sharp gradient spikes observed with ReLU and the fluctuations present in Swish. The

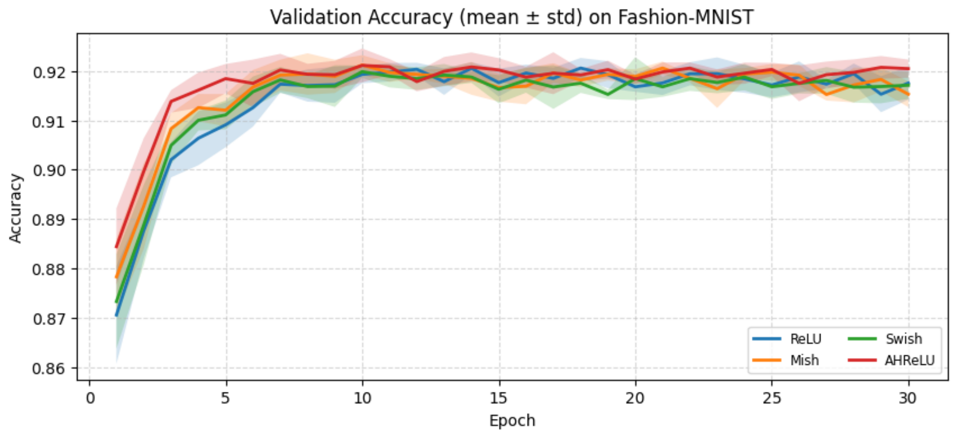

FIGURE 11. Validation accuracy convergence (mean $\pm$ std over 5 runs) on Fashion-MNIST.

TABLE 12. Convergence metric: epochs required to reach 95% of the final validation accuracy (mean $\pm$ std over 5 runs).

Activation	Epochs to 95% Final Accuracy
ReLU	1.4 ± 0.5
Mish	1.2 ± 0.4
Swish	1.2 ± 0.4
AHReLU	1.2 ± 0.4

bounded negative region of AHReLU contributes to this stability by preventing the uncontrolled growth of gradients. The Convergence analysis (Table 12) shows that AHReLU reaches 95% of its final accuracy as quickly as Mish and Swish, and faster than ReLU. These findings strengthen the theoretical foundation of AHReLU beyond universal approximation, providing empirical evidence of its favorable optimization dynamics.

5. CONCLUSION

The AHReLU activation function proposed in this study outperforms conventional activation functions such as ReLU, GELU, and Mish. During initial experiments with fixed hyperparameters, AHReLU demonstrated performance on or slightly better than ReLU. However, after optimizing hyperparameters through backpropagation and enabling the activation function to be trainable, AHReLU showed a significant boost in model performance, surpassing ReLU and other baseline activations in a range of tasks. The improved performance of AHReLU can be attributed to its nonlinearity and the inclusion of a non zero bounded negative region. This investigation underscores the value of incorporating trainable parameters in activation functions, allowing models to better adapt and optimize their performance. Extensive empirical testing across diverse deep learning applications, including image classification and machine translation, on datasets like CIFAR-10, CIFAR-100, MNIST, and WMT 2014, confirms that AHReLU delivers state-of-the-art results. Given its superior performance, AHReLU presents a promising alternative to commonly used activations such as ReLU, GELU, and Mish, offering potential for more efficient and effective deep learning models.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

FUNDING

No funding was received for this research.

AUTHORSHIP CONTRIBUTION STATEMENT

Abaid Ullah: Writing the original draft. MuhammadImran:Software and Supervision. Siama Akram: Verified results. Madeeha Tahir: Conceptualization and formal analysis

REFERENCES

[1] B. Carlile, G. Delamarter, P. Kinney, A. Marti, and B. Whitney. *Improving deep learning by inverse square root linear units (ISRLUs)*, arXiv preprint arXiv:1710.09967 (2017).

[2] C.-T. Chen and W.-D. Chang. *A feedforward neural network with function shape autotuning*, Neural networks **9**, No. 4 (1996) 627–641.

[3] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. *Fast and accurate deep network learning by exponential linear units (ELUs)*, arXiv preprint arXiv:1511.07289 (2015).

[4] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. *Incorporating second-order functional knowledge for better option pricing*, dvances in neural information processing systems,**13**, (2000) 451–457.

[5] S. Elfwing, E. Uchibe, and K. Doya. *Sigmoid-weighted linear units for neural network function approximation in reinforcement learning*, Technical Report (2017).

[6] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. Seung. *Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit*, Nature **405** (2000) 947–951.

[7] K. He, X. Zhang, S. Ren, and J. Sun. *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*, In Proceedings of the IEEE international conference on computer vision (ICCV) (2015) 1026–1034.

[8] K. He, X. Zhang, S. Ren, and J. Sun. *Identity mappings in deep residual networks*, In European conference on computer vision (ECCV), (2016) 630–645.

[9] K. He, X. Zhang, S. Ren, and J. Sun. *Deep residual learning for image recognition*, In Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), (2016) 770–778.

[10] D. Hendrycks and K. Gimpel. *Gaussian error linear units (GELUs)*, arXiv preprint arXiv:1606.08415 (2016).

[11] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. *Densely connected convolutional networks*. InProceedings of the IEEE conference on computer vision and pattern recognition (CVPR), (2017) 4700–4708.

[12] S. Ioffe and C. Szegedy. *Batch normalization: Accelerating deep network training by reducing internal covariate shift*. InInternational conference on machine learning (2015) 448-456.

[13] P. Kidger and T. Lyons. *Universal approximation with deep narrow networks.* In Conference on learning theory, (2020) 2306-2327.

[14] D. P. Kingma and J. Ba. *A method for stochastic optimization*. arXiv preprint arXiv:1412.6980, **1412**, no. 6 (2014).

[15] A. Krizhevsky and G. Hinton. *Learning multiple layers of features from tiny images*. Tech. Rep., Toronto, ON, Canada (2009).

[16] Y. Le and X. Yang. *Tiny ImageNet visual recognition challenge*. CS 231N **7**, No. 7 (2015) 3.

[17] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation applied to handwritten zip code recognition*. Neural Comput. **1**, No. 4 (1989) 541–551.

[18] Y. LeCun, *MNIST handwritten digit database*. ATT Labs. (2010).

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. *Gradient based learning applied to document recognition.* Proceeding IEEE **86**, No. 11 (1998) 2278–2324.

[20] A. L. Maas, A. Y. Hannun, and A. Y. Ng. *Rectifier nonlinearities improve neural network acoustic models*. Proceeding ICML **30**, No. 1 (2013) 3.

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

[22] A. Molina, P. Schramowski, and K. Kersting. *Padé activation units: End-to-end learning of flexible activation functions in deep networks*, arXiv preprint arXiv:1907.06732 (2019).

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

[24] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. *Reading digits in natural images with unsupervised feature learning.* InNIPS workshop on deep learning and unsupervised feature learning, **2011**, No. 5, (2011), 7.

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

[26] S. Qiu, X. Xu, and B. Cai. *FReLU: Flexible rectified linear units for improving convolutional neural networks.* In2018 24th international conference on pattern recognition (icpr) **2018**, (2018), 1223-1228.

[27] P. Ramachandran, B. Zoph, and V. Quoc Le. *Searching for activation functions*. arXiv preprint arXiv:1710.05941, (2017).

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

[29] Y. Singh and P. Chandra. *A class+ 1 sigmoidal activation functions for FFANNs.* Journal of Economic Dynamics and Control **28**, No. 1 (2003), 183-187.

[30] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. *Dropout: a simple way to prevent neural networks from overfitting.* The journal of machine learning research **15**, No. 1 (2014), 1929-1958.

[31] M. Tan and V. Q. Le. *Efficientnet: Rethinking model scaling for convolutional neural networks.* In International conference on machine learning, (2019), 6105-6114.

[32] S. Targ, D. Almeida, and K. Lyman. *Resnet in resnet: Generalizing residual architectures.* arXiv preprint arXiv:1603.08029, (2016)

[33] L. Trottier, P. Giguere, and B. Chaib-draa. *Parametric exponential linear unit for deep convolutional neural networks.* In 2017 16th IEEE international conference on machine learning and applications (ICMLA), (2017), 207-214.

[34] A. Ullah, M. Imran, M. A. Basit, M. Tahir, and J. Younis. *AHerfReLU: a novel adaptive activation function enhancing deep neural network performance.* Complexity **2025**, No. 1 (2025), 8233876

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, N. A. Gomez, L. Kaiser, and I. Polosukhin. *Attention is all you need.* Advances in neural information processing systems, **2017**, (2017), 5998-6008.

[36] H. Xiao, K. Rasul, and R. Vollgraf. *Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms*. arXiv preprint arXiv:1708.07747, (2017).

[37] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He. *Aggregated residual transformations for deep neural networks*. In Proceedings of the IEEE conference on computer vision and pattern recognition, (2017), 1492-1500.

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

[39] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. *Deep layer aggregation*. InProceedings of the IEEE conference on computer vision and pattern recognition, (2018), 2403-2412.

[40] S. Zagoruyko and N. Komodakis. *Wide residual networks.* In Proceeding British Machine Vision Conference, (2016), 1-15.

[41] H. Zheng, Z. Yang, W. Liu, J. Liang, and Y. Li. *Improving deep neural networks using softplus units.* In Proceeding International Joint Conference Neural Network, (IJCNN), (2015), 1-4.

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