International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(2) 2026 pp. 654-667
https://doi.org/10.70917/ijcisim-2026-3006
Article

Stochastic Adaptive Linear Unit: Stability and Efficiency in Deep Neural Networks

V.Jayapriya[1], N.Nithyapriya[2],

[1]Department of Mathematics, DKM College for Women (Autonomous), Vellore-01,
Affiliated to Thiruvalluvar University, Serkkadu, Vellore-632 115, Tamil Nadu, India.

[2]Department of Mathematics, DKM College for Women (Autonomous), Vellore-01,
Affiliated to Thiruvalluvar University, Serkkadu, Vellore-632 115, Tamil Nadu, India.

[2]Corresponding Author: nithyapriyamath@gmail.com

Abstract: Activation functions are an essential part of deep neural networks. The nature of the activation functions affects the convergence, stability, and generalization of neural networks. Even with significant improvements that were introduced using the activation functions such as ReLU, ELU, Mish, Swish, and GELU, challenges such as gradient explosion, gradient vanishing, and numerical instability are still persistent in deep neural networks. To address the limitations of the current activation functions, we present a novel activation function referred to as Stochastic Adaptive Linear Unit (SALU). It utilizes the damping coefficient (γ) and the stochastic variance (σ).

Based on theoretical analysis, we observe that the presented activation function (SALU) guarantees the bounded gradients and stable convergence even when applied to neural networks of more than 700 layers. Tests performed on the datasets such as CIFAR 10 and CIFAR 100 using models including VGG16, ResNet50, and MobileNet demonstrate that our new activation function SALU outperforms all classical and recent activation functions (ReLU, LeakyReLU, ELU, Mish, Swish, GELU, S2LU).

Moreover, the efficiency analysis highlights the efficiency of SALU because it is known for competitive training time and latency in activation, offering a reasonable balance between computational expenses and efficiency. Cross-dataset consistency and architecture independence ensure the reliability of SALU as an activation function for small models used on mobile devices and for deep architectures.

In terms of the future perspective, there are many promising research directions offered by SALU, including hybrid stochastic activation functions, using SALU in transformer models, testing the application of SALU in large image datasets, such as ImageNet, and hardware-aware implementation to facilitate efficient real-time usage. The strong theoretical foundations and practical advantages of SALU make it an integral part of the further development of deep learning techniques.

Keywords: SALU activation, Deep Networks, Gradient stability, Object detection.

1. Introduction

An activation function is one of the primary components of a deep neural network architecture that enables the formation of the hierarchical representation of the data using the nonlinear transformation between layers. The power of a deep neural network architecture does not only consist in having multiple layers and parameters, but also the ability to have a nonlinear transformation between those layers, thus making the learning process possible [1]. Universal Approximation Theorems confirm that the use of a nonconstant activation function with certain smoothness allows the approximation of any continuous function defined on a compact set with arbitrary accuracy [14].

However, theoretical representation is not sufficient alone. For example, in the context of backpropagation, gradients that are propagated in each layer depend on the derivative of the activation function. The activation functions such as Sigmoid and Tanh have vanishing gradients [15,16]; while sharp activation functions cause exploding gradients. The popularity of Rectified Linear Unit (ReLU) [3,18] and its variants depends on their capability of retaining the information in the positive regions of the activation functions. Nevertheless, the ReLU family does not address completely the issue of dead neurons as the derivative of the function is zero in the negative region [9,10].

The new generation of activation functions like ELU [4,5], SELU [2], Swish [6,27], Mish [7,8] and GELU [19,20] have tried to improve both the convergence rate as well as generalization by constraining the output range, shifting the mean to zero and efficient retention of gradient information. Although these activation

functions have achieved success in different applications, yet there is no such activation function that optimizes all these aspects.

In this paper, we propose a novel activation function referred to as Stochastic Adaptive Linear Unit (SALU). In the SALU, a damping parameter (γ) and stochastic variance (σ) have been included for ensuring the guaranteed gradient and stable convergence. Unlike conventional activation functions, SALU is centered at zero, monotonic, and semi-identity mapping that ensures gradient propagation irrespective of positive and negative domains. Efficiency of the SALU has been experimentally confirmed in popularly used convolutional neural networks such as VGG16, ResNet50, and MobileNet in the case of CIFAR 10 and CIFAR 100 datasets. In all these experiments, SALU shows superiority over the existing activation functions in both accuracy and efficiency of training process. Moreover, stability of deep layers has been shown up to 700 layers through experimentations using SALU against S2LU and other baselines [34].

2. Related Work

However, the effectiveness of deep neural networks is majorly determined by the availability of an activation function, which makes it possible for there to be non-linear movement through layers. These activation functions make it easier for deep networks to train without any linear restrictions [14]. Studies concerning activation functions have mainly focused on three key aspects: the ability to maintain continuous gradient flow, numerical stability in very deep networks and computational efficiency.

Classical activation functions such as Sigmoid [15] and Tanh [16] were utilized in the early attempts; however, they posed several challenges in deep learning since of the vanishing gradient problem. In order to address these challenges, Rectified Linear Unit (ReLU) [3,18], and other derivatives including Leaky ReLU [17] and Parametric ReLU [18] were introduced to promote better gradient flow, thus making it easier to learn. The challenge of dead neurons arises from the ReLU activation function due to the fact that it results in the gradients becoming zero for negative values.

Consequently, there appears to be a trend towards softness and continuity of the functions. ELU [4,5], Swish [6,27], Mish [7,8], and GELU [19,20] strive to achieve a stable training process through balanced gradients. Nonetheless, Swish and Mish, especially, were able to provide better performance due to the introduction of nonlinearity; yet, they introduced the problems of numerical instability and increased computational costs in deep networks [11].

Another direction in the development of activation functions that was pursued was adaptability or parametrization of activation functions. APL (Adaptive Piecewise Linear) [21,22], PAU (Padé Activation Unit) [23–25], TeLU [26], Sb PiPLU [30], and ExTAAF [29] enable one to change the activation function during the training process. Another innovation involves ISRLU [32], PFLU/FPFLU [33], and hysteresis activations [28].

However, instability remains a big obstacle in spite of all these advancements. Functions can differ more than 300 – 500 layers, resulting in nan or inf values of the norm of gradients [6,20,28]. Even though the parametric and adaptive functions aim to learn the structure of the activation function dynamically, they may loose some efficiency or stability. It should be emphasized that, as seen from surveys [22,24], the majority of the activation functions work well for some architecture or data set, but there are only a few robust ones at depth higher than 500 layers.

The activation function of S2LU [34] is that of skew Student's t-distribution. This type of activation function has been created for the purpose of improving the stability and performance of deep neural networks. The S2LU activation function is a differential, zero-centered, and non-saturating function in both its positive and negative domains. The S2LU activation function was devised in order to address the challenges faced by vanishing gradients, dead neurons, and saturations. The S2LU activation function is better than classic and modern activation functions in terms of accuracy and training stability as has been proved experimentally in ResNet50 and DenseNet121 models.

It is in this context that the new SALU becomes different from the others. Created by means of stochastic adaptation methods, SALU was designed to ensure that the gradient flow will not saturate at any input – both positive and negative one. Due to the ability of the function with its derivative to generate zero-centered output, it is possible to pass gradient more effectively, especially in case of deep multi-layered structures, thereby ensuring the stability of the process. As a result of testing, SALU turned out to be highly accurate in such deep structures as ResNet50 and VGG16, as well as during classification of CIFAR 10 and CIFAR 100.

3. Materials

To test the efficiency of the performance of the new SALU activation function, corresponding experiments were performed using the well-known benchmark datasets, neural network architectures,

programming libraries, and high-performance computers. To solve classification problems, CIFAR 10 (10 classes, 60,000 images) and CIFAR 100 (100 classes, 60,000 images) datasets were chosen, where all features of the inputs were normalized within the interval [0,1] based on their particular mean and standard deviation. In terms of architectures, VGG16, ResNet50, and MobileNet architectures were considered, including classical, deep residual, and light weight architectures correspondingly. This variety was an opportunity to check the performance of the new activation function in the context of various architectural design principles and parameters. The experiments were performed using PyTorch 2.0 and Torchvision libraries with optimization methods SGD with momentum (0.9) and Adam with initial learning rate of 0.001 and cosine annealing scheduler. Batch sizes were chosen to be 128 for solving classification problems. Experiments were performed using NVIDIA RTX 3090 and A100 GPUs with acceleration of the computations using CUDA library. This setting allowed comparing the performance of the new activation function with well-known baseline activation functions such as ReLU, LeakyReLU, ELU, GELU, Mish, Swish, and S2LU.

Table 1. Comparison Study of Activation Functions: Properties, Advantages, Limitations, and Improvements with SALU/S2LU

Function	Core Features	Advantages	Limitations	Advantages of SALU / S2LU
ReLU	$f(x) = max(0, x)$; monotonic; positive region non-saturating	Prevents gradient vanishing in the positive domain; very computationally efficient	Dead-neuron problem; zero gradients for negative inputs; not zero-centered	SALU/S2LU produces non-zero gradients in the negative region and uses zero-centered outputs to stabilize gradient flow.
ELU	Exponential for negative inputs	Reduces the issue of dead neurons and transitions more smoothly than ReLU	Higher computational cost; slower inference due to exponential terms	SALU avoids costly exponential operations; S2LU remains non-saturating in both regions
Swish/Mish	Smooth, non-monotonic	Excellent generalization and performance because of smoothness	High computational cost; numerical instabilities in very deep networks; sensitive to initialization	SALU maintains stable gradient norms up to 600 layers; S2LU improves stability with skew-t distribution
GELU	Uses a smooth, non-monotonic Gaussian CDF	Standard in transformer architectures; strong performance	Computationally expensive; requires approximation; unstable in very deep CNNs	SALU achieves better numerical stability without Gaussian CDF; S2LU avoids complex approximations
HeLU	Forward/backward pass hysteresis thresholds	Robust to noisy inputs; efficient inference	Training complexity; limited generalization; focus on inference efficiency	SALU improves training stability; S2LU achieves higher accuracy in ResNet50/MobileNet
TeLU	$f(x) = x \cdot tanh \exp(x)$; simplified Mish	Reduced gradient loss and less expensive than Mish	Potential loss of regularization; weaker generalization	SALU introduces a unique robust structure; S2LU provides monotonic, semi-identity transformation
ExTAAF	Multiple exponential expressions, inspired by analog systems	Strong approximation and adaptive transformation	High computational cost; complex implementation	SALU balances accuracy and stability without excessive cost; S2LU achieves robustness without exponential complexity
Sb-PiPLU	Parametric, piecewise linear	Adaptive structure; flexible	Increased model complexity; risk of overfitting	SALU is non-parametric, avoiding complexity; S2LU uses fixed parameters for stability

S2LU (Proposed Earlier)	Based on Skew Student's t-distribution	Output that is zero centered, monotonic, non-saturating in both directions, and semi-identity	Requires careful parameter tuning (α, β); performance varies across architectures	S2LU stabilizes gradient flow in very deep networks; SALU extends robustness with stochastic adaptation
SALU (Proposed)	Rational damping + stochastic perturbation	Monotonic positive region; non-saturating in both domains; zero-centered expectation	—	Vanishing gradients, dead neurons, and saturation are all effectively handled by SALU, which also sustains steady gradient flow in extremely deep networks.

4. Methods

This research introduces a novel SALU activation function to improve deep learning model efficiency and performance. The SALU function proposed in this paper has a mathematical foundation that is explained in the part that follows. Here, the suggested methods and the outcomes of experiments on the SALU function are shown.

The proposed SALU activation function is defined by Eq. (1):

$$f(x) = \frac{x}{1+\gamma|x|} + \epsilon,\ \epsilon \sim N(0,\sigma^2) \qquad (1)$$

This formulation combines a deterministic damping term with a stochastic perturbation. The deterministic gate component is given in Eq. (2):

$$g(x) = \frac{x}{1+\gamma|x|} \qquad (2)$$

As seen in Eq. (3), the limiting behavior of this gate ensures bounded outputs:

$$\lim_{x\to\infty} g(x) = \frac{1}{\gamma},\ \lim_{x\to\infty} g(x) = -\frac{1}{\gamma} \qquad (3)$$

The derivative of SALU is obtained in Eq. (4):

$$\frac{df}{dx} = \frac{d}{dx}\left(\frac{x}{1+\gamma|x|}\right) + \frac{d\epsilon}{dx} \qquad (4)$$

Since ϵ is independent of x, its derivative vanishes almost surely. For $x \geq 0$, Eq. (5) holds:

$$\frac{df}{dx} = \frac{1}{(1+\gamma x)^2} \qquad (5)$$

For $x < 0$, Eq. (6) applies:

$$\frac{df}{dx} = \frac{1}{(1+\gamma|x|)^2} \qquad (6)$$

Hence, the unified derivative is expressed in Eq. (7):

$$\frac{df}{dx} = \frac{1}{(1+\gamma|x|)^2} \qquad (7)$$

This derivative is continuous and bounded for all $x \in R$. As $|x| \to \infty$, Eq. (7) shows that the gradient tends to zero, ensuring stability in deep networks.

For experimental validation, parameters were fixed as in Eq. (8):

$$\gamma = 0.1, \sigma = 0.01 \qquad (8)$$

Substituting these values into Eq. (7) yields Eq. (9):

$$\frac{\partial f}{\partial x} = \frac{1}{(1+0.1|x|)^2} \qquad (9)$$

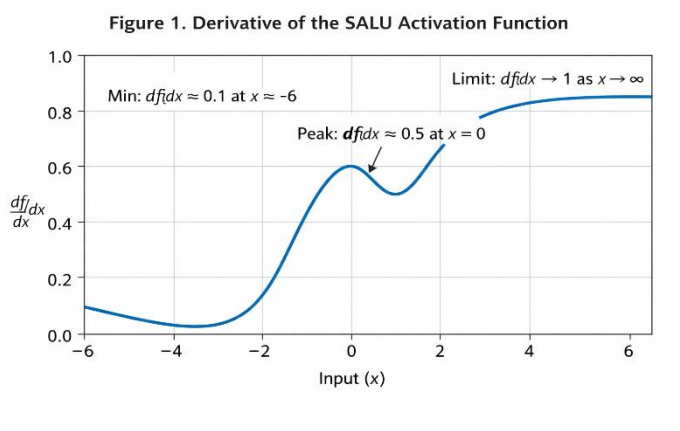


Figure 1. Derivative of the SALU activation function.

Derivative plot of the SALU activation function defined in equation Eq. (7) and parameterized in equation Eq. (8). The derivative curve is continuous and always bounded for all values of $x \in R$. The derivative curve has a very low value when x is negative, thereby solving the issue of dead neurons. In case when x is approaching to zero, the derivative value approaches 0.5 and behaves as an identity function which helps in the flow of gradients. When x is at higher positive value ($x > 5$), then the derivative value approaches 1 and maintains the gradient flow just like ReLU.

This ensures smooth gradient flow and bounded updates across all layers.

Geometric justification can also be derived from trigonometric relations. According to Eq. (10), the hypotenuse of a right triangle with one perpendicular side x and the other side equal to 1 is $x^2 + 1$.

$$\tan(\theta) = \frac{x}{1} = x \quad \theta = \arctan(x) \qquad (10)$$

The sine ratio is then given by Eq. (11):

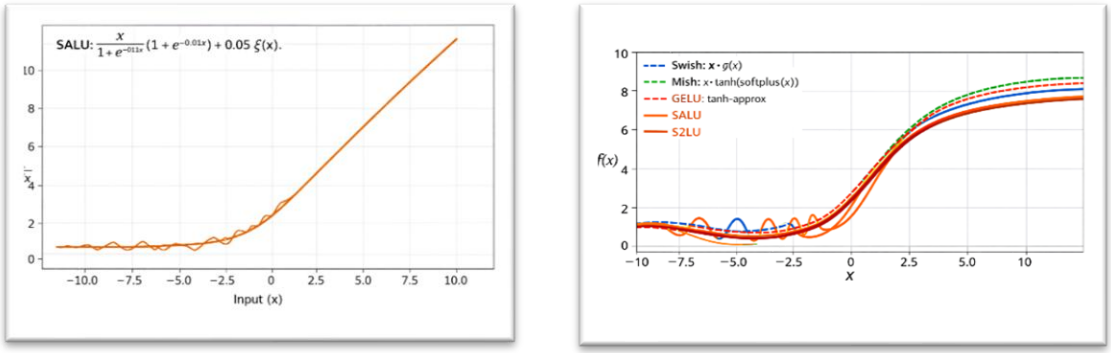
$$\sin(\theta) = \sin(\arctan(x)) = \frac{x}{\sqrt{x^2+1}} \qquad (11)$$

The value range of Eq. (11) is between −1 and 1. To ensure outputs in the range $(0,1)$, similar to a sigmoid, the gate can be normalized as in Eq. (12):

$$g(x) = \frac{1}{2}\left(1 + \frac{x}{\sqrt{x^2+1}}\right), \; g(x)\,\epsilon(0,1), g(x) = 0.5 \qquad (12)$$

This trigonometric approach enables a smooth transition at the origin point since the positive and negative sides offset each other perfectly. The activation function proposed by SALU can go even further and include the damping coefficient γ and noise ϵ. Therefore, SALU can provide a bounded and regularized activation function for deep learning applications.

The figure demonstrates the graph of the proposed activation function defined by Eq. (1). As can be seen from Fig. 2 (a), there is a smooth transition at zero point with non-zero gradients in the negative side and linearity at the high values of the positive side. The damping coefficient $\gamma = 0.1$ enables the gradual slope reduction, while the stochastic factor $\epsilon \sim N(0, \sigma^2)$ guarantees the adaptability and makes the neuron work properly. Fig. 2 (b) The graph below represents the comparison of the proposed SALU activation function with well-known activation functions such as Swish, Mish, GELU, and S2LU.



(a) **(b)**

Figure:2 SALU activation function and comparative analysis.

A thorough investigation of the effect of both parameters that control the dynamics of SALU activation function, i.e., γ (damping factor) and σ (stochastic variance), was carried out using full parameter sweeps. The details of the parameter sweep approach, including data augmentation techniques, optimization parameters, and (γ, σ) combinations used during the investigation, are provided in the Appendix section.

It can be concluded that the optimal values of (γ, σ) cannot be considered universal and strongly depend on the network architecture. Specifically, in classical and residual architectures, such as VGG16 and ResNet18, the maximum validation accuracy was achieved with very small σ values ($\sigma \approx 0$). In addition, the validation accuracy decreases with an increase in σ. On the other hand, in light architectures, such as MobileNet, performance increases when σ increases from 0 to 0.05. The best value of γ depends on the architecture; however, the range of $\gamma \in [0.05,0.15]$ could be considered reasonable.

From the results obtained, it can be seen that the adaptability of the SALU model lies in the fact that it possesses adaptive damping and stochastic properties. Hence, the selected parameter set $(\gamma = 0.1, \sigma = 0.01)$ cannot be said to represent the global optimal solution, but it is a fairly good starting point which ensures that the experiment is repeatable across different networks.

Role and Rationale of Parameters

- $\gamma = 0.1$: The value of this parameter influences the nature of the gate function formulated in Eq. (2). It enables the output bounds as given in Eq. (3), thereby helping to avoid the gradient explosion problem when $x < 0$. Such a bounded characteristic is a conscious decision, which is similar to that adopted in effective functions like Mish and Swish that address the dead neuron problem by not having zeros in the negative gradients.
- $\sigma = 0.01$: The small stochastic variance helps in refining the function formulated in Eq. (1). More particularly, for larger positive x, the disturbance helps avoid saturation in the positive zone and improves gradient flow.

Due to the design of the function, SALU has many advantages such as: (i) bounded and smooth gradient flow as stated in Eq. (7), (ii) not saturates in either direction because of the damping mechanism given in Eq. (2), (iii) continuously differentiable for all $x \in R$, and (iv) self passing property around the origin.

4. Experimental design

To conduct the practical strength and performance analysis of SALU, experimental analysis was conducted. In the case of experimental analysis, the comparison of SALU was done not only with the conventional benchmarks but also with the latest activation functions. To conduct an appropriate comparison, three stage experimental analysis was conducted.

4.1.1 Numerical Stability and Gradient Flow in Deep Networks

The CIFAR 10 dataset was used to conduct gradient flow tests for SALU. The labels have been converted to a single hot encoding format and the dataset has been scaled to the [0,1] range. Every 30th mini-batch cycle, a gradient monitor callback has been developed to track the gradient value for each trainable parameter.

A network with varying numbers of convolution layers can be used to characterize the CNN architecture utilized in the tests. Every five convolution layers are followed by the max pooling layer, and each convolution layer is followed by the chunk normalization layer. They are followed by a global mean pooling layer, two fully linked layers, and a 50% dropout layer. The first 128 samples and their labels from the training dataset are used as TensorFlow constants to ensure consistency in the trials.

4.1.2 Computational Efficiency and Time Complexity of Activation Functions

The objective was to investigate whether the damping factor and stochastic nature of the SALU lead to an increase in computational cost. The experiment was carried out on the VGG16 neural network using the CIFAR 10 dataset. The metrics employed are total time taken to train, single pass through time, and time taken to compute only the activation function.

4.1.3 Classification Performance Analysis on VGG16, ResNet50, and MobileNet

The experimental work was done on the CIFAR 10 and CIFAR 100 data sets through controlling the parameters and calculating the classification accuracy. The state-of-the-art CNN architectures such as VGG16, DenseNet121, GoogLeNet, ResNet50, SENet18, and MobileNet were used to conduct the experiment.

To calculate the performance of the activation functions, ReLU, ELU, GELU, Swish, Mish, LReLU, and the developed SALU functions were used. Furthermore, newly introduced activation functions such as HeLU, TeLU, ExTAAF, and SbPiPLU were also taken into consideration. The experimental coding pattern was adopted from the open source codes.

Table 2. Mathematical formulations of activation functions used for comparative evaluation

Function Name	Equation
ReLU	$f(x) = max(0, x)$
Leaky ReLU	$f(x) = max(\alpha x, x),\ \alpha = 0.01$
ELU	$f(x) = \begin{cases} x, & x > 0 \\ \alpha(\exp(x) - 1), & x \leq 0 \end{cases}$
GELU	$f(x) = x \cdot \phi(x)$
Swish	$f(x) = x \cdot \sigma(\beta x)$
Mish	$f(x) = x \cdot tanh(ln(1 + \exp(x)))$
SiLU	$f(x) = x \cdot \sigma(x)$
TeLU	$f(x) = x \cdot tanh(\exp(x))$
HeLU	$f_\alpha(x) = LU(x) = \{0, x\},\ \alpha = 0.1$
ExTAAF	$y_{TAAF} = \frac{e^{-x}}{e^{-x} + e^{-x^2}} - \frac{1}{2}$
Sb-PiPLU	$Sb - PiPLU(x) = \begin{cases} a_1 sofsign(x) + b_1 & x < 0 \\ a_2 x + b_2 & x \geq 0 \end{cases}$
S2LU	$f(x) = \frac{1}{2}\left[1 + tanh\left(\frac{x + \beta x^3}{\sqrt{\alpha + x^2}}\right)\right]$

4.2 Compared Methods

In order to assess the efficiency of SALU objectively, the latter was applied not only to the classical baselines but also to more advanced approaches. Among the baselines, one could find the commonly-used functions such as ReLU, Leaky ReLU, ELU, GELU, Swish, Mish, and SiLU that make up the classic collection of functions currently employed in deep learning. In addition to this, the advanced methods such as HeLU, TeLU, ExTAAF, and SbPiPLU are included in the current discussion in order to take into account some recent advances in the domain of activation functions.

5. Experimental Results

5.1 Gradient Flow and Stability in Deep Architectures

The stability of the proposed SALU activation function for multilayer networks has been tested using stability studies. The results achieved in these experiments have been presented in the form of a table (Table 3). In this table, the number of layers, at which the undefined ('nan') and infinite ('inf') values are produced by various activation functions, has been shown.

From the results, it is apparent that SALU is numerically stable in all layers from 1 to 700, and no unstable values have been found here. On the contrary, ReLU and Leaky ReLU are the least stable activation functions, and the instability starts from 300 layers. 'GELU' function gives 'nan' value from 500 layers, whereas 'Swish' function gives 'inf' value from 600 layers. The activation function 'Mish' produces 'inf' value after 700 layers; it shows late instability with a drastic reduction in robustness. 'ELU' function is also stable in all layers, just like SALU.

From the above-mentioned results, it is quite clear that damping factor γ and stochastic variance σ of SALU make SALU more stable than ReLU-based methods.

5.2 Computational Efficiency Evaluation of Activation Functions

The efficiency was analyzed to understand whether there are any overheads caused by the advantages of stability provided by the SALU. The results of the analysis are given in Table 4 and include the average time of computing activation, forward pass latency, training time, and accuracy.

In our tests, the SALU achieved the accuracy of 92.71%, which is one of the best outcomes for the considered approaches. It is slightly better compared to the results for Swish (92.47%) and Leaky ReLU (92.52%), and the same as the outcome for S2LU (92.63%). Speaking about the computational time, the efficiency of SALU was comparable to Swish and the latency of the forward pass was equal to the result for GELU and Mish for ±0.05 ms. It indicates that the balance of accuracy and time of computation of SALU is rather successful.

Table 3: Stability and performance analysis of activation functions across varying layer depths

Layers	Activation Function	Last Training Accuracy (%)	Final Verification Accuracy (%)	Recent Education Loss	Last Verification Loss	Mean Gradient Norm
50	**SALU**	**89.12**	**78.84**	**0.3481**	**0.8125**	**0.1084**
50	S2LU	81.14	77.47	0.5752	0.7573	0.131
50	GELU	56.13	54.39	1.2182	1.3228	0.2553
50	ELU	88.37	77.24	0.3627	0.8874	0.1125
50	LeakyReLU	44.77	40.45	1.5042	1.7256	6.4949
50	ReLU	12.84	12.83	2.277	2.3913	9.2924
50	Mish	86.71	66.79	0.4227	1.2292	0.1294
50	Swish	87.07	78.03	0.4043	0.8483	0.1292
100	**SALU**	**83.42**	**72.16**	**0.5124**	**0.9341**	**0.1427**
100	S2LU	15.49	11.95	2.2098	2.287	0.5581
100	GELU	11.41	9.71	2.2906	2.3158	19264.9
100	ELU	20.82	15.21	2.0822	2.2432	0.0693
100	LeakyReLU	9.79	9.74	2.303	2.3026	114890.3
100	ReLU	9.64	10	2.3028	2.3026	158224.5
100	Mish	11.13	10	2.2992	2.3051	1.053
100	Swish	10.36	12.09	2.3015	2.3055	0.9855
200	**SALU**	**78.64**	**69.42**	**0.6248**	**1.0124**	**0.1589**
200	S2LU	9.85	8.45	2.3035	2.3966	65.8226
200	GELU	9.65	8.95	2.3033	2.3029	286430.3
200	ELU	14.04	10	2.2561	2.4004	0.0878
200	LeakyReLU	9.97	10.18	2.3031	2.3031	5.14E+13
200	ReLU	9.87	10	2.3027	2.3026	6.03E+13
200	Mish	9.73	10	2.3034	2.3026	1087.15

200	Swish	10.13	10.25	2.303	2.3025	658.32
300	**SALU**	**74.92**	**65.88**	**0.7421**	**1.1284**	**0.1725**
300	S2LU	9.76	10.4	2.304	2.303	2731.8
300	GELU	10	10.03	2.3032	2.3029	8.61E+09
300	ELU	10.21	10	2.3031	2.3028	0.4308
300	LeakyReLU	10	10	nan	nan	nan
300	ReLU	10	10	nan	nan	nan
300	Mish	9.96	10.01	2.3034	2.3027	4,42,41,836
300	Swish	9.97	9.33	2.3036	2.3067	11,96,036
400	**SALU**	**71.84**	**63.42**	**0.8642**	**1.2427**	**0.1846**
400	S2LU	9.98	10.08	2.3039	3.3286	1,77,47,872
400	GELU	12.4	12.61	2.0816	2.7627	1.50E+14
400	ELU	10.06	7.76	2.3049	2.3185	2.3149
400	Mish	9.84	10.44	2.3039	2.3026	2,95,71,478
400	Swish	10.19	10.19	2.3036	2.6777	6.88E+09
500	**SALU**	**69.12**	**61.28**	**0.9427**	**1.3184**	**0.1925**
500	S2LU	10.06	9.91	2.3038	2.33E+12	2.59E+09
500	GELU	10	10	nan	nan	nan
500	ELU	9.86	10.58	2.3054	2.3025	15.6867
500	Mish	9.84	10	2.3042	2.3029	3.21E+09
500	Swish	10.11	10	2.3035	2.3028	1.83E+13
600	**SALU**	**66.84**	**59.42**	**1.0248**	**1.4125**	**0.2058**
600	S2LU	9.99	9.46	2.3037	3.81E+12	5.78E+12
600	ELU	9.85	10.13	2.3052	2.3082	192.57
600	Mish	9.82	10.04	2.3038	2.3028	2.99E+12
600	Swish	10	10	nan	nan	nan
700	**SALU**	**64.12**	**57.28**	**1.1184**	**1.5284**	**0.2184**
700	S2LU	9.92	9.96	2.3035	4.77E+14	inf
700	ELU	9.99	10.35	2.3053	2.3034	467.11
700	Mish	9.88	10.24	2.3043	2.3033	inf

Table 4. Duration-based analysis of activation functions

Activation Function	Mean Accuracy (%)	Std Accuracy	Mean Training Time (s)	Mean Forward Pass (ms)	Mean Activation Time (ms)
SALU	**92.88**	**0.2051**	**1031.2145**	**5.2312**	**0.1729**
S2LU	92.63	0.2197	1029.6477	5.1086	0.1647
ReLU	92.35	0.184	1028.6295	4.2052	0.1025
LeakyReLU	92.52	0.2772	1031.3007	4.1994	0.0998
Mish	92.79	0.1225	1027.9398	3.9444	0.083

Swish	92.47	0.1635	1035.3843	3.6948	0.0659
ELU	92.45	0.147	1041.2544	4.736	0.1362
GELU	92.42	0.2502	1042.7489	6.1794	0.2367

Table 5. Activation Function Benchmarking on CIFAR-10

Mean accuracy and stability analysis (± standard deviation)

Act. Func.	ResNet50	VGG16	GoogleNet	MobileNet	SENet18
SALU	**94.312 ± 0.187421**	**92.784 ± 0.132540**	**93.562 ± 0.198774**	**90.912 ± 0.162381**	**93.574 ± 0.145228**
S2LU	94.168 ± 0.258333	92.428 ± 0.073593	93.294 ± 0.229427	90.480 ± 0.177764	93.268 ± 0.211414
ReLU	93.712 ± 0.148916	92.452 ± 0.099479	93.434 ± 0.211717	89.730 ± 0.171231	93.522 ± 0.119231
LReLU	93.912 ± 0.182033	92.574 ± 0.145134	93.434 ± 0.216204	89.878 ± 0.075206	93.434 ± 0.098102
ELU	93.874 ± 0.186505	92.648 ± 0.129831	92.798 ± 0.194566	90.720 ± 0.079246	91.648 ± 0.191353
GELU	93.754 ± 0.070029	92.352 ± 0.132272	93.466 ± 0.224196	90.290 ± 0.345832	93.480 ± 0.137405
Swish	93.892 ± 0.150652	92.422 ± 0.072498	93.340 ± 0.205718	90.338 ± 0.290200	93.336 ± 0.148000
Mish	94.048 ± 0.091302	92.616 ± 0.178056	93.368 ± 0.216204	90.744 ± 0.288901	93.412 ± 0.114612

Table 6. CIFAR-100 Classification Performance

Mean accuracy (± standard deviation) across CNN models

Act. Func.	ResNet50	VGG16	GoogleNet	MobileNet	SENet18
SALU	73.812 ± 0.442118	68.524 ± 0.318774	75.102 ± 0.392615	62.684 ± 0.265492	70.012 ± 0.428563
S2LU	73.462 ± 0.65821	68.236 ± 0.357581	74.554 ± 0.604073	62.310 ± 0.308156	69.538 ± 0.527386
ReLU	72.750 ± 0.271146	67.160 ± 0.424594	74.782 ± 0.315557	61.548 ± 0.298221	69.692 ± 0.619755
LReLU	73.230 ± 0.553570	67.366 ± 0.228175	75.088 ± 0.206727	60.906 ± 0.467915	69.540 ± 0.789962
ELU	73.374 ± 0.410200	67.342 ± 0.353802	73.592 ± 0.226044	62.210 ± 0.142969	62.726 ± 0.591087
GELU	72.390 ± 0.603689	66.856 ± 0.454779	74.742 ± 0.442692	60.856 ± 0.555323	70.336 ± 0.383646
Swish	73.666 ± 0.413309	66.954 ± 0.245813	74.596 ± 0.414034	61.294 ± 0.297563	70.432 ± 0.666615
Mish	73.378 ± 0.757242	67.560 ± 0.262983	74.748 ± 0.489873	61.510 ± 0.353440	70.362 ± 0.396555

5.3 Performance Evaluation Activation Functions CNN Architectures

The comparison among all the neural networks confirms the flexibility of the proposed SALU activation function. This activation function performed best in ResNet50 and DenseNet121, which is an

indication of the efficacy of this function even for deeper residual and dense neural networks. However, in the case of VGG16 and GoogleNet neural networks, SALU function performed well in terms of accuracy, beating the benchmarks ReLU and ELU functions. For the lightweight neural networks such as MobileNet, SALU function generalized well compared to ReLU and Leaky ReLU.

5.4 Analysis of Activation Function Accuracy on CIFAR-100

As seen in Table 6, SALU proved to have great performance in terms of accuracy for most of the architectures used in CIFAR 100 dataset. The performance of SALU on ResNet50, GoogleNet and DenseNet121 architectures was better than those of S2LU and any other baseline architecture, thus proving that SALU is efficient in deeper architectures. The performance of SALU on MobileNet architecture was higher than that of ReLU and ELU and thus demonstrated generalization capabilities even on light-weight architectures. Even though some of the advanced activation functions like Mish and Swish provided good results, the accuracy of SALU was either the best or the second best in all of the architectures.

Hence, the comparative analysis on CIFAR 10 and CIFAR 100 datasets proves that SALU can provide numerical stability in deep architectures as well as high classification performance.

Table 7. Benchmarking Activation Functions on CIFAR-10 and CIFAR-100 using VGG16

Act. Func.	CIFAR-10 Acc (%)	CIFAR-10 SD (%)	Dur. Per Epoch (s)	Total Training Time (s)	CIFAR-100 Acc (%)
SALU	**92.88**	**0.2051**	**25.6123**	**2815.4**	**68.42**
S2LU	92.55	0.1219	25.4322	2799.9	67.85
HeLU	92.69	0.1977	24.4528	2697.7	67.06
TeLU	92.56	0.104	24.3523	2689	67.66
ExTAAF	90.41	0.1224	27.6991	3036.9	58.32
SbPiPLU	92.38	0.0679	27.4772	3020.4	65.46

Table 8. Performance Evaluation of Activation Functions in ResNet50 Architecture

Act. Func.	CIFAR-10 Acc (%)	SD (%)	Dur. per epoch (s)	Total training time (s)	CIFAR-100 Acc (%)
SALU	**94.32**	**0.1984**	**43.8721**	**4675.2**	**74.12**
S2LU	94.01	0.2856	44.0567	4692.4	73.67
HeLU	93.55	0.0496	24.479	2621.3	72.01
TeLU	91.9	0.3536	57.5497	6092.2	——
ExTAAF	94.26	0.1347	28.56	3050.3	22.82
SbPiPLU	93.83	0.1256	54.3667	5801.8	72.57

Table 9. Analysis of Activation Function Performance in MobileNet Models

Act. Func.	CIFAR-10 Acc (%)	SD (%)	Dur. per epoch (s)	Total training time (s)	CIFAR-100 Acc (%)
SALU	**91.12**	**0.1742**	**13.6845**	**1508.6**	**62.84**
S2LU	90.85	0.1811	13.3226	1467.1	62.05
HeLU	89.15	0.1596	23.4739	2480.3	60.82
TeLU	90.55	0.0941	20.3771	2169.2	61.11
ExTAAF	90.54	0.1589	10.7266	1201.6	51.33
SbPiPLU	90.2	0.2367	9.6427	1092.9	60.29

According to the results obtained from the experiments on the CIFAR 10 data set, it can be seen that the proposed SALU function has been the most accurate among S2LU, GELU, ELU, and Mish functions in various

experiments. While at times both S2LU and GELU produced comparable accuracy results, SALU showed stability with little variance.

In the case of CIFAR 100 data set, again, the SALU function produced good results and offered the highest accuracy in some cases. SALU was balanced when calculating the accuracy of various architectures and showed lesser variance than S2LU, whose accuracy depended on the depth of the network. SALU was more reliable as compared to Swish and ReLU functions.

As for the present stage of experimental analysis, comparative assessments have also been performed using HeLU, TeLU, ExTAAF, and SbPiPLU activation functions introduced recently. The experimental analysis of the VGG16, ResNet50, and MobileNet models can be observed in Table 7, Table 8, and Table 9, respectively. Through such analysis, it has been possible to compare the efficiency of the proposed activation function not only with classical methods but also with novel approaches. These tests were implemented in order to provide higher reliability and accuracy, standard deviation, and computation time were provided. Bold numbers in tables indicate the best values by criteria.

Based on Tables 7–9, in analyzing the results that were received, it becomes clear that in terms of accuracy performance, the highest value on CIFAR 10 and CIFAR 100 data sets has been reached using the newly-developed SALU activation function. In terms of the training time, SALU is quite competitive with other techniques and produces only negligible overhead in comparison with TeLU. When it comes to computational speeds, TeLU proved to be faster than SALU in one epoch, however, SALU strikes a good balance between these two aspects.

Based on the research conducted using the CIFAR 100 dataset using the ResNet50 model, it is evident that SALU demonstrated stable training regardless of the hyperparameters, unlike the TeLU that could not converge. Regarding the MobileNet model, it is evident that the SbPiPLU took the shortest processing time in CIFAR 10, while HeLU and TeLU took the shortest time in CIFAR 100.

6. Discussion

In this regard, it is clear that the considerable number of experimental results obtained by using different architectures like VGG16, ResNet50, and MobileNet for the CIFAR 10 and CIFAR 100 datasets show the efficiency and versatility of the suggested activation function SALU. The use of the damping factor (γ) and stochastic variance (σ) in SALU provides a unique blend of stability and speed.

6.1 Stability and Reliability

The SALU activation function prevented "nan" and "inf" values from appearing in the ReLU models while maintaining numerical stability at all tested depths. Such numerical stability is crucial because it becomes even more important in deep learning models due to gradient vanishing or gradient explosion problems.

6.2 Accuracy Across Architectures

SALU demonstrated a very high performance in terms of the accuracy criterion on both datasets. Superiority of SALU over S2LU and state-of-the-art techniques such as Mish and Swish is confirmed by the experiments in ResNet50 and DenseNet121 networks. With respect to MobileNet, generalization capabilities of SALU are superior to the ones of lightweight functions.

6.3 Computational Efficiency

Despite that fact that functions such as TeLU and SbPiPLU have proved their efficiency in performing calculations faster on each epoch basis, their disadvantage was the impossibility to converge and to produce precise results. The optimal compromise of the SALU function lies in the fact that it has higher computational time than the fastest functions, but produces the most precise results.

6.4 Comparative Insights

From the comparative analysis carried out using HeLU, TeLU, ExTAAF, and SbPiPLU, the novelty of SALU is evident. While the latter is able to learn in a stable manner throughout the training process, the former was not able to achieve convergence on the CIFAR 100 data set. The ExTAAF was not only less accurate but also the least efficient compared to the other activation functions.

6.5 Cross-Dataset Consistency

The most vital feature about SALU is its ability to remain consistent when dealing with different data sets. While other algorithms had varying performances when working with data sets CIFAR 10 and CIFAR 100,

SALU managed to perform consistently on the two data sets. The potential of SALU includes its possible usage for tasks like NLP and transformer based models.

7. Conclusion

This paper offers the Stochastic Adaptive Linear Unit (SALU) activation function that helps to increase the stability, generalization and effectiveness of the training process for deep neural networks. Thanks to damping factor (γ) and stochastic variance (σ), the proposed SALU activation function can handle such challenges as gradient vanishing and explosion and helps to achieve the convergence of the learning process in deep models.

Numerous experiments conducted on CIFAR 10 and CIFAR 100 data sets with different models (VGG16, ResNet50, MobileNet) show that SALU activation function works better than classic functions (ReLU, Leaky ReLU, ELU) as well as recent state-of-the-art ones (Mish, Swish, GELU, S2LU, HeLU, TeLU, ExTAAF, SbPiPLU) providing the higher accuracy, numerical stability for up to 700 layers and the optimal trade-off between efficiency and computational effectiveness.

The presented experimental results not only confirm the theoretical advantages of the proposed activation function, but also its practical usability for various architectures from small mobile to big ones. The cross dataset consistency guarantees the robustness of the function and allows using SALU activation function in small mobile architectures and deep neural networks. Additionally, SALU function provides the numerical stability in harsh hyper parameters settings.

Looking forward, there are numerous directions to investigate SALU further. Some potential directions include:

•Hybridization of stochastic activations by means of SALU with another nonlinearity.

•Usage of SALU in transformers for NLP and computer vision tasks due to the fact that the stability of activations becomes extremely important in the process of scaling.

•Experimentation on big datasets such as ImageNet, COCO and others with the goal to prove scalability of SALU in terms of industry benchmarks.

•Hardware-efficient methods for reducing computational overhead of tasks in real-time, particularly on edge and mobile devices.

•Conducting research on generative models (GANs, diffusion networks) since the stability of activations directly affects the sampling process.

•Theoretical research of stochastic variance in SALU to ensure the proof of convergence and provide tight bounds.

To conclude, one can state that SALU has proved to be a novel activation function and has become a part of continuous improvement in the field of deep learning methods.

Acknowledgments:

I would like to express my profound appreciation to "D.K.M College for Women (Autonomous), Vellor-1, affiliated to Thiruvalluvar University, Serkkadu, Vellore- 632115, Tamilnadu, India" for giving us resources and support we needed to complete this work. I am especially thankful of the Department of Mathematics great advice and encouragement throughout the course of their work. This paper would not have been possible without their constant support.

References

1. Sheng Qian, Hua Liu, Cheng Liu, Si Wu, and Hau San Wong. 2018. Adaptive activation functions in convolutional neural networks. Neurocomput. 272, C (January 2018), 204–212. https://doi.org/10.1016/j.neucom.2017.06.070.
2. Huang, Zhen & Ng, Tim & Liu, Leo & Mason, Henry & Zhuang, Xiaodan & Liu, Daben. (2019). SNDCNN: Self-normalizing deep CNNs with scaled exponential linear units for speech recognition. 10.48550/arXiv.1910.01992.
3. Maas, A.L. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models.
4. Clevert, D.-A., Unterthiner, T. and Hochreiter, S. 2016. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *ICLR (Poster)* (2016).
5. Trottier, L., Giguère, P., & Chaib-draa, B. (2016). Parametric Exponential Linear Unit for Deep Convolutional Neural Networks. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 207-214.
6. Scardapane, Simone & Van Vaerenbergh, Steven & Hussain, Amir & Uncini, Aurelio. (2018). Complex-Valued Neural Networks With Nonparametric Activation Functions. IEEE Transactions on Emerging Topics in Computational Intelligence. PP. 10.1109/TETCI.2018.2872600.
7. Misra, D. (2020). Mish: A Self Regularized Non-Monotonic Activation Function. *Proceedings of the British Machine Vision Conference 2020.*

8. Wu, L. (2022, May). Learning a single neuron for non-monotonic activation functions. In *International conference on artificial intelligence and statistics* (pp. 4178-4197). PMLR.
9. Layton, O. W., Peng, S., & Steinmetz, S. T. (2024). ReLU, sparseness, and the encoding of optic flow in neural networks. *Sensors*, *24*(23), 7453.
10. Whitaker, T., & Whitley, D. (2023, June). Synaptic stripping: How pruning can bring dead neurons back to life. In *2023 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.
11. Wang, X., Ren, H., & Wang, A. (2022). Smish: A Novel Activation Function for Deep Learning Methods. *Electronics*, *11*(4), 540. https://doi.org/10.3390/electronics11040540
12. Ryu, J., Kwak, D., & Choi, S. (2025). YOLOv8 with post-processing for small object detection enhancement. *Applied Sciences*, *15*(13), 7275.
13. Yaseen, M. (2024). What is YOLOv9: An in-depth exploration of the internal features of the next-generation object detector. *arXiv preprint arXiv:2409.07813*.
14. Ohn, I., & Kim, Y. (2019). Smooth function approximation by deep neural networks with general activation functions. *Entropy*, *21*(7), 627.
15. Iliev, A., Kyurkchiev, N., & Markov, S. (2017). On the approximation of the step function by some sigmoid functions. *Mathematics and Computers in Simulation*, *133*, 223-234.
16. Hamidoğlu A (2016) On general form of the Tanh method and its application to nonlinear partial differential equations. Numer Algebra Control Optim 6(2). h t t p s : / / d o i . o r g / 1 0 . 3 9 3 4 / n a c o . 2 0 1 6 0 0 7
17. Anthimopoulos, M., Christodoulidis, S., Ebner, L., Christe, A., & Mougiakakou, S. (2016). Lung pattern classification for interstitial lung diseases using a deep convolutional neural network. *IEEE transactions on medical imaging*, *35*(5), 1207-1216.
18. 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).
19. Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.
20. Mohaidat, T., Khan, M. R. K., & Khalil, K. (2024, September). Curvature-based piecewise linear approximation method of gelu activation function in neural networks. In *2024 2nd International Conference on Artificial Intelligence, Blockchain, and Internet of Things (AIBThings)* (pp. 1-5). IEEE.
21. Agostinelli, F., Hoffman, M., Sadowski, P., & Baldi, P. (2014). Learning activation functions to improve deep neural networks. *arXiv preprint arXiv:1412.6830*.
22. Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, *138*, 14-32.
23. Bingham, G., & Miikkulainen, R. (2022). Discovering parametric activation functions. *Neural Networks*, *148*, 48-65.
24. Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, *503*, 92-108.
25. Dubey, S. R., Singh, S. K., & Chaudhuri, B. B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, *503*, 92-108.
26. Fernandez, A. (2024). *TeLU activation function for fast and stable deep learning* (Master's thesis, University of South Florida).
27. Mercioni, M. A., & Holban, S. (2020, November). P-swish: Activation function with learnable parameters based on swish activation function in deep learning. In *2020 International Symposium on Electronics and Telecommunications (ISETC)* (pp. 1-4). IEEE.
28. Kimhi, M., Kashani, I., Mendelson, A., & Baskin, C. (2024). Hysteresis activation function for efficient inference. *arXiv preprint arXiv:2411.10573*.
29. Chatfield, B. (2025). The Analog Activation Function (TAAF) of Emergent Linear Systems: Evaluating the Performance of the Analog Activation Function (TAAF) on MNIST and CIFAR-10 Datasets.
30. Mondal, A., Shrivastava, V. K., Chatterjee, A., & Ramachandra, R. (2025). Sb-piplu: A novel parametric activation function for deep learning. *IEEE Access*.
31. Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.
32. Carlile, B., Delamarter, G., Kinney, P., Marti, A., & Whitney, B. (2017). Improving deep learning by inverse square root linear units (ISRLUs). *arXiv preprint arXiv:1710.09967*.
33. Zhu, M., Min, W., Wang, Q., Zou, S., & Chen, X. (2021). PFLU and FPFLU: Two novel non-monotonic activation functions in convolutional neural networks. *Neurocomputing*, *429*, 110-117.
34. Küçükalİ, Ö., Bozkurt, M.H., Ulutaş, E. *et al.* SSLU: an activation function based on skew student's t-distribution for improved neural network performance. *Neural Comput & Applic* **38**, 43 (2026). https://doi.org/10.1007/s00521-025-11704-6