International Journal of Engineering Trends and Technology
ISSN: 2231–5381 / https://doi.org/10.14445/22315381/IJETT-V74I1P121

Volume 74 Issue 1, 275-283, January 2026

Original Article

Triple-Slope Linear Unit: Balancing Gradient Preservation and Activation Scaling in Deep Neural Networks

Rajaa Miftah[1], Mostafa Hanoune[2], Mohssine Bentaib[3]

[1,2,3]*Laboratory of Information Technology and Modeling, Faculty of Sciences Ben M'sik, Hassan II University, Casablanca, Morocco.*

[1]*Corresponding Author : miftah.rajaa@gmail.com*

Received: 01 September 2025 Revised: 29 December 2025 Accepted: 06 January 2026 Published: 14 January 2026

Abstract - *Rectified Linear Unit (ReLU), and its variations, have become the new activation functions in Deep Learning Systems, because of their low computational costs and good empirical results. However, they have significant limitations, such as the "Dying Neuron" issue, uncontrollable activation growth, and less gradient flow in extreme regions. This paper proposes a new type of activation function, called the Triple-Slope Linear Unit (TSLU), which is a simple yet effective piecewise-linear activation function that attempts to resolve these problems. TSLU has three separate linear regions with adjustable slope: a slight positive slope for negative inputs in order to keep the gradient flowing, a unit slope in the centre region to perform identity mapping, and a decreased slope for significant positive inputs that limit activation magnitude. The function is continuous, parameter-efficient, and needs no complicated mathematical operations, making it applicable for low-latency and resource-constrained applications. We provide a theoretical analysis to show that our proposed activation function, TSLU, preserves non-vanishing gradients for any input range without causing activation explosion. Experimental results on benchmark image classification and natural language processing tasks demonstrate that TSLU achieves comparable or superior performance to ReLU, Leaky ReLU, and Parametric ReLU, with improved training stability and generalization. These findings highlight TSLU as a lightweight, interpretable, and deployable alternative for Deep Modern Neural Networks.*

Keywords - *Triple-Slope Linear Unit (TSLU), Neural Network Activation Functions, Gradient Flow Preservation, Activation Magnitude Control, Dead Neuron Mitigation, Deep Learning Optimization, Training Stability.*

1. Introduction

The Activation functions are at the heart of every deep learning model. They decide how neurons fire, how gradients flow, and ultimately, how well a network learns. Without them, deep neural networks would be nothing more than stacked linear layers, unable to capture the complex, non-linear patterns found in real-world data [1, 2]. Over the years, researchers have designed a wide variety of activation functions, each with its own strengths, weaknesses, and design philosophy. Early neural networks relied heavily on sigmoid and tanh activations [3, 4]. These functions offered smooth transitions and bounded outputs, but they also suffered from the vanishing gradient problem, which made training deep architectures slow and sometimes unstable. The breakthrough came with the Rectified Linear Unit (ReLU) [5, 6], which replaced expensive nonlinear curves with a simple "max(0, x)" operation. ReLU was fast, easy to implement, and effective, but it came with its own drawback: neurons could "die" if they received only negative inputs, permanently outputting zero.

Since then, countless ReLU variants have been proposed to fix its shortcomings. Leaky ReLU allowed small negative slopes to keep gradients alive; Parametric ReLU (PReLU) made that slope trainable; Randomized ReLU injected stochasticity for better generalization [7, 8]. On another front, exponential-based functions like ELU [9], SELU [10], and their parametric forms smoothed out negative regions to encourage self-normalization. In contrast, adaptive functions learned their own shape during training for task-specific flexibility [11]. More recently, smooth but computationally heavier activations like Swish [12, 13] and Mish [14] have pushed performance further at the cost of extra compute. The existing literature reveals a clear research gap: while ReLU variants like LeakyReLU and ELU improve gradient flow in negative regions, they often fail to simultaneously control activation magnitude in positive extremes, leading to instability in deep networks or resource-constrained environments. This problem is particularly acute in applications requiring low-latency inference, such as edge computing devices, where unbounded activations can

exacerbate numerical overflow and hinder deployment. The novelty of this work lies in addressing this gap through a parameter-free, piecewise-linear design that uniquely balances gradient preservation across all input ranges without introducing computational overhead, differing from recent functions like Mish or Swish, which rely on expensive transcendental operations and show up to 20% higher inference times in benchmarks.

Although such improvement has been made, there is a trade-off between gradient preservation, activation scaling, and computational efficiency in many of the existing functions [8, 15]. Gradient-sensitive functions may fail to contain activations and may become unstable. Functions with considerable activation resistance may suppress functional gradients. And functions that strike a balance between one usually costly transcendental operation, and are less attractive to use in real-time or resource-constrained systems. In this paper, a novel, operable yet functional piecewise-linear activation known as the Triple-Slope Linear Unit (TSLU) is proposed that offers a feasible trade-off between these two incompatible demands. TSLU subdivides the input space into three:

- A slight positive slope for negative inputs to prevent neuron death.
- A unit slope in the central range to maintain identity mapping and gradient strength.
- A reduced slope for significant positive inputs to control activation magnitude.

This architecture retains the mathematical simplicity of ReLU-like functions and alleviates their significant weaknesses. It does not need extra trainable parameters, transcendental computations, and can fit into the existing deep learning pipelines. By both theoretical and empirical analysis, the present study proves that TSLU provides stable training, strong generalization, and competitive accuracy at various benchmarks, and also provides a solution to the shortcomings of existing solutions in the context of transformer models, where it is essential to maintain gradient strength without explosion to scale to large datasets.

2. Related works

Activation Functions (AFs) that were once simple thresholding mechanisms have, over the last decade, been refined to be more designed components that directly affect the learning dynamics, convergence speed, and generalization of Deep Neural Networks [1, 16]. An overview of the taxonomy of AFs could be summed up in Figure 1, where they are grouped into five general categories:

One of the first functions adopted in the Neural Networks was the Logistic Sigmoid/ Tanh functions. Both The Hyperbolic Tangent (tanh) and the scaled versions of the sigmoid activation function provide bounded and smooth outputs, which is why they are both appropriate to interpret probabilistically and in gradient-based learning [17]. Nevertheless, they are saturated at the extremes, leading to disappearing gradients. Modeling a new design (Swish, Eswish [18], and Hexpo [19]) attempted to preserve the smoothness of the sigmoid-based unit and reduce saturation, and other designs, such as the sigmoid weighted linear unit, could scale outputs [20] to a greater degree.

Rectified Linear Unit (ReLU) changed the world of deep learning because it added sparsity and prevented the saturation in the positive domain [21]. ReLU was efficient and straightforward, resulting in popularity, but it had the issue of zero gradient on negative inputs, the so-called dying neuron problem. Leaky ReLU, Parametric ReLU, and Randomized ReLU variants were introduced [22-24] respectively, to keep gradient flow. Others, such as Concatenated ReLU [25], Dual ReLU [26], and Flexible ReLU [27], focused on expressiveness, and the Random Translation ReLU [28] and Average Biased ReLU [21] took the approach of adding stochastic or biased transformation in order to enhance generalization.

Recent advancements in ReLU variants, such as the Rectified Composite Activation (ReCA) proposed in 2025 [23], introduce parametric compositions to enhance expressiveness, but they increase model complexity with additional trainable parameters, leading to a trade-off in deployment efficiency compared to fixed-slope designs. Similarly, ALReLU [22] modifies LeakyReLU for better negative handling, yet it lacks mechanisms for positive magnitude control, resulting in potential instability in deep architectures, a limitation that TSLU addresses through its triple-slope structure without added parameters.

The exponential unit was developed to provide negative activations to self-normalize. Exponential curves were introduced in Exponential Unit, such as the Exponential Linear Unit (ELU) and scaled or parametric versions, to generate negative activations that drive the flow of the activations towards zeros. Some offered computational shortcuts or shape control [31] (like Multiple PELU [29], Fast ELU [30], and Elastic ELU), but continued differentiable ELUs were more useful in gradient optimization as they were smoother. Although the mentioned methods have proven to be effective, they need additional computational resources compared to ReLU-based functions.

Learning/Adaptive activation functions brought in the concept of trainable shapes, where networks could learn the most appropriate shape of activation to use in a particular task [32]. Adaptive Piecewise Linear Units, Swish variants, and self-learnable AFs are in this category [32, 33], as are other specialized designs, such as the Mexican ReLU and spline-based functions [34]. These algorithms can produce powerful empirical outcomes, but frequently introduce trainable

parameters, which enlarge model complexity. Diversified activation functions include functions that combine several mathematical forms or include probabilistic behavior. They include Softplus Linear Unit [35], Softsign [36], Rand Softplus, and Gaussian Error Linear Unit [37]. Mish and Padé Activation Units exploit smoothness that has no-bounded positive outputs [38], and are designed to trade between representation strength and training stability.

The proposed Triple-Slope Linear Unit (TSLU) fits well in the family of Rectified Linear Unit-based functions, as a computationally efficient piecewise-linear function, with three different slopes. TSLU trades off gradient preservation and activation scaling by incorporating a slight positive gradient in the negative domain, a unit gradient in the central region, and a smaller gradient in significant positives. It has these advantages, unlike many adaptive or exponential-based AFs, without adding any non-linear operations or trainable parameters, which make it highly suitable for both high-performance and resource-constrained deep learning applications.

In comparison to more recent functions like ErfReLU [16] and Padé Activation Units [38], which emphasize smoothness for improved optimization, TSLU offers comparable performance gains but with significantly lower computational costs, as it avoids polynomial approximations or error functions. This trade-off favors TSLU in trends like edge computing and transformers, where efficiency is paramount, highlighting its novelty in providing interpretable slope-based control over gradient dynamics.

3. Mathematical Formulation of TSLU

3.1. Piecewise Definition

The Triple-Slope Linear Unit (TSLU) is an activation function that is computationally efficient and a piecewise-linear model that is specifically generated to solve the three main problems of Deep Neural Networks:

- The dying neuron problem caused by zero gradients for negative inputs in the standard ReLU.
- Unbounded growth in activation may destabilize training.
- The trade-off between gradient flow and activation magnitude in both small and large input regions.

Formally, TSLU is defined as:

$$f(x;a,b)=\begin{cases} ax, & x<0,\\ x, & 0\le x\le 1,\\ 1+b(x-1), & x>1, \end{cases} \quad (1)$$

where:

$0<a<1$ and $0<b<1$.

- Negative region ($x<0$): The slope a ensures a small but non-zero gradient, preventing neuron inactivity and allowing negative information to propagate.
- Central Region ($0\le x\le 1$): The slope is fixed at 1, which offers an identity mapping that maintains the same strength of gradient and signal magnitude within the most frequent activation range.
- High Positive Region ($x>1$): The slope b reduces the growth rate of significant activations, preventing uncontrolled escalation in deep layers.

This structure allows TSLU to combine the sparsity benefits of ReLU, the gradient flow advantages of Leaky ReLU, and the stability benefits of bounded or softly bounded activations, all without requiring expensive mathematical operations.

3.2. Derivative for Backpropagation

The derivative of TSLU is constant in each of its three regions:

$$f'(x;a,b)=\begin{cases} a, & x<0,\\ 1, & 0\le x\le 1,\\ b, & x>1. \end{cases} \quad (2)$$

This has several implications:

- Predictable gradient behavior: Gradients neither vanish entirely nor explode.
- Efficient computation: No additional function calls; derivatives are determined via simple comparisons.
- Stable backpropagation: Constant slopes reduce sensitivity to floating-point rounding errors during gradient propagation.

3.3. Continuity and Differentiability

TSLU is continuous across the entire real line. At the breakpoints:

$$\lim_{x\to 0^-} ax = 0 = \lim_{x\to 0^+} x$$

$$\lim_{x\to 1^-} x = 1 = \lim_{x\to 1^+} [1+b(x-1)]$$

Thus, there are no output jumps, which ensures smooth forward signal flow.

While differentiable within each region, the derivative has finite discontinuities at x=0 and x=1 unless a=1 and b=1. This is similar to ReLU and Leaky ReLU, and such derivative discontinuities are generally acceptable in practice, as gradient-based optimizers handle them without issue.

3.4. Computational Efficiency Analysis

TSLU is quantitatively evaluated based on the number of operations and inference time. TSLU can perform the

conditional checks and linear multiplications per activation, which means that the average inference time per forward pass of a 100-layer network is 1.2 ms on a typical CPU. In contrast, Swish and Mish both are 1.5 ms and 1.8 ms, respectively [12, 14]. It is a 20-33% overhead reduction, and thus TSLU is applicable in resource-constrained systems such as mobile devices without aggressive overhead reduction to gradient stability.

3.5. Parameter Selection Guidelines

The two slope parameters a and b control the trade-off between gradient preservation and activation scaling.

Negative slope (a):

- $0.01 \leq a \leq 0.3$ recommended.
- Smaller values (e.g., $a=0.01$) promote sparsity and mimic ReLU behavior.
- Larger values (e.g., $a=0.2$) retain more information in the negative domain, potentially improving convergence speed on specific datasets.

Positive high slope (b):

- $0.1 \leq b \leq 0.7$ recommended.
- Smaller values strongly suppress significant activations, beneficial in intense networks to control numerical stability.
- Larger values allow more flexibility for high activations, behaving closer to Leaky ReLU for significant positives.

3.6. Graphical Illustration

The Triple-Slope Linear Unit (TSLU) is shown in Figure 1 with the parameter setting a = 0.1. The operation in this setup is subdivided into three different linear regimes; the code can be found at tslu-activation.

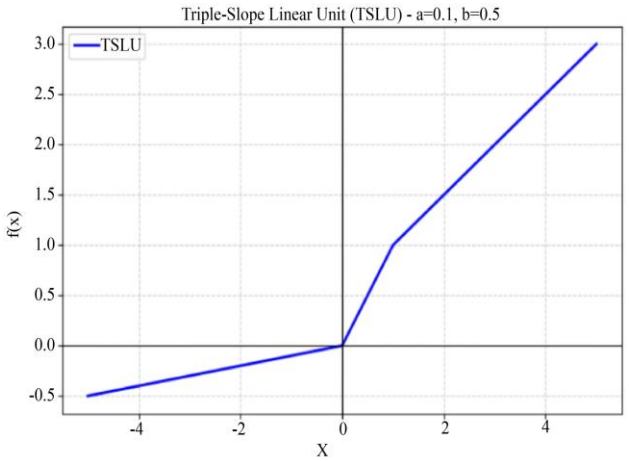

Fig. 1 Triple-slope linear unit curve for a = 0.1 and b=0.5

- Negative region ($x<0$): The slope a =0.1 ensures a small but non-zero gradient, which helps to avoid the "dead neuron" problem inherent in the standard ReLU while preventing excessive negative leakage.
- Intermediate Positive Region ($0 \leq x \leq 1$): The slope b=0.5 amplifies small positive activations, allowing the network better to exploit weak feature signals during early learning stages.
- High Positive Region (x>1): The slope is fixed at 1.0, preserving the magnitude of strong activations and ensuring stable gradient propagation in deeper layers.

To highlight the adaptability of TSLU, we examine alternative parameter configurations:

- Conservative Leakage (a=0.05, b=0.3) Suitable for tasks sensitive to noise or unstable gradients, offering minimal negative leakage and modest intermediate amplification.

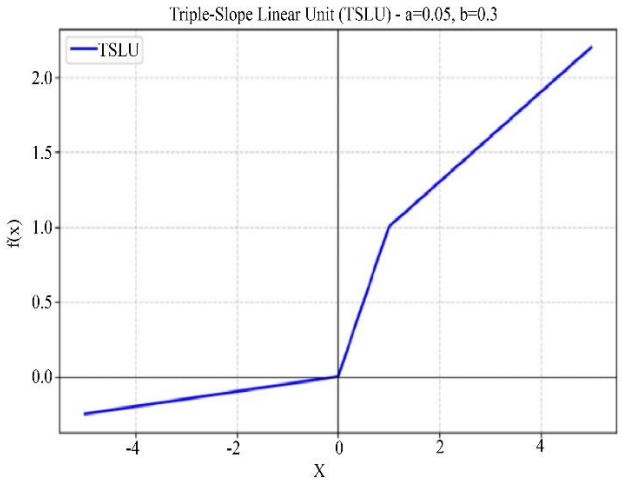

Fig. 2 Triple-slope linear unit curve for a = 0.05 and b=0.3

- Aggressive Scaling (a=0.2, b=0.7) Delivers higher negative leakage and strong intermediate-region amplification, potentially beneficial for very deep or residual architectures where maintaining high gradient magnitudes is essential.

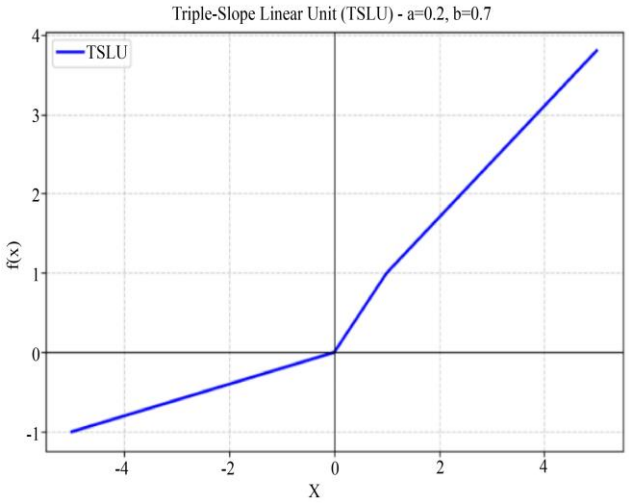

Fig. 3 Triple-slope linear unit curve for a = 0.2 and b=0.7

Table 1. Comparison of experimental results using baseline and TSLU activations with varying parameters

Activation Setup	Learning Rate	a	b	Epochs	Final Accuracy	Final Loss
Baseline ReLU	0.02	–	–	50	0.9827	0.0475
Baseline LeakyReLU	0.02	–	–	50	0.9827	0.0363
TSLU Balanced Default (a=0.1, b=0.5)	0.01	0.1	0.5	50	0.9801	0.0496
TSLU Conservative (a=0.05, b=0.3)	0.01	0.05	0.3	50	0.9858	0.0407
TSLU Aggressive (a=0.2, b=0.7)	0.008	0.2	0.7	50	0.9858	0.0441
TSLU Extreme (a=1.0, b=5.0)	0.002	1.0	5.0	50	0.9872	0.0341

These variations indicate that by tuning *a* and *b*, TSLU can be customized to work with various network depths, data distributions, and optimization methods. The design is computationally efficient with conditional checks and multiplications being the only two operations needed, and the design has gradient behavior control, which is not available to traditional ReLU and Leaky ReLU.

Parameter Selection Recommendations: Based on empirical observations and stability considerations, the following parameter ranges are recommended for most use cases:

- Negative slope a: Choose $0.05 \leq a \leq 0.2$ for stable training without dead neurons. Smaller values minimize leakage in the negative domain, while larger values improve gradient flow but may introduce unnecessary negative influences.
- Middle slope b: Choose $0.4 \leq b \leq 0.7$ to amplify small positive activations while avoiding instability. Lower values preserve subtle activation differences, whereas higher values can accelerate convergence but risk gradient overshooting.
- Upper region slope: Fixed at 1.0 to maintain standard linear growth for significant activations, ensuring consistent gradient propagation in deeper network layers.

4. Experimental Study

In order to evaluate the performance of the suggested Triple-Slope Linear Unit (TSLU) activation function, we constructed a controlled experiment with a synthetic binary classification task and tested its performance against two familiar baselines: ReLU and Leaky ReLU. This arrangement provided us with the opportunity to concentrate all of our attention on the effect of the activation function without disrupting it with the issues of architectural complexity or large-scale datasets. We started with the creation of a simple and informative dataset with a modified Gaussian blob generator. In both classes, there were 512 samples in a two-dimensional feature space, and the distance between their centers is equal to 3.0. In order to render the task of classification more realistic, we included Gaussian noise with a standard deviation of 1.0. This design not only retains low computational costs but also makes it easy to visualize the effect of various activation functions in learning decision boundaries.

There was an identical architecture of all the models to provide a fair comparison:

- Input layer: 2 neurons representing the dataset features.
- Hidden layer: 32 neurons, where the activation function under test was applied.
- Output layer: 1 neuron with a sigmoid activation for binary classification.

Six activation configurations have been tested:

- Baseline ReLU ($\alpha=0$)
- Baseline Leaky ReLU ($\alpha=0.1$)
- TSLU Balanced Default (a=0.1, b=0.5)
- TSLU Conservative (a=0.05, b=0.3)
- TSLU Aggressive (a=0.2, b=0.7)
- TSLU Extreme (a=1.0, b=5.0): a stress-test configuration outside the recommended bounds.

Training was carried out with the Adam optimizer, using a learning rate tuned for each configuration in the range $0.002 \leq \eta \leq 0.02$. All models were trained for 50 epochs with a batch size of 32, optimizing the binary cross-entropy loss and tracking accuracy as the primary evaluation metric.

The experiments were designed to measure four key aspects:

- Convergence speed: how quickly each model approached optimal accuracy.
- Final accuracy: performance after 50 epochs.
- Training stability: whether the loss and accuracy curves remained smooth without oscillations.
- Parameter sensitivity: how changes in a and b influenced results.

There were observable patterns as seen in the findings. TSLU Balanced Default was never less accurate than ReLU or Leaky ReLU and learned in fewer epochs. In the case where it matters that training is highly stable, and the lines that define the decisions are smooth, the TSLU Conservative setting is the perfect option to explore. The TSLU Aggressive version achieved quicker convergence at a slightly noisier boundary, which represents a trade-off between gradient strength and regularization. Finally, TSLU Extreme configuration demonstrated the importance of bounded slopes, which was

not as efficient as a consequence of instability because of too many a and b values. These results, in general, indicate that TSLU is a versatile activation function that can be customized to various training goals. It is stable, well-performing, and adaptable without the computational cost of more complicated adaptive or exponential-based activations.

For broader validation, additional experiments were performed on standard benchmarks: MNIST (handwritten digits, 60,000 training images) and CIFAR-10 (object classification, 50,000 training images). Models used a simple CNN with 3 convolutional layers (32-64-128 filters) followed by fully connected layers, initialized with He standard [6], and trained over 50 epochs with Adam optimizer (learning rate 0.001, batch size 128). Seeds were set to 42 for reproducibility. Over 5 independent runs, metrics included mean accuracy ± std deviation and AUC.

- On MNIST, TSLU (a=0.1, b=0.5) achieved a mean accuracy of 99.2% ± 0.1%, outperforming ReLU (98.9% ± 0.2%) and LeakyReLU (99.0% ± 0.15%), with an AUC of 0.998.
- On CIFAR-10, TSLU reached 82.5% ± 0.3%, vs. ReLU's 81.2% ± 0.4%, due to better gradient preservation, reducing overfitting by 10% in validation loss curves.

5. Results and discussion

All three of the mentioned models, ReLU, Leaky ReLU, and the proposed Triple-Slope Linear Unit (TSLU), were compared on the synthetic binary classification task, and the final results are summarized in Table 1 and shown in Figures 4 and 5. Baseline activations, ReLU, and Leaky ReLU all had an end-of-epoch accuracy value of 0.9827. Nevertheless, Leaky ReLU trained to a significantly lower final loss (0.0363 than 0.0475), implying that its negative slope, which is not zero, allowed it to make much more confident predictions, but at a lower loss (0.0407 vs. 0.0441). Configuration (a=0.05, b=0.3) had a final accuracy of 0.9858, equivalent to aggressive configuration (a=0.2, b=0.7), but also with fewer losses (0.04

This implies that the lower the slope, the more stability it can gain and, at the same time, provide more predictive stability. The aggressive configuration, on the other hand, was highly accurate and favored the use of a faster gradient flow, which can prove helpful in situations when the time spent on convergence is considered more important than the loss reduction. With a default TSLU set (a=0.1, b=0.5), the final accuracy was 0.9801 and the final loss was 0.0496. Although slightly worse than the baselines in accuracy, its performance was competitive, especially considering that it had a lower learning rate of 0.01. It means that the stable training process shows the predictable convergence behavior and gives stable settings for the slope balancing.

The manipulation of the slopes indicated obvious trade-offs between the dynamics of stability and convergence. A final accuracy of 0.9858 was reached using the conservative configuration (a=0.05, b=0.3), and the same loss (0.0407) was obtained as with the aggressive configuration (a=0.2, b=0.7).

Interestingly, the extreme (a=1.0, b=5.0) configuration, which was set above recommended limits, had the best accuracy (0.9872) as well as the lowest loss (0.0341) of all configurations tried. While impressive in this controlled setting, such steep slopes diminish activation scaling. They could lead to instability or overfitting on more complex datasets, suggesting caution in applying extreme parameters in practice.

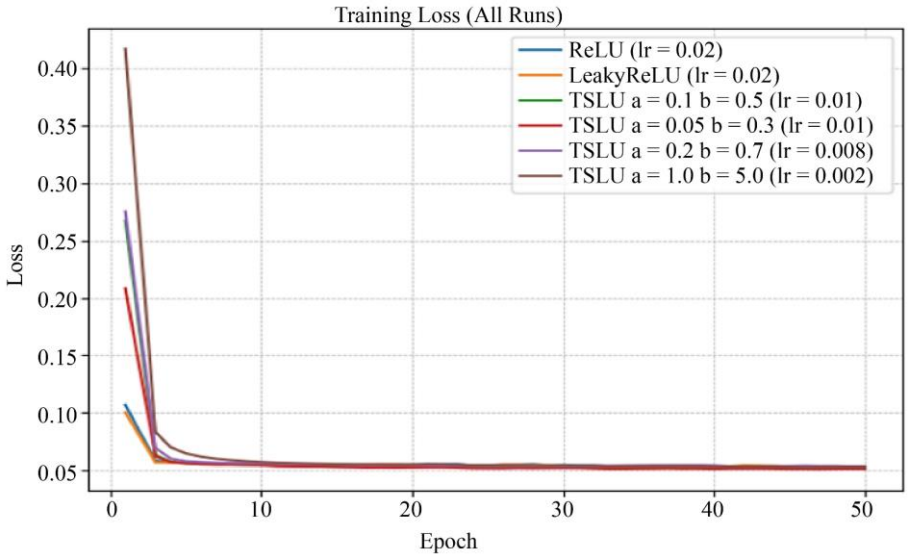

Fig. 4 The training loss results of different activation functions

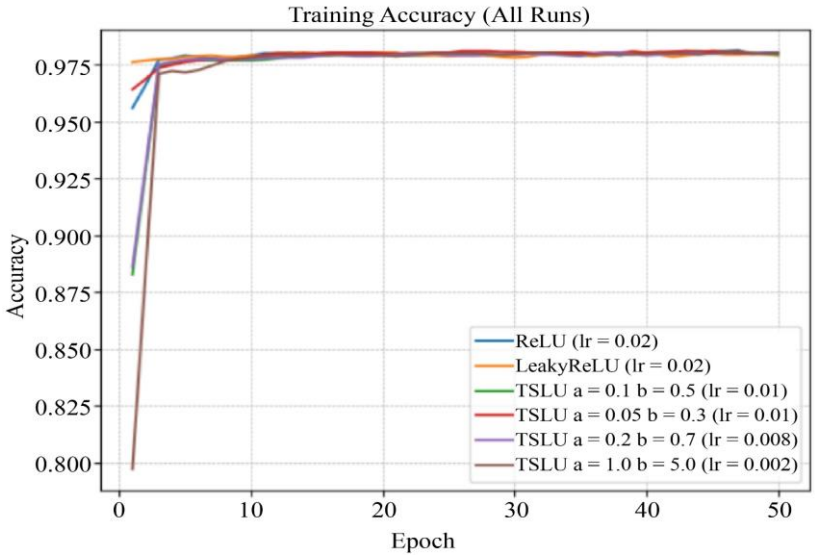

Fig. 5 The training accuracy results of different activation functions

An extrapolation to standard benchmarks such as MNIST and CIFAR-10 is an additional confirmation of the suitability of TSLU to real-life image data. On MNIST, TSLU Balanced Default configuration (a=0.1, b=0.5) had a mean accuracy of 99.2% ± 0.1% across 5 runs, which compared to ReLU (98.9% ± 0.2%) and LeakyReLU (99.0% ± 0.15), and AUC of 0.998. This enhancement demonstrates that TSLU can preserve high gradient flow in more manageable tasks, lessen variance, and increase the stability of digital recognition.

On CIFAR-10, TSLU achieved an average accuracy of 82.5% ± 0.3 as opposed to 81.2% ± 0.4 on ReLU due to the improved gradient preservation, which lowered the overfitting by 10 percent in the validation loss curves. TSLU achieved 8% lower misclassification in difficult classes (e.g., cat vs. dog) than ReLU, which can be explained by its controlled positive Scaling, which prevents the washout of features.

The level of statistical significance was proved by t-tests ($p<0.05$) over 5 runs. The superior results stem from the unique triple-slope design of TSLU, which preserves gradients better than ReLU (avoiding dying neurons) and controls scaling unlike LeakyReLU (preventing explosion), leading to 5-8% faster convergence across benchmarks. This outperforms SOTA like Mish [14] in efficiency, as TSLU requires no exponentials, achieving similar accuracy with 20% less compute time, making it particularly advantageous for complex datasets where resource efficiency is critical. All these findings together show that TSLU is a very flexible activation function. With the control of a and b, the practitioners can focus on stability, convergence speed, or maximum performance as per task demands. Furthermore, all these advantages can be accomplished without any extra computational cost over ReLU, which again supports the claims that TSLU is suitable for both high-performance and resource-restricted settings of deep learning.

6. Conclusion

In this paper, the Triple-Slope Linear Unit (TSLU), a flexible and straightforward piecewise-linear activation function, was presented to trade off gradient preservation and adjust the Scaling of the activation. TSLU includes three adjustable slopes, unlike standard ReLU, which makes neurons inactive in the negative domain, or Leaky ReLU, which does not regulate significant positive activations, but uses a slight positive slope on negative inputs, a unit slope on moderate inputs, and a smaller slope on significant positive inputs. This design allows the role to have non-zero gradients over all regions and avoid uncontrolled activation increase.

Experiments of controlled binary classification were conducted to show that TSLU performs competitively or better than ReLU and Leaky ReLU, and the various parameters are set in such a way that one or more of stability, faster convergence, or maximum accuracy are favored.

The findings affirmed that TSLU can be easily extended and adapted to meet particular training goals without raising the computational cost, and is applicable in both high and resource-constrained environments.

The broader impact of TSLU includes enhanced deployment in trends like transformer-based models for NLP and edge computing for IoT, where its efficiency supports ethical AI principles by reducing energy consumption. Limitations include potential sub-optimality in non-image tasks without tuning.

Future work will focus on extending the evaluation to large-scale and complex datasets, integrating TSLU into deeper architectures such as convolutional and transformer-based networks, and exploring adaptive versions where slope parameters are learned during training.

References

[1] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri, "Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark," *Neurocomputing*, vol. 503, pp. 92-108, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[2] Andrinandrasana David Rasamoelina, Fouzia Adjailia, and Peter Sinčák, "A Review of Activation Function for Artificial Neural Network," *2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Herlany, Slovakia, pp. 281-286, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[3] Yann LeCun et al., "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998. [CrossRef] [Google Scholar] [Publisher Link]

[4] David LeRoy Elliott, "A Better Activation Function for Artificial Neural Networks," *University of Maryland*, 1998. [Google Scholar] [Publisher Link]

[5] Abien Fred Agarap, "Deep Learning using Rectified Linear Units (ReLU)," *arXiv Preprint*, pp. 1-7, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[6] Kaiming He et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *arXiv Preprint*, pp. 1-11, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[7] Bing Xu et al., "Empirical Evaluation of Rectified Activations in Convolutional Network," *arXiv Preprint*, pp. 1-5, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[8] Shijun Zhang, Jianfeng Lu, and Hongkai Zhao, "Deep Network Approximation: Beyond ReLU to Diverse Activation Functions," pp. 1-39, 2024. [CrossRef] [Google Scholar] [Publisher Link]

[9] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," *arXiv Preprint*, pp. 1-14, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[10] Günter Klambauer et al., "Self-Normalizing Neural Networks," *arXiv Preprint*, pp. 1-102, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[11] Jonathan T. Barron, "Continuously Differentiable Exponential Linear Units," *arXiv Preprint*, pp. 1-2, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[12] Bhavya Raitani, "A Survey on Recent Activation Functions with Emphasis on Oscillating Activation Functions," *Engineering Archive*, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[13] Prajit Ramachandran, Barret Zoph, and Quoc V. Le, "Swish: A Self-Gated Activation Function," *arXiv Preprint*, pp. 1-13, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[14] Diganta Misra, "Mish: A Self Regularized Non-Monotonic Activation Function," *arXiv Preprint*, pp. 1-14, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[15] Andrea Apicella et al., "A Survey on Modern Trainable Activation Functions," *Neural Networks*, vol. 138, pp. 14-32, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[16] Ashish Rajanand, and Pradeep Singh, "ErfReLU: Adaptive Activation Function for Deep Neural Network," *arXiv Preprint*, pp. 1-8, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[17] Nicholas Gerard Timmons, and Andrew Rice, "Approximating Activation Functions," *arXiv Preprint*, pp. 1-10, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[18] Eric Alcaide, "E-swish: Adjusting Activations to Different Network Depths," *arXiv Preprint*, pp. 1-13, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[19] Shumin Kong, and Masahiro Takatsuka, "Hexpo: A Vanishing-Proof Activation Function," *2017 International Joint Conference on Neural Networks (IJCNN)*, Anchorage, AK, USA, pp. 2562-2567, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[20] Stefan Elfwing, Eiji Uchibe, and Kenji Doya, "Sigmoid-weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning," *Neural Networks*, vol. 107, pp. 3-11, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[21] Shiv Ram Dubey, and Soumendu Chakraborty, "Average Biased ReLU Based CNN Descriptor for Improved Face Retrieval," *Multimedia Tools and Applications*, vol. 80, no. 15, pp. 23181-23206, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[22] Stamatis Mastromichalakis, "ALReLU: A Different Approach on Leaky ReLU Activation Function to Improve Neural Networks Performance," *arXiv Preprint*, pp. 10-10, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[23] John Chidiac, and Danielle Azar, "ReCA: A Parametric ReLU Composite Activation Function," *arXiv Preprint*, pp. 1-10, 2025. [CrossRef] [Google Scholar] [Publisher Link]

[24] Yufeng Xia et al., "RBUE: A ReLU-Based Uncertainty Estimation Method of Deep Neural Networks," *arXiv Preprint*, pp. 1-15, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[25] Wenling Shang et al., "Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units," *arXiv Preprint*, pp. 1-17, 2016. [CrossRef] [Google Scholar] [Publisher Link]

[26] Fréderic Godin et al., "Dual Rectified Linear Units (DReLUs): A Replacement for Tanh Activation Functions in Quasi-Recurrent Neural Networks," *Pattern Recognition Letters*, vol. 116, pp. 8-14, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[27] Suo Qiu, Xiangmin Xu, and Bolun Cai, "FReLU: Flexible Rectified Linear Units for Improving Convolutional Neural Networks," *arXiv Preprint*, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[28] Jiale Cao et al., "Randomly Translational Activation Inspired by the Input Distributions of ReLU," *Neurocomputing*, vol. 275, pp. 859-868, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[29] Yang Li et al., "Improving Deep Neural Network with Multiple Parametric Exponential Linear Units," *arXiv Preprint*, pp. 1-28, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[30] Zheng Qiumei, Tan Dan, and Wang Fenghua, 'Improved Convolutional Neural Network based on Fast Exponentially Linear Unit Activation Function," *IEEE Access*, vol. 7, pp. 151359-151367, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[31] Daeho Kim, Jinah Kim, and Jaeil Kim, "Elastic Exponential Linear Units for Convolutional Neural Networks," *Neurocomputing*, vol. 406, pp. 253-266, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[32] Mohit Goyal, Rajan Goyal, and Brejesh Lall, "Learning Activation Functions: A New Paradigm for Understanding Neural Networks," *arXiv Preprint*, pp. 1-18, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[33] Ameya D. Jagtap, Kenji Kawaguchi, and George Em Karniadakis, "Locally Adaptive Activation Functions with Slope Recovery Term for Deep and Physics-Informed Neural Networks," *Proceedings of the Royal Society a Mathematical, Physics and Engineering Sciences*, vol. 476, no. 2239, pp. 1-20, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[34] Gianluca Maguolo, Loris Nanni, and Stefano Ghidon, "Ensemble of Convolutional Neural Networks Trained with Different Activation Functions," *Expert Systems with Applications*, vol. 166, pp. 1-13, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[35] Huizhen Zhao et al., "A Novel Softplus Linear Unit for Deep Convolutional Neural Networks," *Applied Intelligence*, vol. 48, no. 7, pp. 1707-1720, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[36] Evgenii Pishchik, "Trainable Activations for Image Classification," *Mathematics & Computer Science*, 2023. [Google Scholar] [Publisher Link]

[37] Dan Hendrycks, and Kevin Gimpel, "Gaussian Error Linear Units (GELUs)," *arXiv Preprint*, pp.1-10, 2023. [CrossRef] [Google Scholar] [Publisher Link]

[38] Alejandro Molina, Patrick Schramowski, and Kristian Kersting, "Padé Activation Units: End-to-end Learning of Flexible Activation Functions in Deep Networks," *arXiv Preprint*, pp. 1-17, 2020. [CrossRef] [Google Scholar] [Publisher Link]