mathematics

MDPI

Article

A Periodic Mapping Activation Function: Mathematical Properties and Application in Convolutional Neural Networks

Xu Chen [1,†], Yinlei Cheng [2,†], Siqin Wang [3], Guangliang Sang [3,4,*], Ken Nah [3] and Jianmin Wang [1]

[1] College of Design and Innovation, Tongji University, Shanghai 200092, China; rikaxu@tongji.edu.cn (X.C.); wangjianmin@tongji.edu.cn (J.W.)
[2] School of Artificial Intelligence and Innovative Design, Beijing Institute of Fashion Technology, Beijing 100029, China; 202320001086@stu.bift.edu.cn
[3] International Design Trend Center, Hongik University, Seoul 04068, Republic of Korea; wangsiqin@g.hongik.ac.kr (S.W.); knah@hongik.ac.kr (K.N.)
[4] School of Engineering, Korea National University of Transportation, Chungju 27469, Republic of Korea
* Correspondence: sang@ut.ac.kr
† These authors contributed equally to this work.

Abstract

Activation functions play a crucial role in ensuring training stability, convergence speed, and overall performance in both convolutional and attention-based networks. In this study, we introduce two novel activation functions, each incorporating a sine component and a constraint term. To assess their effectiveness, we replace the activation functions in four representative architectures—VGG16, ResNet50, DenseNet121, and Vision Transformers—covering a spectrum from lightweight to high-capacity models. We conduct extensive evaluations on four benchmark datasets (CIFAR-10, CIFAR-100, MNIST, and Fashion-MNIST), comparing our methods against seven widely used activation functions. The results consistently demonstrate that our proposed functions achieve superior performance across all tested models and datasets. From a design application perspective, the proposed functional periodic structure also facilitates rich and structurally stable activation visualizations, enabling designers to trace model attention, detect surface biases early, and make informed aesthetic or accessibility decisions during interface prototyping.

Keywords: activation function; convolutional neural networks; vision transformers; benchmark datasets

MSC: 68T07

check for updates

Academic Editors: Zne-Jung Lee and Liang-Hung Wang

Received: 7 July 2025
Revised: 29 July 2025
Accepted: 15 August 2025
Published: 3 September 2025

Citation: Chen, X.; Cheng, Y.; Wang, S.; Sang, G.; Nah, K.; Wang, J. A Periodic Mapping Activation Function: Mathematical Properties and Application in Convolutional Neural Networks. *Mathematics* **2025**, *13*, 2843. https://doi.org/10.3390/math13172843

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Activation functions play a pivotal role in shaping a neural network's convergence behavior, convergence rate, and learning dynamics. They enable hierarchical feature abstraction, enhance representational capacity, mitigate gradient explosion, and establish nonlinear decision boundaries, thereby increasing model robustness. Equally important for human-centered design workflows is that activation functions determine the extent to which intermediate representations can be inspected and interpreted. This affects how designers debug interfaces, justify algorithmic choices to stakeholders, and create transparent user experiences. In this work, we focus on convolutional neural networks (CNNs)—originally developed by LeCun et al. [1]—which represent a major derivative of deep neural networks (DNNs) and have been widely applied to image classification,

semantic segmentation, multimodal learning, and object detection [2]. Recent user experience research indicates that nuanced interface affordances are strong predictors of sustainable technology adoption, suggesting that the interpretability of activation levels may likewise influence stakeholder acceptance [3]. Based on their mapping properties, activation functions are classified into the following three categories: linear, nonlinear monotonic, and nonlinear non-monotonic [4,5]. Given the limited expressive power of linear activations, nonlinear monotonic functions such as the sigmoid [6] and rectified linear unit (ReLU) were introduced [7,8]. Although the sigmoid [9] was extensively used in early networks, it suffers from vanishing gradients [10]; ReLU alleviates this by zeroing negative inputs [11,12] but may discard valuable information in the negative domain [13,14]. To address these issues, the Leaky ReLU maintains a small [15], nonzero slope for negative inputs, while the exponential linear unit (ELU) applies an exponential mapping to negative values to preserve a small yet nonzero gradient [16].

Non-monotonic, nonlinear activation functions have recently garnered significant attention owing to their superior generalization across diverse network architectures and complex datasets. Prominent examples include EELU [17], GELU [18], and Mish [19], while the newly introduced log-Softplus ERror (SERF) has demonstrated notable performance gains attributable to its non-monotonic region design. These functions are characterized by unbounded upper limits, bounded lower limits, global smooth differentiability, and intrinsic non-monotonicity, facilitating more efficient information propagation through deep networks and thereby enhancing overall model performance. In a systematic evaluation on CIFAR-10 [20], CIFAR-100 [21], CINIC-10 [22], and ImageNet [23], Vargas et al. [24] compared two novel activation functions against established baselines, reporting improvements in both classification accuracy and convergence speed. Similarly, Hu et al. [25] proposed the parameter-adaptive AReLU activation, which dynamically adjusts its parameters during training; their experiments on PASCAL VOC [26] and COCO [27] demonstrated that AReLU consistently elevates object detection and segmentation performance across multiple tasks.

Dubey et al. [28] conducted a comprehensive review of activation functions, examining their effectiveness across five neural network families and offering insights for future research directions. Building on this groundwork, our study extends beyond conventional comparative analyses by systematically testing widely used activation functions on various benchmark datasets and representative architectures, while further contributing an innovative periodicity-based design. In particular, we introduce two novel periodic activation functions specifically designed for image classification, aiming to strengthen feature extraction and boost predictive accuracy through architectural refinement. This approach leverages the periodicity of PSR/LPSR as controllable parameters for generative art design, motion graphics, or textile pattern creation. The more stable activations also make lightweight models feasible on edge devices, enabling energy-efficient or accessibility-oriented products [29]. The proposed approach is based on the following two core principles: (1) Mapping network weights to periodic functions—The oscillatory nature and bounded characteristics of periodic functions expand the representational capacity of neural networks and help alleviate training challenges such as vanishing or exploding gradients. (2) Embedding regularization constraints within the periodic functions—During optimization, certain weight parameters are constrained, promoting the learning of informative features while suppressing redundant representations, ultimately improving generalization. To rigorously assess the proposed functions, comprehensive experiments were conducted on four publicly available image classification datasets, employing the following four representative deep learning architectures: VGG-16, ResNet-50, DenseNet-121, and Vision Transformer.

The remainder of this paper is organized as follows. In Section 2, we review advanced activation functions. In Section 3, we propose optimization schemes and two periodic activation functions. In Section 4, we describe the experimental design, comparing the periodic activation functions with previous activation functions. In Section 5, we analyze the experimental results. Finally, in Section 6, we summarize this paper and suggest future research directions.

2. Related Work

This section systematically compares state-of-the-art activation functions, emphasizing their underlying mathematical attributes and operational behavior in neural network contexts. To aid intuitive understanding, Figure 1 depicts the functions and their corresponding derivatives, highlighting their distinct dynamic responses.

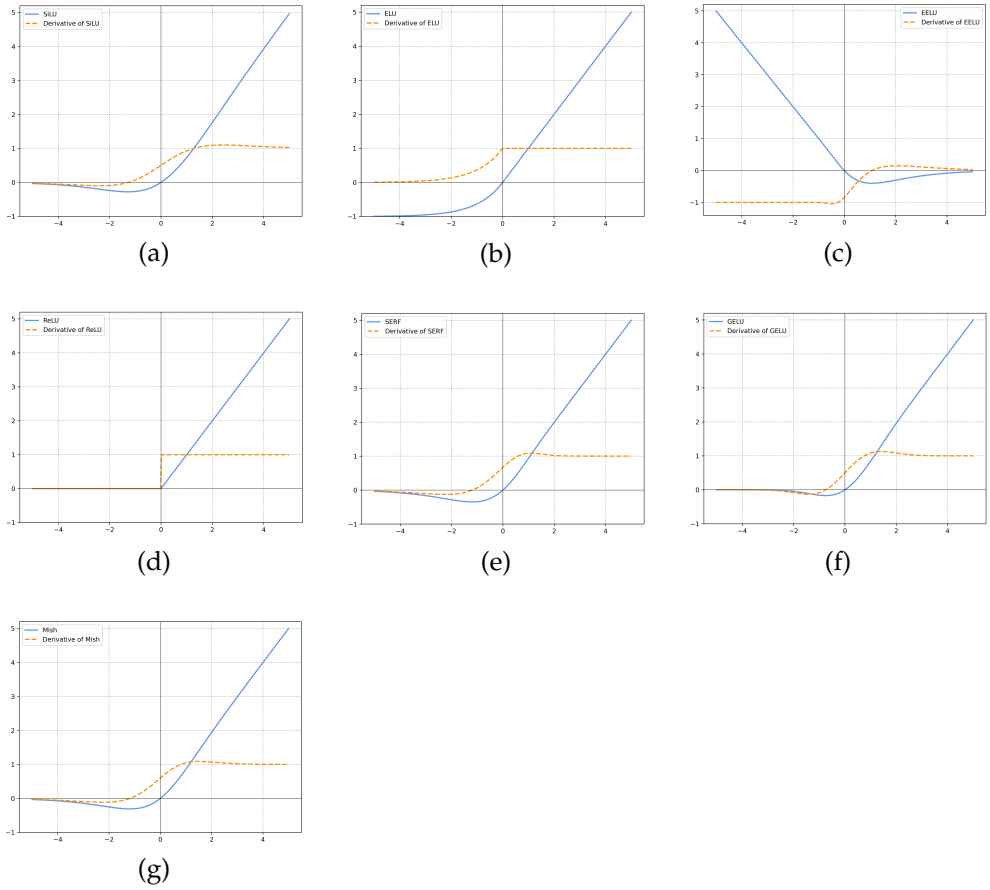


Figure 1. Graphical representations of seven publicly available activation functions. (**a**) SiLU. (**b**) ELU. (**c**) EELU. (**d**) ReLU. (**e**) SERF. (**f**) GELU. (**g**) Mish.

The Sigmoid function was extensively used in early neural networks because of its simple form and easily differentiable nature. However, it presents notable drawbacks in deep network settings. In particular, for inputs with large absolute values, it saturates and produces gradients that are nearly zero. This saturation causes the gradient to diminish progressively during backpropagation, which hinders effective parameter updates in earlier layers—a challenge known as the vanishing gradient problem [30,31]. Due to these well-known drawbacks, the Sigmoid function is not included in the set of activation functions evaluated in the following theoretical discussions and empirical experiments.

2.1. Sigmoid Linear Unit

The Sigmoid Linear Unit (SiLU), introduced by the Google research team, is an activation function characterized by an unbounded upper limit, bounded lower limit, global smoothness, and non-monotonic behavior [32]. Its mathematical expression is

$$f_{SiLU} = x.sigmoid(x) \tag{1}$$

The Sigmoid-Weighted Linear Unit (SiLU) extends the Rectified Linear Unit (ReLU) by introducing smoothness, and has consistently demonstrated enhanced performance over conventional monotonic activations in a range of deep network scenarios [33]. Its main advantage lies in effectively reducing the vanishing gradient problem across a wider spectrum of input values, which supports stable training in deeper layers. Unlike standard sigmoid-based functions, SiLU retains information about both the input's sign and magnitude, enabling it to better capture complex nonlinear features. Moreover, its full differentiability across the entire input domain ensures compatibility with gradient-based optimization algorithms. Although the embedded sigmoid calculation adds slight computational cost, extensive experiments confirm that this is offset by noticeable performance improvements. Owing to these merits, SiLU has become prevalent in modern neural architectures and has shown robust results in visual tasks such as image classification, solidifying its status as a dependable activation function in current deep learning practice [34].

2.2. Exponential Linear Units

The Exponential Linear Unit (ELU), introduced by Clevert et al. [35], was proposed to address the limitations of the Rectified Linear Unit (ReLU) in the negative input domain, where the output is truncated to zero, potentially leading to inactive neurons. The ELU function and its derivative are formally defined as:

$$f_{ELU} = \begin{cases} x, & x > 0 \\ \alpha(e^x - 1), & x \leq 0 \end{cases} \tag{2}$$

$$f'_{\mathrm{ELU}}(x) = \begin{cases} 1, & x > 0 \\ \alpha e^x, & x \leq 0 \end{cases} \tag{3}$$

Here, α is a tunable parameter, commonly set to 1. When the input is positive, the Exponential Linear Unit (ELU) behaves similarly to ReLU by maintaining a linear response, which supports fast convergence during gradient-based training. For negative inputs, it employs an exponential term, allowing the output to asymptotically approach a small negative constant rather than zero. This design ensures the gradient remains non-zero on the negative side, helping to mitigate the "dying neuron" phenomenon often seen with ReLU activations. Furthermore, by generating outputs with a mean closer to zero, ELU reduces internal covariate shift and contributes to more stable training dynamics. Its smooth differentiability and near-zero-centered output facilitate better optimization and help models reach more favorable local minima. However, the exponential computation required for negative values introduces additional cost, which can be a limiting factor for resource-constrained applications, despite its practical performance benefits. Known for improving training stability and maintaining activation outputs closer to zero mean, ELU is particularly useful in applications such as medical image segmentation [36].

2.3. Exponential Error Linear Unit

The Exponential Error Linear Unit (EELU) is an activation function that integrates the Gaussian error function erf with an exponential decay component e^{-x}, aiming to enhance the nonlinear representation capacity and generalization performance of deep neural networks. This function and its derivative are formally defined as follows:

$$f_{EELU} = -x.erf(e^{-x}) \tag{4}$$

$$f'_{\mathrm{EELU}}(x) = -\operatorname{erf}(e^{-x}) + \frac{2x}{\sqrt{\pi}} e^{-e^{-2x}} \cdot e^{-x} \tag{5}$$

EELU exhibits characteristic nonlinear behavior with a bounded lower limit and an unbounded upper limit [37]. The bounded lower region contributes to regularization and helps mitigate overfitting, while the unbounded upper region prevents saturation, thereby alleviating gradient vanishing and promoting efficient gradient flow during training. In the positive input domain ($x > 0$), EELU presents a non-monotonic peak, improving the network's ability to capture complex nonlinear patterns. In the negative domain ($x < 0$), unlike ReLU which truncates all negative values to zero, EELU retains partial gradient flow through the combination of the error and exponential terms, thus effectively avoiding the "dying neuron" phenomenon. Moreover, the function is continuous and differentiable across its entire domain, ensuring smooth optimization and compatibility with gradient-based learning algorithms.

In contrast to traditional activation functions like ReLU and ELU, EELU provides distinct benefits including smoother gradients, a broader dynamic response range, and inherent non-monotonicity. These features jointly contribute to more consistent gradient flow, accelerated convergence, and a stronger ability to avoid poor local minima, ultimately improving the training effectiveness and performance of deep neural networks. As an extended version of ELU, EELU provides enhanced expressive power in the negative input range, and is applicable in complex tasks such as video understanding and graph neural networks.

2.4. Rectified Linear Unit

The Rectified Linear Unit (ReLU) [38] has emerged as one of the most prevalent activation functions in deep learning due to its ability to introduce nonlinearity and suppress irrelevant information during inference. Moreover, ReLU contributes to sparsity in the activation outputs, which can improve computational efficiency in neural network inference. However, this sparsity also introduces the risk of neuron inactivity, particularly in deep or large-scale architectures, where a significant proportion of neurons may remain non-responsive [39]. Prior studies have reported that up to 40% of neurons can become permanently inactive in such settings [40]. Although methods like residual connections have been introduced to address this limitation, ReLU's performance can still deteriorate with increasing network depth. This is primarily attributed to its zero response for all negative inputs, which disrupts gradient propagation during backpropagation and hampers the optimization process. As a result, the network's ability to learn representations involving negative activations becomes constrained [41,42]. The ReLU function and its derivative are mathematically defined as:

$$f_{ReLU} = max(0, x) \tag{6}$$

$$f'_{\mathrm{ReLU}}(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{7}$$

Due to its computational efficiency and strong capability to mitigate the vanishing gradient problem, ReLU is widely adopted in convolutional neural networks (CNNs) for image classification and object detection tasks, such as in AlexNet, VGG, and ResNet architectures [43].

2.5. Log-Softplus Error Activation Function

Inspired by the design of the Mish activation function, Sayan Nag et al. [44] proposed a novel activation function known as SERF, whose mathematical definition and derivative are given by:

$$f_{SERF} = x.erf(ln(1 + e^x)) \tag{8}$$

$$f'_{\text{SERF}}(x) = \text{erf}(\ln(1 + e^x)) + \frac{2x}{\sqrt{\pi}} \cdot e^{-(\ln(1+e^x))^2} \cdot \frac{e^x}{1 + e^x} \tag{9}$$

SERF is characterized by a bounded lower limit and an unbounded upper limit. In contrast to traditional activation functions such as tanh and sigmoid—which are constrained both above and below and often require initialization in their linear operating regions to avoid early saturation—SERF alleviates such limitations by design. Saturation issues, typically caused by vanishing gradients, hinder effective learning, especially in deeper networks [45]. ReLU, with its unbounded upper range, mitigates this problem and has inspired several subsequent activation functions such as GELU, Swish, and Mish. SERF inherits this advantage, exhibiting a nearly linear response in the positive domain, thereby enabling improved representational capacity and making it a strong candidate for deep neural architectures.

A lower bound in activation functions plays a critical role in promoting regularization. The "dying ReLU" phenomenon—where neurons become inactive—frequently arises from overly large learning rates or substantial negative biases. SERF addresses this issue by preserving a small degree of negative output, which not only enhances the network's expressivity but also facilitates stable gradient flow during backpropagation.

The SERF activation possesses several desirable mathematical characteristics, as follows: it is smooth, non-monotonic, and fully differentiable, which promotes stable gradient propagation throughout training. Its formulation incorporates a self-modulating mechanism—similar in spirit to Swish and Mish—where the input is scaled by a nonlinear function of itself.

Notably, SERF permits limited transmission of negative outputs, ensuring that gradients remain non-zero even when inputs are negative. This mitigates neuron inactivation and enhances the model's ability to capture richer representations. In summary, SERF achieves a practical trade-off between computational cost and learning effectiveness, making it a robust choice for deep neural architectures [46]. As a newly proposed periodic activation function, SERF demonstrates enhanced gradient stability and noise robustness, making it suitable for tasks in video analysis and multimodal perception, while expanding the model's representational capacity [44].

2.6. Gaussian Error Linear Unit

Dan et al. [18] proposed the Gaussian Error Linear Unit (GELU), an activation function for neural networks that is based on the Gaussian error function and is known for its high performance. The GELU function has a non-zero gradient in the negative value region, thus avoiding the "dying neuron" problem. Furthermore, GELU is smoother around zero compared to ReLU, making it easier to converge during training. It is worth noting that

the computation of GELU is more complex, and therefore requires more computational resources. Its mathematical expression is as follows:

$$GELU(x) = x.P(X \leq x) = x.\Phi(x) \tag{10}$$

where $\Phi(x)$ represents the cumulative distribution function of the standard normal distribution, i.e.,

$$\Phi(x) = \frac{1}{2}.(1 + erf(\frac{x}{\sqrt{2}})) \tag{11}$$

$erf(x)$ is the Gaussian error function. This function can be further expressed as follows:

$$x.P(X \leq x) = x \int_{-\infty}^{x} \frac{e. - \frac{(X-\mu)^2}{2\sigma^2}}{\sqrt{2\pi}\sigma} dX \tag{12}$$

where μ and σ represent the mean and standard deviation of the normal distribution, respectively. Since this function is not directly computable, researchers have found that the GELU function can be approximately represented as

$$f_{GELU} = 0.5x(1 + tanh\left[\sqrt{\frac{2}{\pi}}(x + 0.044715x^3)\right]) \tag{13}$$

and its derivative is given by

$$f'_{\text{GELU}}(x) = 0.5(1 + \tanh(g(x))) + 0.5x\Big(1 - \tanh^2(g(x))\Big) \cdot g'(x) \tag{14}$$

where

$$g(x) = \sqrt{\frac{2}{\pi}}\Big(x + 0.044715x^3\Big) \tag{15}$$

$$g'(x) = \sqrt{\frac{2}{\pi}}\Big(1 + 0.134145x^2\Big) \tag{16}$$

where $\sqrt{2/\pi}$ and 0.044715 are two adjustment coefficients of the GELU function.

The Gaussian Error Linear Unit (GELU) implements a smooth, magnitude-aware activation approach, differing from the binary gating characteristic of ReLU. Unlike convex and monotonic functions such as ReLU and ELU, GELU is inherently non-convex and non-monotonic, maintaining nonlinearity throughout its entire domain, which enhances its expressive capacity. Its continuous curvature and gentle modulation provide more precise control over activations, allowing the network to model complex relationships more effectively. Furthermore, GELU can be interpreted probabilistically as the expected output under a stochastic regularization scheme, lending it both theoretical robustness and improved generalization in deep architectures. This activation function has shown great performance in natural language processing (NLP) models such as BERT and other Transformer-based architectures, where its smooth probabilistic nature improves both generalization and convergence speed [47].

2.7. Mish

Misra et al. [19] proposed the Mish activation function, a novel self-regularizing, non-monotonic activation function that is smooth and continuous. This function and its derivative are mathematically defined as follows:

$$f_{Mish} = x.tanh(softplus(x)) = x.tanh(ln(1 + e^x)) \tag{17}$$

$$f'_{\text{Mish}}(x) = \tanh(\ln(1+e^x)) + x \cdot \left(1 - \tanh^2(\ln(1+e^x))\right) \cdot \frac{e^x}{1+e^x} \tag{18}$$

The Mish activation function, inspired by Swish, leverages a self-gating mechanism in which the input is modulated by its own nonlinear transformation. This approach enables Mish to retain a small amount of negative activation, thereby avoiding the conditions that typically lead to the "dying ReLU" phenomenon. By allowing negative information to pass through, Mish enhances the network's representational power and supports more effective information flow across layers.

Another key strength of Mish is its mathematical smoothness. Unlike ReLU, which introduces a discontinuity at zero, Mish is continuously differentiable across its entire input range. This smoothness eliminates potential optimization singularities and ensures stable gradient propagation. Recognized for its non-monotonicity and self-regularizing behavior, Mish is employed in object detection models like YOLOv4 and generative tasks, helping improve model robustness and accuracy [48].

3. Proposed Activation Functions

In this section, we present a novel design framework that leverages periodic activation functions and introduces two intermittent periodic variants, whose functional curves are illustrated in Figure 2. This framework integrates periodic mappings with a constraint mechanism to enhance model prediction accuracy. Specifically, it maps network weights onto periodic functions, exploiting their oscillatory nature to enrich the network's representational capacity. The periodic function T encompasses both intermittent and continuous periodic forms. Compared to linear activations, periodic functions provide a wider functional space while naturally constraining values, thereby alleviating issues such as gradient explosion and vanishing gradients. Furthermore, a Restraint Device (RD) is embedded within the periodic mapping—aligned with standard activation behaviors—to restrict certain weights or parameters during training. This prevents the network from capturing redundant components and helps concentrate learning on relevant features, reducing the impact of noise and improving overall generalization.

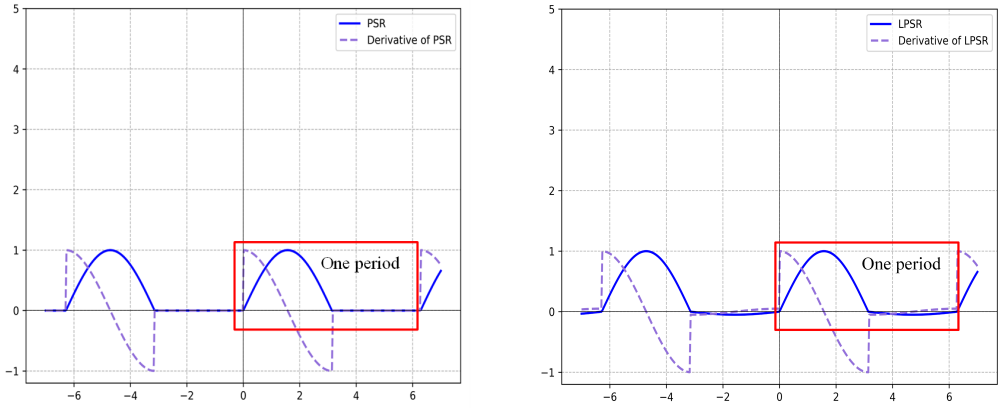


Figure 2. Function plots of the proposed periodic activation functions: PSR and LPSR.

3.1. Periodic Sine Rectifier (PSR)

For a long time, activation functions used in neural networks have mainly been unconstrained functions. While this allows for better positive correlation during the training process, unconstrained activation functions often lead to issues like gradient explosion or gradient vanishing. In this work, we propose a method to constrain the values of infinite numbers within a specific range. Prior to this, the sigmoid activation function also offered a similar solution, but its excessively steep curve led to highly imbalanced value distributions.

Values near the regions close to 0 or 1 were compressed into either $\lim_{n\to-\infty} sigmoid(x) = 0$ or $\lim_{n\to+\infty} sigmoid(x) = 1$, limiting its use to only the final layer of the model. In contrast, using a periodic activation function can ensure a more balanced distribution of parameters with positive correlations. The core idea of this function is to map the parameters we aim to learn periodically to the function's period, while maintaining sufficient simplicity to avoid additional computational overhead during training, especially for deep networks. The function consists of the following two important parts: (1) It maps the positive parameters related to the predicted class onto the waveform of the function. The continuity of the function effectively alleviates the gradient explosion problem commonly encountered in linearly growing functions. Additionally, the smooth transition in the region near zero helps reduce the risk of gradient vanishing. (2) For the noise data we do not wish to learn, ReLU is used as a RD, filtering negative values directly through a regularization method. However, using ReLU as a RD may lead to the "dead neuron" issue mentioned earlier. To address this, we add a very small constant value β, to the negative-correlated data constrained to zero by ReLU, ensuring that the neuron remains active in subsequent calculations [49]. Moreover, the periodic activation function can map multiple classes of classification to the same level during the classification process. In deep neural networks, the vanishing gradient problem refers to the phenomenon where gradients propagated backward through layers become progressively smaller, especially when the derivative of the activation function approaches zero. This can lead to extremely small weight updates in early layers during training, effectively stalling learning in those layers and degrading the performance of deep architectures. The proposed PSR activation function addresses this issue by employing a piecewise sinusoidal form, Mathematically, the periodic mapping activation function is defined as follows:

$$f_{PSR} = RD_1(s(x)) + \epsilon \tag{19}$$

where $RD_1 = ReLU$, $\epsilon = 1 \times 10^{-10}$, $s(x) = sin(x)$ and $\alpha = cos(x)$. This piecewise function is represented depending on the sign of $s(x)$ as follows:

$$f_{PSR} = \begin{cases} s(x) + \epsilon, & s(x) > 0 \\ \epsilon, & s(x) \leq 0 \end{cases} \tag{20}$$

Its derivative is as follows:

$$f'_{PSR} = \begin{cases} \alpha, & s(x) > 0 \\ 0, & s(x) \leq 0 \end{cases} \tag{21}$$

At $s(x) = 0$ (i.e., $x = k\pi, k \in Z$), the function is non-differentiable at these points. A very small constant term, 1×10^{-10}, is added to prevent gradient vanishing during numerical calculations.

Nonlinearity: PSR exhibits nonlinear mapping in the region where $s(x) > 0$, retaining the characteristics of nonlinear activation, which helps the network capture complex feature relationships.

Introduction of periodic mapping: Standard ReLU performs linear mapping for non-negative inputs, and negative inputs are truncated to zero. This monotonic and non-periodic characteristic may have limitations in tasks requiring periodic features. However, since $s(x)$ is a periodic function, PSR inherits this characteristic. This periodicity allows the activation function to repeatedly map feature patterns within the input feature space, helping the neural network learn data features with repetitive or periodic patterns.

Smoothness: $s(x)$ is a smooth and differentiable function, which ensures that within the interval $s(x) > 0$, the gradient does not exhibit discontinuities like those of ReLU, thereby facilitating the stable propagation of gradients.

PSR, by incorporating the periodic characteristics of the sine function and the linear mapping features of ReLU, provides a novel and expressive activation mechanism. It shows significant advantages in both gradient propagation and feature expression capabilities.

3.2. Leaky Periodic Sine Rectifier (LPSR)

The design of the leaky periodic sine rectifier was initially motivated by the need to address neuron deactivation. Although we proposed a method within the periodic activation function that adds an infinitesimally small constant to weights approaching zero, we did not intend for this limiting constant to constrain the effective learning parameters during subsequent computations. Therefore, we designed an extremely small negative domain interval to accommodate outlier weights. Experimental results have confirmed that this approach is indeed effective. However, the size of the negative domain must be carefully controlled, as excessive enlargement may lead to the model overfitting to the parameters in the negative domain. After empirical adjustments, we set the maximum value of the negative domain to 0.05. This small value effectively mitigates neuron deactivation while compressing negatively correlated data into the negative domain, enabling the model to learn parameters more comprehensively. The LPSR controls the slope of the negative activation value, with a default value of 1×10^{-2}. The mathematical definition of this activation function is as follows:

$$f_{LPSR} = RD_2(g(x)) + \epsilon \tag{22}$$

where $RD_2 = LeakyReLU$, $\epsilon = 1 \times 10^{-10}$, $g(x) = sin(x)$, $\varphi = cos(x)$ and $\omega = 0.05$. This piecewise function is represented based on the sign of $T2$ as follows:

$$f_{LPSR} = \begin{cases} g(x) + \epsilon, & g(x) > 0 \\ \omega g(x) + \epsilon, & g(x) \leq 0 \end{cases} \tag{23}$$

its derivative is as follows:

$$f'_{LPSR} = \begin{cases} \varphi, & g(x) > 0 \\ \omega\varphi, & g(x) \leq 0 \end{cases} \tag{24}$$

The constraint is implemented using the LeakyReLU activation function, with the periodic function defined as $g(x)$. At $g(x) = 0$ (i.e., $x = k\pi, k \in Z$), the derivative of this function is discontinuous, but the function is continuous over its entire domain. Other properties of this function are similar to those of PSR.

The proposed PSR and LPSR activation functions exhibit derivative discontinuities at certain transition points, but our analysis and experiments show that these have minimal impact on the training process. Mainstream optimization algorithms such as SGD and Adam are robust to local gradient variations, and neural networks can adaptively adjust parameters to effectively handle minor non-smoothness in activation functions.

4. Experiments

4.1. Model

To evaluate the performance of these activation functions, we used four commonly benchmarked datasets, as detailed in Table 1.

Table 1. Datasets used to study the activation functions.

Dataset	Classes	Training Images	Test Images	Image Shape
CIFAR10	10	50,000	10,000	$32 \times 32 \times 3$
CIFAR100	100	50,000	10,000	$32 \times 32 \times 3$
MNIST [50]	10	60,000	10,000	$32 \times 32 \times 1$
Fashion MNIST [51]	10	60,000	10,000	$32 \times 32 \times 1$

The high performance of an activation function on a specific dataset and neural network does not necessarily guarantee optimal performance across other architectures and datasets. To examine the generalization of various neural architectures, we consider a range of networks, from computationally intensive to lightweight, including Vision Transformers (ViTs) [52], DenseNet121 [53], ResNet50 [54], and VGG-16 [55].

This study employs four representative deep learning architectures to evaluate the proposed activation functions. The classical VGG-16 network, containing approximately 138 million trainable parameters, serves as a common baseline in computer vision tasks and applies dropout ($p = 0.5$) before and after its fully connected layers to enhance generalization, with batch normalization used after each activation and softmax producing the final predictions. ResNet-50, a 50-layer residual network with about 25.6 million parameters, alleviates the vanishing gradient problem via identity shortcut connections and balances computational cost with stable and accurate classification performance. DenseNet121, known for its dense inter-layer connectivity, requires only 9 million parameters, making it suitable for computationally constrained environments; it ends with a fully connected layer of 2048 neurons and uses a dropout rate of 0.4 to mitigate overfitting, maintaining consistency with VGG-16's output structure. Finally, the Vision Transformer (ViT) replaces convolutions entirely with a pure Transformer architecture that leverages self-attention to model global dependencies; it has approximately 87.71 million parameters and generally needs large-scale pretraining to achieve state-of-the-art performance on extensive vision benchmarks.

4.2. Experiment Establishment

All experiments in this study were conducted on a high-performance computing server running Ubuntu 22.04 as the operating system. Python 3.10 was used as the main programming language, with PyCharm 2024.1.4 serving as the code editor and compiler. The deep learning framework was configured with CUDA version 11.8 to enable GPU acceleration. In terms of hardware, the server was equipped with an Intel(R) Xeon(R) Gold 5218 CPU and an NVIDIA A100-SXM4 GPU with 40GB of memory. It also featured 256 GB of RAM and a 4.4 TB ROM storage capacity. The server model was UniServer R5300 RS54M2C1S. Due to the fewer convergence cycles and minimized oscillating gradients of PSR and LPSR, they can be deployed in edge-constrained or battery-sensitive scenarios—such as wearable devices for assistive vision. This efficiency aligns with the goals of sustainable user experience (UX), as it reduces energy consumption and enables equitable access on low-cost hardware.

In this study, the proposed activation functions were integrated into the hidden layers of the chosen models by systematically replacing their original activations. The modified models were then trained and validated on four diverse datasets to assess their generalizability. To mitigate the impact of randomness due to weight initialization, each experiment was independently repeated eight times under consistent test set partitions. It is also important to highlight that some of the activation functions employed required the prior specification of certain hyperparameters. The cross-entropy loss function used is shown in Equation (17), expressed as follows:

$$\mathcal{L}_{CE}(x, y) = -\sum_{i=1}^{n} x_i log(y_i) \tag{25}$$

where x_i represents the true label of the i-th element, and y_i denotes the probability that the model predicts x belongs to the i-th class. A smaller cross-entropy loss indicates more accurate predictions by the model.

The training process utilizes a batch size of 64, with model parameters optimized using the Adam algorithm. Adam operates by concurrently estimating the first-order moment (mean) and second-order moment (uncentered variance) of the gradients, employing exponential moving averages and applying bias correction to prevent the gradients from vanishing in the initial training phases [56]. The initial learning rate is set to 1×10^{-3} and is adaptively adjusted throughout training using the CosineAnnealingLR scheduler, which gradually reduces the learning rate to improve convergence efficiency and training stability. Each model is trained for 300 epochs, with weight decay regularization applied at a rate of 0.01 every 20 epochs to reduce overfitting. The training process uses the standard cross-entropy loss function. Considering the classification focus of the tasks, model performance is evaluated using Accuracy, Precision, and Recall, where higher scores reflect better classification outcomes. The formulas for these metrics are as follows:

$$Acc = \frac{TP + TN}{TP + FN + FP + TN} \tag{26}$$

$$Precision = \frac{TP}{TP + FP} \tag{27}$$

$$Recall = \frac{TP}{TP + FN} \tag{28}$$

In the context of evaluation metrics, TP (True Positives) refers to instances correctly identified as belonging to the positive class, while TN (True Negatives) denotes instances accurately classified as negative. FN (False Negatives) are cases where positive samples are misclassified as negative, and FP (False Positives) represent negative samples that are incorrectly labeled as positive. When applied to multi-class classification tasks, macro-averaging is typically favored over micro-averaging. Macro-averaging computes each metric independently for every class and then calculates the unweighted mean across all classes, ensuring equal importance is given to each class regardless of its frequency. In contrast, micro-averaging aggregates the contributions of all classes before computing the overall metric, which may bias results toward classes with more instances. The macro-average formula used is as follows, where n is the number of classes:

$$macro_A = \frac{1}{n} \sum_1^n A_i \tag{29}$$

$$macro_P = \frac{1}{n} \sum_1^n P_i \tag{30}$$

$$macro_R = \frac{1}{n} \sum_1^n R_i \tag{31}$$

To promote generalization and minimize redundant computation, an early stopping strategy is implemented based on validation loss as follows: training is terminated if the validation loss fails to decrease for 50 consecutive epochs. This approach not only mitigates overfitting but also reduces training time. Furthermore, all hyperparameters and training settings are meticulously optimized to ensure effective feature extraction while maintaining computational efficiency.

5. Results

This section presents the experimental results. We evaluated the performance of the activation functions we developed using the MNIST, FashionMNIST, CIFAR10, and CIFAR100 datasets across four models, and we compared them with seven existing state-of-the-art activation functions. Tables 2–5 record the performance of the activation functions on different models for each dataset, with a total of 160 training experiments conducted. The tables display the accuracy, precision, recall, and the standard deviation of accuracy for each dataset and model. The mean values of accuracy, precision, and recall for each model across all datasets are summarized in Table 6. The results shown in Tables 2–6 indicate that the proposed PSR and LPSR consistently outperform other activation functions across all datasets. The best result is highlighted in bold italics, and the second-best result is highlighted in italics.

In contrast, Table 7 presents the performance rankings of each activation function for various model architectures, along with their average ranks. These rankings are calculated by averaging each function's position across multiple runs on all datasets. The last column shows the overall mean rank, aggregated over the two tested architectures.

Table 2. Comparison of evaluation indicators of activation functions on VGG16 with different datasets.

Dataset	Activation	ACC	Std	Precision	Recall
CIFAR10	SiLU	0.843	0.039	0.843	0.841
CIFAR10	ELU	0.812	0.070	0.816	0.809
CIFAR10	EELU	0.839	0.064	0.836	0.829
CIFAR10	ReLU	0.818	0.038	0.818	0.807
CIFAR10	SERF	0.827	0.049	0.829	0.819
CIFAR10	GELU	0.818	0.039	0.819	0.809
CIFAR10	Mish	0.829	0.038	0.828	0.819
CIFAR10	PSR	***0.873***	0.012	*0.844*	*0.852*
CIFAR10	LPSR	*0.864*	0.029	***0.864***	***0.862***
CIFAR100	SiLU	0.589	0.036	0.278	0.279
CIFAR100	ELU	0.576	0.039	0.375	0.374
CIFAR100	EELU	0.448	0.049	0.549	0.538
CIFAR100	ReLU	0.540	0.028	0.541	0.539
CIFAR100	SERF	0.537	0.017	0.530	0.536
CIFAR100	GELU	0.529	0.038	0.527	0.517
CIFAR100	Mish	0.524	0.047	0.527	0.518
CIFAR100	PSR	***0.606***	0.024	***0.586***	***0.587***
CIFAR100	LPSR	*0.596*	0.245	*0.576*	*0.578*
MNIST	SiLU	*0.991*	0.004	*0.989*	*0.989*
MNIST	ELU	0.985	0.023	0.982	0.982
MNIST	EELU	0.975	0.003	0.973	0.972
MNIST	ReLU	0.963	0.003	0.964	0.967
MNIST	SERF	0.973	0.004	0.979	0.972
MNIST	GELU	0.959	0.003	0.954	0.954
MNIST	Mish	0.939	0.035	0.938	0.938
MNIST	PSR	***0.994***	0.003	***0.991***	***0.991***
MNIST	LPSR	0.901	0.270	0.889	0.899
Fashion MNIST	SiLU	0.911	0.018	0.910	0.909
Fashion MNIST	ELU	0.870	0.049	0.871	0.869
Fashion MNIST	EELU	0.924	0.049	*0.928*	0.914
Fashion MNIST	ReLU	0.914	0.041	0.915	0.907
Fashion MNIST	SERF	0.928	0.039	0.927	*0.917*
Fashion MNIST	GELU	0.909	0.049	0.903	0.897
Fashion MNIST	Mish	0.913	0.038	0.918	0.904
Fashion MNIST	PSR	***0.931***	0.009	***0.929***	***0.929***
Fashion MNIST	LPSR	*0.929*	0.402	0.917	0.928

Table 3. Comparison of evaluation indicators of activation functions on ResNet50 with different datasets.

Dataset	Activation	ACC	Std	Precision	Recall
CIFAR10	SiLU	*0.829*	0.015	*0.829*	*0.827*
CIFAR10	ELU	0.824	0.012	0.823	0.822
CIFAR10	EELU	0.827	0.017	0.825	0.819
CIFAR10	ReLU	0.807	0.037	0.802	0.804
CIFAR10	SERF	0.797	0.004	0.798	0.793
CIFAR10	GELU	0.768	0.001	0.769	0.768
CIFAR10	Mish	0.789	0.014	0.784	0.780
CIFAR10	PSR	***0.846***	0.009	***0.846***	***0.845***
CIFAR10	LPSR	0.814	0.020	0.815	0.812
CIFAR100	SiLU	0.535	0.023	0.253	0.254
CIFAR100	ELU	***0.557***	0.015	0.262	0.263
CIFAR100	EELU	0.460	0.028	0.459	0.459
CIFAR100	ReLU	0.528	0.004	***0.529***	***0.527***
CIFAR100	SERF	0.528	0.007	*0.527*	*0.520*
CIFAR100	GELU	0.479	0.008	0.479	0.472
CIFAR100	Mish	0.490	0.039	0.490	0.489
CIFAR100	PSR	*0.538*	0.023	0.253	0.254
CIFAR100	LPSR	0.467	0.032	0.219	0.221
MNIST	SiLU	0.992	0.002	***0.990***	***0.990***
MNIST	ELU	*0.992*	0.002	0.990	0.989
MNIST	EELU	0.987	0.006	0.982	0.986
MNIST	ReLU	0.968	0.003	0.961	0.968
MNIST	SERF	0.939	0.003	0.928	0.929
MNIST	GELU	0.957	0.002	0.958	0.957
MNIST	Mish	0.978	0.003	0.977	0.969
MNIST	PSR	***0.992***	0.002	*0.989*	*0.989*
MNIST	LPSR	0.992	0.003	0.989	0.989
Fashion MNIST	SiLU	0.903	0.008	0.908	0.907
Fashion MNIST	ELU	0.905	0.009	*0.909*	*0.907*
Fashion MNIST	EELU	0.900	0.008	0.901	0.904
Fashion MNIST	ReLU	0.897	0.003	0.897	0.893
Fashion MNIST	SERF	0.849	0.005	0.848	0.849
Fashion MNIST	GELU	0.879	0.007	0.872	0.875
Fashion MNIST	Mish	0.892	0.007	0.896	0.890
Fashion MNIST	PSR	***0.911***	0.008	***0.909***	***0.909***
Fashion MNIST	LPSR	*0.906*	0.010	0.906	0.904

Table 4. Comparison of evaluation indicators of activation functions on DenseNet121 with different datasets.

Dataset	Activation	ACC	Std	Precision	Recall
CIFAR10	SiLU	0.841	0.012	0.839	0.838
CIFAR10	ELU	0.828	0.019	0.828	0.826
CIFAR10	EELU	0.827	0.039	0.829	0.823
CIFAR10	ReLU	0.839	0.038	0.839	0.835
CIFAR10	SERF	0.838	0.063	0.830	0.839
CIFAR10	GELU	*0.851*	0.074	*0.851*	*0.850*
CIFAR10	Mish	0.840	0.038	0.840	0.840
CIFAR10	PSR	***0.859***	0.011	***0.857***	***0.857***
CIFAR10	LPSR	0.842	0.014	0.841	0.839
CIFAR100	SiLU	0.588	0.018	0.277	0.278
CIFAR100	ELU	0.572	0.023	0.269	0.271
CIFAR100	EELU	0.559	0.039	0.559	0.553
CIFAR100	ReLU	0.573	0.040	***0.572***	***0.575***
CIFAR100	SERF	0.568	0.063	0.560	0.562
CIFAR100	GELU	0.578	0.038	*0.570*	*0.573*
CIFAR100	Mish	0.539	0.048	0.533	0.540
CIFAR100	PSR	***0.608***	0.016	0.286	0.288
CIFAR100	LPSR	*0.589*	0.018	0.278	0.279

Table 4. *Cont.*

Dataset	Activation	ACC	Std	Precision	Recall
MNIST	SiLU	0.992	0.002	0.989	0.989
MNIST	ELU	0.991	0.003	0.989	0.989
MNIST	EELU	0.978	0.003	0.979	0.970
MNIST	ReLU	0.959	0.002	0.958	0.950
MNIST	SERF	0.983	0.005	0.983	0.972
MNIST	GELU	0.939	0.004	0.938	0.928
MNIST	Mish	0.950	0.007	0.948	0.942
MNIST	PSR	***0.993***	0.002	***0.991***	***0.991***
MNIST	LPSR	*0.992*	0.002	*0.990*	*0.989*
Fashion MNIST	SiLU	0.905	0.008	0.904	0.903
Fashion MNIST	ELU	0.904	0.009	0.903	0.902
Fashion MNIST	EELU	0.894	0.006	0.893	0.891
Fashion MNIST	ReLU	0.893	0.005	0.890	0.884
Fashion MNIST	SERF	0.859	0.006	0.859	0.859
Fashion MNIST	GELU	0.889	0.007	0.882	0.881
Fashion MNIST	Mish	0.824	0.074	0.829	0.829
Fashion MNIST	PSR	***0.912***	0.007	***0.909***	***0.909***
Fashion MNIST	LPSR	*0.907*	0.009	*0.907*	*0.905*

Table 5. Comparison of evaluation indicators of activation functions on Vision transformers with different datasets.

Dataset	Activation	ACC	Std	Precision	Recall
CIFAR10	SiLU	0.662	0.035	0.647	0.646
CIFAR10	ELU	0.663	0.044	0.647	0.646
CIFAR10	EELU	0.668	0.037	0.584	0.582
CIFAR10	ReLU	0.668	0.048	0.588	0.587
CIFAR10	SERF	0.648	0.039	0.577	0.576
CIFAR10	GELU	0.629	0.058	0.558	0.556
CIFAR10	Mish	0.659	0.039	0.565	0.564
CIFAR10	PSR	***0.672***	0.029	***0.669***	***0.668***
CIFAR10	LPSR	*0.669*	0.019	*0.657*	*0.656*
CIFAR100	SiLU	0.527	0.015	0.516	0.514
CIFAR100	ELU	0.543	0.020	0.522	0.520
CIFAR100	EELU	0.547	0.013	0.532	0.530
CIFAR100	ReLU	0.558	0.028	*0.552*	*0.551*
CIFAR100	SERF	0.537	0.038	0.538	0.536
CIFAR100	GELU	0.556	0.064	0.547	0.546
CIFAR100	Mish	0.550	0.058	0.543	0.542
CIFAR100	PSR	*0.569*	0.013	***0.557***	***0.556***
CIFAR100	LPSR	***0.569***	0.017	0.552	0.550
MNIST	SiLU	0.729	0.009	*0.718*	*0.717*
MNIST	ELU	0.728	0.009	0.714	0.712
MNIST	EELU	0.718	0.008	0.709	0.708
MNIST	ReLU	0.719	0.007	0.704	0.702
MNIST	SERF	0.698	0.005	0.638	0.637
MNIST	GELU	0.717	0.004	0.708	0.706
MNIST	Mish	0.689	0.007	0.627	0.625
MNIST	PSR	*0.730*	0.007	0.708	0.707
MNIST	LPSR	***0.731***	0.005	***0.729***	***0.727***
Fashion MNIST	SiLU	0.656	0.008	0.648	0.647
Fashion MNIST	ELU	0.656	0.008	0.641	0.640
Fashion MNIST	EELU	0.622	0.004	0.625	0.624
Fashion MNIST	ReLU	0.638	0.004	0.634	0.633
Fashion MNIST	SERF	0.640	0.006	0.637	0.636
Fashion MNIST	GELU	0.648	0.008	0.637	0.636
Fashion MNIST	Mish	0.639	0.007	0.637	0.635
Fashion MNIST	PSR	*0.656*	0.006	*0.649*	*0.648*
Fashion MNIST	LPSR	***0.657***	0.003	***0.659***	***0.658***

Table 6. Comparison of ACC Mean for each model across all datasets.

Activation	VGG16	ResNet50	DenseNet121	ViT	Mean *
SiLU	*0.834*	0.815	0.831	0.643	*0.781*
ELU	0.811	*0.819*	0.824	0.647	0.775
EELU	0.797	0.794	0.815	0.639	0.761
ReLU	0.809	0.800	0.816	0.647	0.768
SERF	0.817	0.778	0.812	0.631	0.759
GELU	0.804	0.771	0.814	0.637	0.757
Mish	0.801	0.787	0.788	0.634	0.753
PSR	***0.851***	***0.822***	***0.843***	***0.657***	***0.793***
LPSR	0.825	0.795	*0.833*	*0.657*	0.777

Mean * represents the overall mean of all datasets across the four models.

Table 7. ACC Average ranking across all datasets for each model.

Activation	VGG16	ResNet50	DenseNet121	ViT	Average
SiLU	3.8	3.0	3.3	5.3	3.8
ELU	6.0	2.5	5.5	5.0	4.8
EELU	5.0	5.5	7.0	6.3	5.9
ReLU	5.5	5.8	6.0	4.8	5.5
SERF	4.8	7.8	6.8	7.5	6.7
GELU	7.3	8.0	5.5	6.3	6.8
Mish	6.5	6.8	7.8	7.0	7.0
PSR	**1.0**	**1.3**	**1.0**	1.8	**1.3**
LPSR	5.5	4.8	2.3	**1.3**	3.4

5.1. Result Analysis

First, we will describe the results presented in Tables 2–6. Based on the average accuracy (ACC) results, the PSR activation function consistently outperformed all others across the following four evaluated architectures: Vision Transformer, DenseNet121, ResNet-50, and VGG-16. In the ResNet-50 and VGG-16 models, SiLU and ELU attained the second-highest accuracy, respectively. For Vision Transformer and DenseNet121, the LPSR function ranked second. When considering the overall average performance across all architectures, PSR emerged as the top-performing function, followed by SiLU.

An important observation during experimentation was the occurrence of gradient explosion in certain datasets when training the VGG-16 model. Notably, this issue did not arise in ResNet-50, DenseNet121, or Vision Transformer. This behavior is likely attributed to the absence of residual connections in VGG-16. Residual connections are known to facilitate more stable gradient flow by introducing identity shortcuts, which help counteract both gradient vanishing and explosion in deep networks. These shortcut pathways provide direct routes for gradients during backpropagation, preventing the excessive amplification or attenuation commonly encountered in deep architectures. As a result, the findings suggest that the proposed activation functions demonstrate improved compatibility and effectiveness when applied to more advanced network architectures that incorporate residual learning mechanisms.

Table 7 presents the comparative rankings of the activation functions across the evaluated model architectures. Both PSR and LPSR consistently achieved top ranks, with PSR exhibiting a slight performance advantage in ResNet-50, VGG-16, and DenseNet121, while LPSR led in the Vision Transformer model.

This preliminary analysis indicates that while the performance of most activation functions is relatively close, the proposed PSR and LPSR variants demonstrate superior results overall. To yield more conclusive findings, the next subsection introduces statistical significance testing, followed by an analysis using attention-based feature maps to further illustrate the advantages of the proposed activation strategies.

5.2. Friedman Statistical Test

We performed the Friedman test [57] to investigate the statistical significance of the performance differences among the activation functions. The null hypothesis H_0 posits that there is no difference in performance among all activation functions, while the alternative hypothesis H_A suggests that a difference exists. The Friedman statistic is computed using Formulas (32) and (33), as follows:

$$R_i = \frac{\sum_{q=1}^{K_M} r_{iq}}{K_M} \tag{32}$$

$$\chi^2 = \frac{12K_M}{K_A(K_A+1)}\left(\sum_{i=1}^{K_A} {R_i}^2 - \frac{K_A(K_A+1)^2}{4}\right) \tag{33}$$

where K_A is the number of activation functions being compared, K_M is the number of neural networks (or the number of datasets used to compare performance across different datasets). In our study, K_A and K_M are 9 and 4, respectively (since we have four datasets). R_i represents the average rank of the i-th activation function, and r_{iq} is the rank of the i-th activation function on the q-th neural network (or dataset). For each neural network (or dataset), activation functions are ranked from 1 to 4, where rank 1 indicates the lowest accuracy, and rank 4 indicates the highest accuracy. Tables 8 and 9 show the Friedman test results at a significance level of 0.05, Bold numbers indicate the best ranking results. For larger values of K_A and K_M, the Friedman statistic follows a chi-square distribution χ^2, so we computed the χ^2 values for both the neural networks and datasets. In this study, the degrees of freedom are $(K_M - 1)$, where K_M is the number of individual cases used for performance evaluation. Therefore, the degrees of freedom for both neural networks and datasets are 3. The calculated ${\chi^2}_{Critical}$ values for both the neural networks and datasets are approximately 7.81. Since in both cases, the values is lower than the critical χ^2 statistic, we reject the null hypothesis. Therefore, we conclude that the performance of all four activation functions is significantly different. Furthermore, the higher rankings and significant performance differences highlight the superiority of the proposed series of activation functions. The average rank (R_i) of PSR is the highest across all four activation functions, across different neural networks and datasets, ultimately indicating that PSR outperforms the current state-of-the-art activation functions. The properties of PSR, such as periodicity, nonlinearity, non-monotonicity, and boundedness (both upper and lower bounds), are similar to other activation functions. The only distinguishing feature of PSR is that both the upper and lower bounds are positive. With this unique characteristic, PSR consistently outperforms current state-of-the-art activation functions in nearly all scenarios.

Table 8. Results of Friedman test based on different neural networks.

Activation Function	Dataset Based Rank R_i	χ^2	*p*-Value	Hypothesis Test Result
SiLU	6.25			
ELU	3.5			
EELU	5.0			
ReLU	4.5			
SERF	3.0	17.93	<0.05	Reject H_0
GELU	3.0			
Mish	3.75			
PSR	**9.0**			
LPSR	7.0			

Table 9. Results of Friedman test based on different datasets.

Activation Function	Dataset Based Rank R_i	χ^2	p-Value	Hypothesis Test Result
SiLU	6.25			
ELU	3.75			
EELU	4.75			
ReLU	4.25			
SERF	5.0	16.33	<0.05	Reject H_0
GELU	2.5			
Mish	3.25			
PSR	**9.0**			
LPSR	6.25			

Another observation from the Friedman test regards the generalization ability of PSR performance. The χ^2 value based on the neural network ranking is more significant than that based on the dataset ranking. This is because the p value associated with the χ^2 value of 16.33 is 0.04, which, although significant, is not as strong because it is greater than 0.001 but less than 0.05. In contrast, the p value associated with the χ^2 value of 17.93 is 0.02, which is also significant, but similarly not as strong, as it is greater than 0.001 but less than 0.05. However, it is more significant than the results based on different datasets. The lower p value indicates the consistency of PSR performance across different neural networks, in other words, demonstrating its "generalization" ability. In contrast, the generalization ability across different datasets is weaker compared to across different neural networks.

5.3. The Superior Performance of the Developed Activation Function

To better adapt to gradient-based optimization, it is crucial to avoid singularities. In terms of smoothness and convergence, PSR outperforms other variants in terms of performance. The smooth transition in the output space effectively demonstrates a smoother loss landscape. As shown in Figure 3 and 4, we visualize the 2D attention heatmaps for the classes "bird" and "deer" . Using Grad-CAM heatmaps [58], we present the attention distribution of the DenseNet121 model in the denseblock1 layer to evaluate the performance of the model with nine different activation functions, indirectly reflecting the superiority of activation functions in the model. The network architecture combining PSR and LPSR in DenseNet121 stands out in classifying these two categories. Attention is primarily focused on the wings of the bird and the limbs of the deer, demonstrating the network's strong focus on target regions during classification, while the attention distribution for other activation functions is more dispersed. A comparative overview of PSR's functional attributes relative to other activation functions is presented in Table 10, highlighting its practical advantages in deep learning-based visual recognition tasks. Heatmaps serve not only as evidence of accuracy but also as material for design critique and decision-making. The development of visualization-based analytics dashboards enables designers to explore how PSR/LPSR alters feature maps and Grad-CAM heatmaps [59]. Beyond visual tasks, the sinusoidal periodicity of PSR can also act as a creative constraint for generative art and animated surface treatments (e.g., loading indicators that reflect feature map oscillations). Embedding model-driven rhythms into UI micro-interactions can reinforce the feedback loop between underlying intelligence and perceived experience.



Figure 3. Visualization of class activation maps for bird categories using DenseNet121 with different activation functions: SiLU, ELU, EELU, ReLU, SERF, GELU, Mish, PSR, and LPSR.



Figure 4. Visualization of class activation maps for deer categories using DenseNet121 with different activation functions: SiLU, ELU, EELU, ReLU, SERF, GELU, Mish, PSR, and LPSR.

Table 10. Comparison of mathematical properties of activation functions.

Property	SiLU	ELU	EELU	ReLU	SERF	GELU	Mish	PSR	LPSR
Nonlinear	✓	✓	✓	✓	✓	✓	✓	✓	✓
Upper-bounded	✗	✗	✗	✗	✗	✗	✗	✓	✓
Lower-bounded	✓	✓	✓	✓	✓	✓	✓	✓	✓
Differentiable	✓	✓	✓	✗	✗	✓	✓	✗	✓
Non-monotonic	✓	✗	✓	✗	✓	✓	✓	✓	✓

6. Conclusions

In this work, we present two periodic activation functions—PSR and LPSR—that satisfy key properties including nonlinearity, smoothness, differentiability, periodicity, and non-monotonicity. Extensive experiments on diverse neural architectures and datasets validate their effectiveness. Specifically, these functions were evaluated against standard

activation functions using three convolutional networks, one attention-based model, four benchmark datasets, and two large-scale datasets. PSR and LPSR consistently achieved top performance on the standard datasets and maintained competitive results on larger tasks. Although the differences among activation functions are generally subtle, careful selection remains important for task-specific optimization. Notably, the proposed functions demonstrated stable convergence and robust accuracy improvements across all tested configurations, outperforming widely used counterparts in rank-based analyses.

However, results indicate that PSR and LPSR exhibit stronger generalization across network types than across different datasets. Future work may explore tailored network designs to enhance adaptability and ensure consistent generalization on a broader range of tasks. Further research could also investigate how structural constraints influence LPSR's behavior. Extending these activations to additional domains—such as multimodal learning, speech recognition, and time series analysis—offers a promising avenue for future exploration. We contend PSR/LPSR enable real-time inference on metaverse platforms or in cultural heritage tourism AR, where latency and visual fidelity are critical determinants of user experience.

Author Contributions: Conceptualization, X.C., Y.C., and G.S.; Methodology, G.S. and Y.C.; Validation, X.C. and Y.C.; Formal analysis, S.W. and G.S.; Investigation, G.S., K.N., and J.W.; Resources, K.N.; Data curation, S.W.; Writing—original draft, X.C. and Y.C.; Writing—review and editing, G.S., K.N., and J.W.; Visualization, S.W.; Supervision, G.S., K.N., and J.W.; Project administration, K.N. and J.W.; Funding acquisition, G.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yann, L.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
2. Wang, S.; Ahn, S. Exploring User Behavior Based on Metaverse: A Modeling Study of User Experience Factors. In Proceedings of the International Conference on Human-Computer Interaction, Washington DC, USA, 29 June–4 July 2024; Springer Nature: Cham, Switzerland, 2024; pp. 99–118.
3. Wang, S.; Yu, J.; Yang, W.; Yan, W.; Nah, K. The impact of role-playing game experience on the sustainable development of ancient architectural cultural heritage tourism: A mediation modeling study based on S-O-R theory. *Buildings* **2025**, *15*, 2032. [CrossRef]
4. ZahediNasab, R.; Mohseni, H. Neuroevolutionary based convolutional neural network with adaptive activation functions. *Neurocomputing* **2020**, *381*, 306–313. [CrossRef]
5. Zhu, H.; Zeng, H.; Liu, J.; Zhang, X. Logish: A new nonlinear nonmonotonic activation function for convolutional neural network. *Neurocomputing* **2021**, *458*, 490–499. [CrossRef]
6. Apicella, A.; Donnarumma, F.; Isgrò, F.; Prevete, R. A survey on modern trainable activation functions. *Neural Netw.* **2021**, *138*, 14–32. [CrossRef]
7. Courbariaux, M.; Bengio, Y.; David, J.-P. Binaryconnect: Training deep neural networks with binary weights during propagations. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 3123–3131.
8. Gulcehre, C.; Moczulski, M.; Denil, M.; Bengio, Y. Noisy activation functions. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; PMLR; pp. 3059–3068.
9. Narayan, S. The generalized sigmoid activation function: Competitive supervised learning. *Inf. Sci.* **1997**, *99*, 69–82. [CrossRef]
10. Hochreiter, S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* **1998**, *6*, 107–116. [CrossRef]
11. Li, Y.; Yuan, Y. Convergence analysis of two-layer neural networks with ReLU activation. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 597–607.
12. Lu, L.; Shin, Y.; Su, Y.; Karniadakis, G.E. Dying ReLU and initialization: Theory and numerical examples. *arXiv* **2019**, arXiv:1903.06733. [CrossRef]

13. Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel, 21–24 June 2010; pp. 807–814.
14. Glorot, X.; Bordes, A.; Bengio, Y. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, Fort Lauderdale, FL, USA, 11–13 April 2011; JMLR Workshop and Conference Proceedings; pp. 315–323.
15. Xu, J.; Li, Z.; Du, B.; Zhang, M.; Liu, J. Reluplex made more practical: Leaky ReLU. In Proceedings of the 2020 IEEE Symposium on Computers and Communications (ISCC), Rennes, France, 7–10 July 2020; IEEE: New York, NY, USA, 2020; pp. 1–7.
16. Devi, T.; Deepa, N. A novel intervention method for aspect-based emotion using Exponential Linear Unit (ELU) activation function in a Deep Neural Network. In Proceedings of the 2021 5th International Conference on Intelligent Computing and Control Systems (ICICCS), Madurai, India, 6–8 May 2021; IEEE: New York, NY, USA, 2021; pp. 1671–1675.
17. Kim, D.; Kim, J.; Kim, J. Elastic exponential linear units for convolutional neural networks. *Neurocomputing* **2020**, *406*, 253–266. [CrossRef]
18. Hendrycks, D.; Gimpel, K. Gaussian error linear units (GELUs). *arXiv* **2016**, arXiv:1606.08415.
19. Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv* **2019**, arXiv:1908.08681.
20. Krizhevsky, A. Convolutional Deep Belief Networks on CIFAR-10. 2010. Available online: https://www.semanticscholar.org/paper/Convolutional-Deep-Belief-Networks-on-CIFAR-10-Krizhevsky/bea5780d621e669e8069f05d0f2fc0db9df4b50f#extracted (accessed on 9 August 2025).
21. Sharma, N.; Jain, V.; Mishra, A. An analysis of convolutional neural networks for image classification. *Procedia Computer Science* **2018**, *132*, 377–384. [CrossRef]
22. Sharif, M.; Kausar, A.; Park, J.; Shin, D.R. Tiny image classification using Four-Block convolutional neural network. In Proceedings of the 2019 International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 16–18 October 2019; IEEE: New York, NY, USA, 2019; pp. 1–6.
23. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Li, F. Imagenet: A large-scale hierarchical image database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009; IEEE: New York, NY, USA, 2009; pp. 248–255.
24. Vargas, V.M.; Gutiérrez, P.A.; Barbero-Gómez, J.; Hervás-Martínez, C. Activation functions for convolutional neural networks: Proposals and experimental study. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *34*, 1478–1488. [CrossRef] [PubMed]
25. Chen, D.; Li, J.; Xu, K. AReLU: Attention-based rectified linear unit. *arXiv* **2020**, arXiv:2006.13858.
26. Hoiem, D.; Divvala, S.K.; Hays, J.H. Pascal VOC 2008 challenge. *World Literature Today* **2009**, *24*, 1–4.
27. Sharma, D.K. Information measure computation and its impact in MI COCO dataset. In Proceedings of the 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 19–20 March 2021; IEEE: New York, NY, USA, 2021; Volume 1, pp. 1964–1969.
28. Dubey, S.R.; Singh, S.K.; Chaudhuri, B.B. Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing* **2022**, *503*, 92–108. [CrossRef]
29. Xia, Y.; Liu, Z.; Wang, S.; Huang, C.; Zhao, W. Unlocking the Impact of User Experience on AI-Powered Mobile Advertising Engagement. *J. Knowl. Econ.* **2025**, *16*, 4818–4854. [CrossRef]
30. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; JMLR Workshop and Conference Proceedings; pp. 249–256.
31. Srivastava, R.K.; Greff, K.; Schmidhuber, J. Highway networks. *arXiv* **2015**, arXiv:1505.00387. [CrossRef]
32. Elfwing, S.; Uchibe, E.; Doya, K. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Netw.* **2018**, *107*, 3–11. [CrossRef]
33. Paul, A.; Bandyopadhyay, R.; Yoon, J.H.; Geem, Z.W.; Sarkar, R. SinLU: Sinu-Sigmoidal Linear Unit. *Mathematics* **2022**, *10*, 337. [CrossRef]
34. Zhang, Z.; Li, X.; Yang, Y.; Shi, Z. Enhancing deep learning models for image classification using hybrid activation functions. *Preprint* 2023. [CrossRef]
35. Clevert, D.-A. Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv* **2015**, arXiv:1511.07289.
36. Deng, Y.; Hou, Y.; Yan, J.; Zeng, D. ELU-Net: An Efficient and Lightweight U-Net for Medical Image Segmentation. *IEEE Access* **2022**, *10*, 35932–35941. [CrossRef]
37. Wei, C.; Kakade, S.; Ma, T. The implicit and explicit regularization effects of dropout. In Proceedings of the International Conference on Machine Learning, Online, 13–18 July 2020; PMLR; pp. 10181–10192.
38. Daubechies, I.; DeVore, R.; Foucart, S.; Hanin, B.; Petrova, G. Nonlinear approximation and (deep) ReLU networks. *Constr. Approx.* **2022**, *55*, 127–172. [CrossRef]
39. Andrearczyk, V.; Whelan, P.F. Convolutional neural network on three orthogonal planes for dynamic texture classification. *Pattern Recognit.* **2018**, *76*, 36–49. [CrossRef]

40. Brownlowe, N.; Cornwell, C.R.; Montes, E.; Quijano, G.; Zhang, N. Stably unactivated neurons in ReLU neural networks. *arXiv* **2024**, arXiv:2412.06829. [CrossRef]
41. Maas, A.L.; Hannun, A.Y.; Ng, A.Y. Rectifier nonlinearities improve neural network acoustic models. In Proceedings of the ICML, Atlanta, GA, USA, 17–19 June 2013; Volume 30, p. 3.
42. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 13–16 December 2015; pp. 1026–1034.
43. Lin, G.; Shen, W. Research on convolutional neural network based on improved ReLU piecewise activation function. *Procedia Comput. Sci.* **2018**, *131*, 977–984. [CrossRef]
44. Nag, S.; Bhattacharyya, M.; Mukherjee, A.; Kundu, R. SERF: Towards Better Training of Deep Neural Networks Using Log-Softplus ERror Activation Function. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2–7 January 2023; pp. 5324–5333.
45. Miglani, V.; Kokhlikyan, N.; Alsallakh, B.; Martin, M.; Reblitz-Richardson, O. Investigating saturation effects in integrated gradients. *arXiv* **2020**, arXiv:2010.12697. [CrossRef]
46. Verma, V.K.; Liang, K.; Mehta, N.; Carin, L. Meta-learned attribute self-gating for continual generalized zero-shot learning. *arXiv* **2021**, arXiv:2102.11856.
47. Islam, M.S.; Zhang, L. A Review on BERT: Language Understanding for Different Types of NLP Task. *Preprints* 2024. [CrossRef]
48. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934. [CrossRef]
49. Fanaskov, V.; Oseledets, I. Associative memory and dead neurons. *arXiv* **2024**, arXiv:2410.13866.
50. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
51. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747. [CrossRef]
52. Dosovitskiy, A. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv* **2020**, arXiv:2010.11929.
53. Swaminathan, A.; Varun, C.; Kalaivani, S. Multiple plant leaf disease classification using DenseNet-121 architecture. *Int. J. Electr. Eng. Technol* **2021**, *12*, 38–57.
54. Koonce, B. ResNet 50. In *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 63–72.
55. Theckedath, D.; Sedamkar, R.R. Detecting affect states using VGG16, ResNet50 and SE-ResNet50 networks. *SN Comput. Sci.* **2020**, *1*, 79. [CrossRef]
56. Kingma, D.P. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980.
57. Liu, J.; Xu, Y. T-Friedman test: A new statistical test for multiple comparison with an adjustable conservativeness measure. *Int. J. Comput. Intell. Syst.* **2022**, *15*, 29. [CrossRef]
58. Selvaraju, R.R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; Batra, D. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 22–29 October 2017; pp. 618–626.
59. Wang, S.; Nah, K. Exploring sustainable learning intentions of employees using online learning modules of office apps based on user experience factors: Using the adapted UTAUT model. *Appl. Sci.* **2024**, *14*, 4746. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.