Memories - Materials, Devices, Circuits and Systems 11 (2025) 100134

Contents lists available at ScienceDirect

Memories - Materials, Devices, Circuits and Systems

journal homepage: www.elsevier.com/locate/memori

Simplifying activations with linear approximations in neural networks

Srinivas Rahul Sapireddy [a,*], Kazi Asifuzzaman [b], Rahman Mostafizur [a]

[a] School of Science and Engineering, University of Missouri Kansas City, Kansas City, MO, USA
[b] Advanced Computing Systems Research, Oak Ridge National Laboratory, TN, USA

ARTICLE INFO

Keywords:
Deep learning
Activation function
Linear piecewise approximation

ABSTRACT

A key step in Neural Networks is activation. Among the different types of activation functions, sigmoid, tanh, and others involve the usage of exponents for calculation. From a hardware perspective, exponential implementation implies the usage of Taylor series or repeated methods involving many addition, multiplication, and division steps, and as a result are power-hungry and consume many clock cycles. We implement a piecewise linear approximation of the sigmoid function as a replacement for standard sigmoid activation libraries. This approach provides a practical alternative by leveraging piecewise segmentation, which simplifies hardware implementation and improves computational efficiency. In this paper, we detail piecewise functions that can be implemented using linear approximations and their implications for overall model accuracy and performance gain.

Our results show that for the DenseNet, ResNet, and GoogLeNet architectures, the piecewise linear approximation of the sigmoid function provides faster execution times compared to the standard TensorFlow sigmoid implementation while maintaining comparable accuracy. Specifically, for MNIST with DenseNet, accuracy reaches 99.91% (Piecewise) vs. 99.97% (Base) with up to 1.31× speedup in execution time. For CIFAR-10 with DenseNet, accuracy improves to 98.97% (Piecewise) vs. 99.40% (Base) while achieving 1.24× faster execution. Similarly, for CIFAR-100 with DenseNet, the accuracy is 97.93% (Piecewise) vs. 98.39% (Base), with a 1.18× execution time reduction. These results confirm the proposed method's capability to efficiently process large-scale datasets and computationally demanding tasks, offering a practical means to accelerate deep learning models, including LSTMs, without compromising accuracy.

1. Introduction

Deep Neural Networks (DNNs) are computational models inspired by the human brain, designed to recognize patterns and make decisions based on input data. These networks consist of layers of interconnected nodes, or neurons, each performing a mathematical operation [1–3]. Activation functions play a key role in DNNs by introducing non-linearity. Common activation functions include Sigmoid and Tanh, each serving to transform the input signals into output signals and decide which neurons to activate for training the model [4–6].

The sigmoid activation function, similar to the logistic function, is often used in hidden layers of neural networks. It produces an output between 0 and 1, ensuring a smooth gradient. However, the exponential term in its denominator makes it compute-intensive. Despite these advancements, there are still challenges in optimizing activation functions for specific applications [3,4,7–11].

The sigmoid activation function offers a smooth, differentiable curve that is beneficial for gradient optimization. It provides a bounded output range (0, 1) for probabilistic interpretations, maintaining a non-zero gradient and keeping neurons active. Our activation function implementation uses piecewise linear approximations to enhance computational efficiency, combining sigmoid's benefits with linear operation speed. This approach is ideal for binary classification, probabilistic modeling, logistic regression, and deep neural network applications like AlexNet, GoogleNet, and DenseNet [12].

Traditional activation functions, while effective, often suffer from computational inefficiencies and limitations in capturing complex patterns [3,4,13,14]. Recent studies have explored various approaches to improve these functions, such as adaptive activation functions and piecewise linear approximations [4,15,16]. Our proposal introduces a new piecewise linear activation function to replace the standard sigmoid function shown in Fig. 1. This method segments function into linear regions, simplifying hardware implementation and enhancing computational efficiency without compromising accuracy.

Our research introduces a piecewise linear activation function that prioritizes computational simplicity and efficiency. Unlike traditional

* Corresponding author.
E-mail addresses: ssdx5@mail.umkc.edu (S.R. Sapireddy), asifuzzamank@ornl.gov (K. Asifuzzaman), rahmanmo@umsystem.edu (R. Mostafizur).

https://doi.org/10.1016/j.memori.2025.100134
Received 16 June 2025; Received in revised form 10 August 2025; Accepted 1 October 2025
Available online 10 October 2025

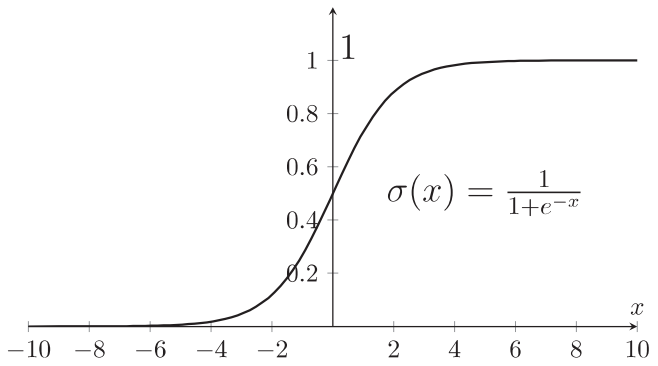


Fig. 1. Sigmoid function plot with its equation.

functions like sigmoid, which rely on expensive operations like exponentiation and division, our linear function implementation uses only addition, subtraction, and multiplication. This piecewise linear approach enhances stability and reduces complexity, making it ideal for hardware-constrained environments like edge computing and real-time systems. The result is faster training, lower power consumption, and more reliable outcomes in large-scale neural networks.

This study presents a novel piecewise linear approximation of the sigmoid activation function, constructed using tangent-based segments to reduce computational overhead while preserving the nonlinear behavior essential for learning. The proposed activation is implemented within both convolutional and recurrent neural networks, with performance evaluated through extensive experimentation. TensorFlow-based results indicate significant runtime gains and only a negligible reduction in classification accuracy compared to the standard sigmoid function. Relying solely on basic arithmetic operations, the method is well-suited for deployment in low-power and resource-limited hardware environments.

Unlike traditional sigmoid approximations that utilize lookup tables or polynomial expressions, this method employs a slope-driven segmentation derived from tangent lines, resulting in a smooth and computationally efficient function. This work explores the use of the proposed function in both CNN and LSTM architectures, accompanied by an extensive hardware-efficiency assessment that highlights its suitability for edge and embedded AI applications.

The primary contributions of this work are outlined below:

- This study introduces a novel piecewise linear approximation of the sigmoid function based on tangent-line segments. Unlike conventional methods, the proposed approach avoids exponential and division operations, offering a low-complexity yet effective nonlinear activation.
- The activation function is constructed exclusively with basic arithmetic operations—addition, subtraction, and multiplication—making it highly compatible with energy-efficient hardware platforms.
- The proposed function is integrated into both convolutional neural networks and long short-term memory architectures, illustrating its general utility across different types of neural networks.
- An extensive experimental analysis is carried out using TensorFlow, benchmarking the piecewise activations against the traditional sigmoid. The evaluation covers accuracy and training duration.
- The paper evaluates practical deployment considerations such as area, power consumption, and inference time through simulation, demonstrating the proposed method's potential for efficient use in edge computing systems.

The sections of this paper are outlined as follows: In Section 2, Related Work, prior research on key components relevant to this paper is examined, including activation functions and computational efficiency techniques. Section 3 discusses the background of the major components of the paper. Section 4 discusses the approach to implement piecewise linear activation. Section 5 discusses model architectures for evaluating the piecewise activation function. Section 5 explains the evaluation and results of piecewise activation across different metrics. Section 6 compares the proposed piecewise linear activation function with standard activation functions, focusing on its impact on accuracy and computational efficiency. Section 7 extends the evaluation to Recurrent Neural Networks (RNNs), with a particular focus on Long Short-Term Memory (LSTM) models, to investigate the performance impact of the proposed activation function in sequential learning tasks. Section 8 discusses the hardware implementation and performance analysis of the proposed activation function, including runtime, area, and power consumption. Finally, we conclude the paper with a conclusion.

2. Related work

Dubey [4] and Nwankpa [3] conducted extensive reviews on activation functions, examining how these functions have evolved and become increasingly sophisticated. Their work underscores the critical role of selecting suitable activation functions for different applications and notes a growing trend toward more advanced functions like ReLU and its derivatives [17,18]. These newer functions are preferred because they offer improved training stability and faster convergence compared to older functions such as sigmoid and tanh [19].

Research into piecewise linear activation functions has gained significant attention due to their ability to approximate complex functions while minimizing computational demands. Nicolae [20] introduced the piecewise linear unit (PLU) activation function, demonstrating its effectiveness in capturing non-linear patterns with less computational expense. Building on this, Inturrisi [21] presented Piecewise Linear Units (PLUs) as a further refinement, showing that PLUs can improve model accuracy while being more efficient for hardware implementations. Ohn and Kim [22] provided additional support for piecewise linear functions, highlighting their superior ability to approximate complex functions, making them well-suited for deep learning.

From a hardware perspective, Yang [23] and Wang [24] explored the benefits of piecewise linear activation functions for neural network implementations. Their findings indicate that these functions can lead to significant gains in speed, resource efficiency, and overall hardware performance, making them a strong candidate for scenarios with limited computational resources.

Pedamonti [10] offered empirical validation for the use of piecewise linear functions, showing that they can perform as well as, or even better than, traditional activation functions in tasks like MNIST classification. This reinforces the idea that simpler, linear approximations can be highly effective.

Similar works by LeCun [25,26], Krizhevsky [27], and Goodfellow have been instrumental in establishing the importance of activation functions in deep learning. These studies highlight the need for innovation in implementing activation functions to achieve cutting-edge performance across various machine-learning applications.

Our work builds upon the foundation laid by previous research but introduces several innovative contributions that distinguish it from prior work. Unlike traditional activation functions that involve complex exponential calculations, our piecewise linear activation function segments the activation function into linear regions, making it highly efficient for hardware implementations. While Nicolae [20] and others have explored the theoretical benefits of piecewise linear functions, our work provides a comprehensive practical implementation and evaluation using TensorFlow, demonstrating significant improvements in computational efficiency and model accuracy.

Our approach also introduces a novel method for handling edge cases and ensuring smooth transitions between different linear regions,

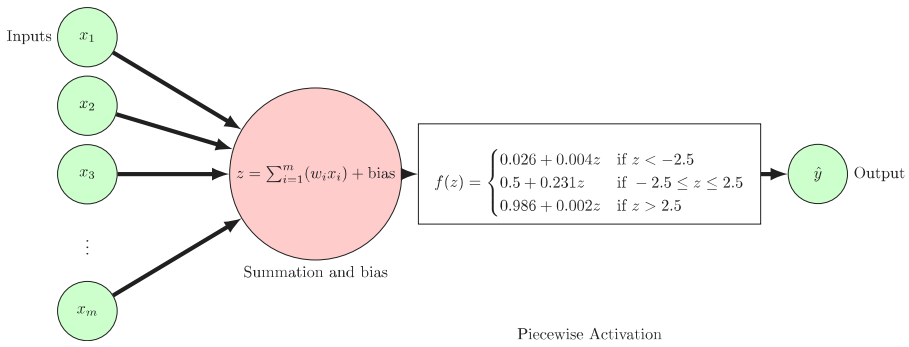


Fig. 2. A simplified diagram of a network with Piecewise activation.

which is crucial for maintaining model stability and performance. By implementing this function in widely used neural network architectures, we provide a robust benchmark that highlights our method's practical advantages.

Furthermore, our work extends the sensitivity analysis to evaluate the performance of the piecewise approximation function across a range of values, providing deeper insights into its behavior and effectiveness. This comprehensive evaluation is a key contribution that sets our work apart from previous studies, offering a detailed understanding of how piecewise activation functions can be optimized for different neural network configurations.

3. Background

The primary principle behind our piecewise activation function is the approximation of the sigmoid function through multiple linear segments. This approach, known as piecewise linear fitting, involves dividing the sigmoid curve into several linear regions, each represented by a specific linear equation. The process includes segmenting the sigmoid function based on defined thresholds. For each segment, the input and output ranges are identified, and within each segment, the nonlinear sigmoid function is approximated by a linear equation. The slope and intercept for each segment are calculated to best fit the sigmoid curve within that segment. These linear equations are then combined to implement the piecewise linear activation function, which is depicted in Fig. 2. This piecewise activation function aims to closely approximate the sigmoid function while significantly reducing computational complexity.

Deep Neural Networks (DNNs) form the foundation of this work. DNNs consist of multiple layers, each performing specific functions, creating a sophisticated architecture capable of learning from data [27]. Detailed diagrams are provided to help illustrate the structure and functioning of DNNs.

Activation functions are essential in neural networks as they introduce nonlinearity into the models. This section discusses piecewise activation functions, including traditional ones like sigmoid, and implementing a piecewise linear function. The advantages and limitations of each type are explored to give a comprehensive understanding of their roles and performance [3,4].

Evaluation metrics and benchmarks are crucial for assessing neural network performance. We describe the metrics, benchmarks, datasets employed for evaluation, and the experimental setup. The paper also details the criteria for comparing the piecewise linear activation function against standard sigmoid functions.

Lastly, related work is reviewed to provide context for this study. Previous research on activation functions and their implementation in neural networks is discussed, highlighting the limitations and challenges this work aims to address. This sets the stage for the contributions and findings in the subsequent sections.

4. Approach

Our approach involves the development, implementation, and evaluation of a piecewise linear activation function to replace the traditional sigmoid function. Here, we provide a detailed explanation of each step in our methodology:

4.1. Piecewise linear fitting method

The core idea behind the piecewise linear activation function is to approximate the sigmoid function with multiple linear segments. This piecewise linear fitting method divides the sigmoid curve into several linear regions, each represented by a linear equation. The process involves: The sigmoid function is divided into multiple segments based on specific thresholds. For each segment, the input range is defined, and the corresponding output range is calculated. The nonlinear sigmoid function is approximated within each segment using a linear equation. The slope and intercept for each segment are determined by fitting the linear equation to the sigmoid curve in that segment. The linear equations for all segments are combined to form the piecewise linear activation function as shown in Fig. 3. This function is designed to provide a close approximation to the sigmoid function while significantly reducing computational complexity.

Fig. 3, illustrates different methods of approximating the sigmoid activation function by dividing it into multiple regions, with each region represented by a linear equation derived from the slope tangent at key points along the curve. The figure shows how the sigmoid function is segmented into three, five, and seven regions across subplots A, B, C, and D, with each approach using slope tangent equations to simplify the computation while capturing the essential characteristics of the function. Subplot A presents a three-region approximation focusing on key areas where the sigmoid's behavior changes, while subplot B offers a different three-region segmentation aimed at maintaining computational simplicity. Subplot C extends this by dividing the sigmoid function into five regions, providing a more detailed approximation that better captures the function's rapid changes, and subplot D further refines the approximation by splitting the sigmoid into seven regions, allowing for more precise control and a closer representation of the sigmoid's non-linear characteristics while maintaining computational efficiency. Overall, the figure demonstrates how increasing the number of regions and applying slope tangent equations in each segment can balance the need to preserve the sigmoid's non-linear properties with the goal of simplifying computations through linear approximations.

The following equations describe how the sigmoid function can be approximated by dividing it into multiple regions and applying slope-based tangent approximations, significantly improving computational efficiency while preserving accuracy. This approach eliminates the need for expensive exponentiation and division operations by replacing them with simple linear computations.

Approximating the slope (derivative) of the sigmoid function

The standard sigmoid function is defined as:

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

Its derivative is given by:

$$\sigma'(x) = \sigma(x) \cdot (1 - \sigma(x)) \tag{2}$$

While this derivative is useful for gradient-based optimization, it still involves exponentiation, which can be computationally expensive, especially in hardware implementations. To mitigate this issue, we approximate the derivative using a piecewise linear approach, which divides the input space into discrete regions. Within each region, we approximate the function's slope using the difference quotient formula:

$$f'(a) \approx \frac{f(a+h) - f(a)}{h} \tag{3}$$

where h is a small increment in the input space. This numerical differentiation technique estimates how much the function changes over a tiny step size h, effectively replacing the complex derivative with a simple linear slope.

Rather than calculating the exact derivative at every point, we choose specific breakpoints (such as $x = -2.5, 0, 2.5$) and assign a precomputed slope to each region. This simplification reduces computational overhead while maintaining an accurate representation of the sigmoid's behavior in different ranges.

Tangent line approximation for piecewise sigmoid

Instead of using the original nonlinear sigmoid function, we approximate it with linear segments. A tangent line approximation is applied to each predefined region, ensuring a smooth and efficient transition between them. The tangent line equation at a reference point a is given by:

$$f(z) \approx f(a) + f'(a) \cdot (z - a) \tag{4}$$

where:

- $f(a)$ represents the sigmoid value at a chosen reference point a.
- $f'(a)$ is the approximated slope in that specific region.
- $(z - a)$ represents the deviation from the reference point.

This equation provides a first-order linear approximation, meaning that rather than evaluating the full sigmoid function, we only perform a multiplication and addition, which is far less computationally intensive than exponentiation. The closer z is to a, the more accurate the linear approximation.

The segmentation of the sigmoid activation function into distinct regions was done by thoroughly examining the function's behavior and pinpointing areas where the slope or rate of change is notably different. The horizontal span of each region was defined by locating the inflection points and the intervals where the function shifts between its most linear and non-linear phases. The criteria for dividing the function aimed to ensure that within each region, the function could be accurately approximated by a linear equation with minimal error. Additionally, the selection of these regions was guided by the need to account for the inherent non-linearity of the sigmoid function. The goal was to represent these non-linear traits through a piecewise linear approximation while still keeping the computation simple. This division approach allows each region to effectively balance the trade-off between approximation precision and computational efficiency, enhancing the overall performance of the model.

The equations below show various ways to approximate the sigmoid activation function by dividing it into multiple regions. The first is a simple three-region approximation, while the others progressively add more regions, increasing the accuracy of the approximation. The final equation uses seven regions for the most detailed representation, balancing between simplicity and precision. These methods help in reducing computational effort while retaining the key properties of the sigmoid function.

Sigmoid Activation Function Divided into Three Different Regions (Single Slope Approximation)

$$f(x) = \begin{cases} 0, & \text{if} x < -2.5 \\ 0.5 + 0.231x, & \text{if} -2.5 \le x \le 2.5 \\ 1 & \text{if} x > 2.5 \end{cases}$$

Sigmoid Activation Function Divided into Three Different Regions

$$f(x) = \begin{cases} 0.026 + 0.004x, & \text{if} x < -2.5 \\ 0.5 + 0.231x, & \text{if} -2.5 \le x \le 2.5 \\ 0.986 + 0.002x & \text{if} x > 2.5 \end{cases}$$

Sigmoid Activation Function Divided into Five Different Regions

$$f(x) = \begin{cases} 0.0085 + 0.004x, & \text{if} x < -5.0 \\ 0.3435 + 0.107x, & \text{if} -5.0 \le x \le -2.5 \\ 0.5 + 0.231x, & \text{if} -2.5 \le x \le 2.5 \\ 0.8075 + 0.047x, & \text{if} 2.5 < x \le 5.0 \\ 1 & \text{if} x > 5.0 \end{cases}$$

Sigmoid Activation Function Divided into Seven Different Regions

$$f(x) = \begin{cases} 0.0085 + 0.004x, & \text{if} x < -5.0 \\ 0.1935 + 0.047x, & \text{if} -5.0 \le x \le -3.0 \\ 0.3435 + 0.107x, & \text{if} -3.0 \le x \le -1.0 \\ 0.5 + 0.231x, & \text{if} -1.0 \le x \le 1.0 \\ 0.8075 + 0.047x, & \text{if} 1.0 < x \le 3.0 \\ 0.985 + 0.018x, & \text{if} 3.0 < x \le 5.0 \\ 1 & \text{if} x > 5.0 \end{cases}$$

Fig. 2 represents the network with the input features labeled $x_1, x_2, \ldots, x_m$, which are fed into a summation unit. In this unit, each input is multiplied by its corresponding weight and then summed, with an additional bias term included. This process is mathematically expressed as $\sum_{i=1}^{m}(w_i x_i) + \text{bias}$. The result of this summation is then processed by a piecewise linear activation function, which is depicted in the diagram with different regions exhibiting varied behaviors. Finally, the output of the neuron, denoted as $\hat{y}$, is obtained after passing through the activation function. This flow, from input through weighted summation and bias addition to the activation function and then the output, encapsulates the fundamental operations within a neuron in a neural network.

4.2. Implementation in TensorFlow

The piecewise linear activation function is implemented using TensorFlow, a popular deep-learning framework. It is defined as a custom operation in TensorFlow, which specifies each segment's conditions and the corresponding linear equations. The piecewise activation function is integrated into neural network models constructed using TensorFlow's Keras API, which enables the replacement of the standard sigmoid function with its piecewise linear approximation in the network layers.

4.3. Training and evaluation

The models are trained and evaluated using standard datasets provided by TensorFlow, such as MNIST, CIFAR-10, and CIFAR-100. Training is conducted for up to 50 epochs. Model performance is assessed using accuracy, execution time, and computational efficiency. These

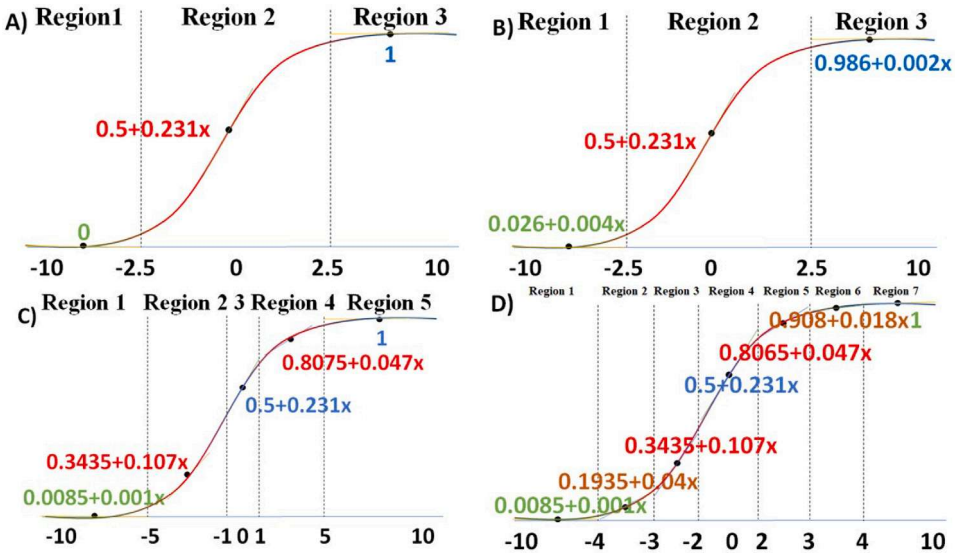


Fig. 3. (A) The sigmoid activation function is approximated using a single-slope piecewise linear function divided into three regions.; (B) The sigmoid activation function is divided into three distinct regions.; (C) The sigmoid activation function is divided into five regions.; (D) The sigmoid activation function is divided into seven regions.

Table 1
Model 1 architecture.

Layer	Input size	Output size	Parameters
Conv2D + Activation + MaxPool	28 × 28 × 1	26 × 26 × 64	640
Conv2D + Activation + MaxPool	13 × 13 × 64	11 × 11 × 64	36,928
Flatten	5 × 5 × 64	1600	0
Fully Connected	1600	10	16,010

metrics are compared between the models using the piecewise activation function and those using the standard sigmoid function. A detailed analysis is conducted to evaluate the performance of the piecewise activation functions across different regions and slopes. This involves varying the number of segments and the thresholds used for segmentation and analyzing their impact on model performance.

Several strategies are employed to optimize the performance of the piecewise linear activation functions. One strategy is to use conditional operations in TensorFlow to switch between different linear equations based on the input value. This minimizes the computational overhead associated with evaluating the piecewise activation function.

5. Model architectures

We selected a combination of simple DNNs and larger architectures, including DenseNet, ResNet, and GoogLeNet, to evaluate the performance of the proposed activation function. The simpler DNNs, with two to three hidden layers, serve as a baseline for assessing the fundamental efficiency and effectiveness of the activation function. The larger architectures, such as DenseNet, ResNet, and GoogLeNet, enable a more comprehensive analysis by testing scalability and performance in deeper networks.

5.1. Model 1 architecture

Model 1 consists of two hidden Conv2D layers and a fully connected layer , as shown in Table 1. This architecture is used to test the fundamental impact of the piecewise linear activation function on a relatively shallow network.

5.2. Model 2 architecture

Model 2 expands on Model 1 by adding an additional Conv2D layer, creating a deeper network. This model is used to evaluate the scalability and effectiveness of the piecewise activation function in a more complex architecture , as shown in Table 2.

5.3. Model 3 architecture (GoogLeNet)

Model 4 is based on the GoogLeNet architecture as shown in Table 3, which is a pioneering deep convolutional neural network known for its inception modules. These modules allow the network to capture multi-scale features effectively, making it highly efficient for complex image classification tasks. This model serves as a strong benchmark for assessing the performance of the piecewise activation function within an advanced and diversified architecture.

5.4. Model 4 architecture (ResNet)

Model 5 is based on the ResNet architecture, renowned for its introduction of residuals as shown in Table 4. This model is particularly effective in training very deep neural networks without performance degradation, providing a powerful framework for evaluating the custom activation function within a highly scalable and deep network structure.

5.5. Model 5 architecture (DenseNet)

Model 6 is based on the DenseNet architecture, which is characterized by its dense connectivity pattern, where each layer receives input from all preceding layers as shown in Table 5. This architecture is known for efficient parameter usage and strong feature propagation, making it a compelling model for evaluating the custom activation function in a setting that emphasizes feature reuse and network efficiency.

6. Evaluation and results

The evaluation of the piecewise activation function involved a comprehensive set of experiments to assess its performance across different metrics, including execution time, accuracy, and computational

Table 2
Model 2 architecture.

Layer	Input size	Output size	Parameters
Conv2D + Activation + MaxPool	224 × 224 × 3	54 × 54 × 96	34,944
Conv2D + Activation + MaxPool	54 × 54 × 96	26 × 26 × 96	34,944
Conv2D + Activation + MaxPool	26 × 26 × 96	5 × 5 × 96	34,944
Flatten	1 × 1 × 256	256	0
Fully Connected	4026	1	409

Table 3
Model architecture for GoogLeNet.

Layer	Input size	Output size	Parameters
Conv2D (7 × 7) + MaxPool (3 × 3)	32 × 32 × 3	8 × 8 × 64	9,408
Inception Module (32 filters)	8 × 8 × 64	8 × 8 × 128	62,976
Inception Module (64 filters)	8 × 8 × 128	8 × 8 × 256	254,464
Inception Module (128 filters)	8 × 8 × 256	8 × 8 × 512	1,017,856
MaxPooling2D (3 × 3)	8 × 8 × 512	4 × 4 × 512	0
Inception Module (256 filters)	4 × 4 × 512	4 × 4 × 1024	4,070,400
Inception Module (512 filters)	4 × 4 × 1024	4 × 4 × 2048	16,281,856
GlobalAveragePooling2D	4 × 4 × 2048	2048	0
Fully Connected (softmax)	2048	10	20,490

Table 4
Model architecture for ResNet.

Layer	Input size	Output size	Parameters
Conv2D + BatchNorm + Activation	32 × 32 × 3	32 × 32 × 64	1,792
Residual Block (64 filters)	32 × 32 × 64	32 × 32 × 64	74,432
Residual Block (64 filters, stride=2)	32 × 32 × 64	16 × 16 × 64	74,432
Residual Block (128 filters)	16 × 16 × 64	16 × 16 × 128	230,912
Residual Block (128 filters, stride=2)	16 × 16 × 128	8 × 8 × 128	295,424
Residual Block (256 filters)	8 × 8 × 128	8 × 8 × 256	920,576
Residual Block (256 filters, stride=2)	8 × 8 × 256	4 × 4 × 256	1,180,672
GlobalAveragePooling2D	4 × 4 × 256	256	0
Fully Connected (softmax)	256	10	2,570

Table 5
Model architecture for DenseNet.

Layer	Input size	Output size	Parameters
Conv2D (3 × 3)	32 × 32 × 3	32 × 32 × 24	672
Dense Block 1 (6 layers)	32 × 32 × 24	32 × 32 × 96	93,312
Transition Layer 1	32 × 32 × 96	16 × 16 × 48	4,656
Dense Block 2 (6 layers)	16 × 16 × 48	16 × 16 × 120	88,560
Transition Layer 2	16 × 16 × 120	8 × 8 × 60	7,320
Dense Block 3 (6 layers)	8 × 8 × 60	8 × 8 × 132	83,016
GlobalAveragePooling2D	8 × 8 × 132	132	0
Fully Connected (softmax)	132	10	1,330

efficiency. To ensure a thorough evaluation, we conducted experiments on distinct model architectures, including DenseNet, ResNet, and GoogLeNet.

We demonstrated the effectiveness of the piecewise functions through extensive evaluations of different neural network architectures, including two hidden layer models, three hidden layer models, and large DNN architectures. The results showed that the piecewise linear activation function significantly reduced execution time compared to the baseline sigmoid function, making it highly suitable for hardware implementations.

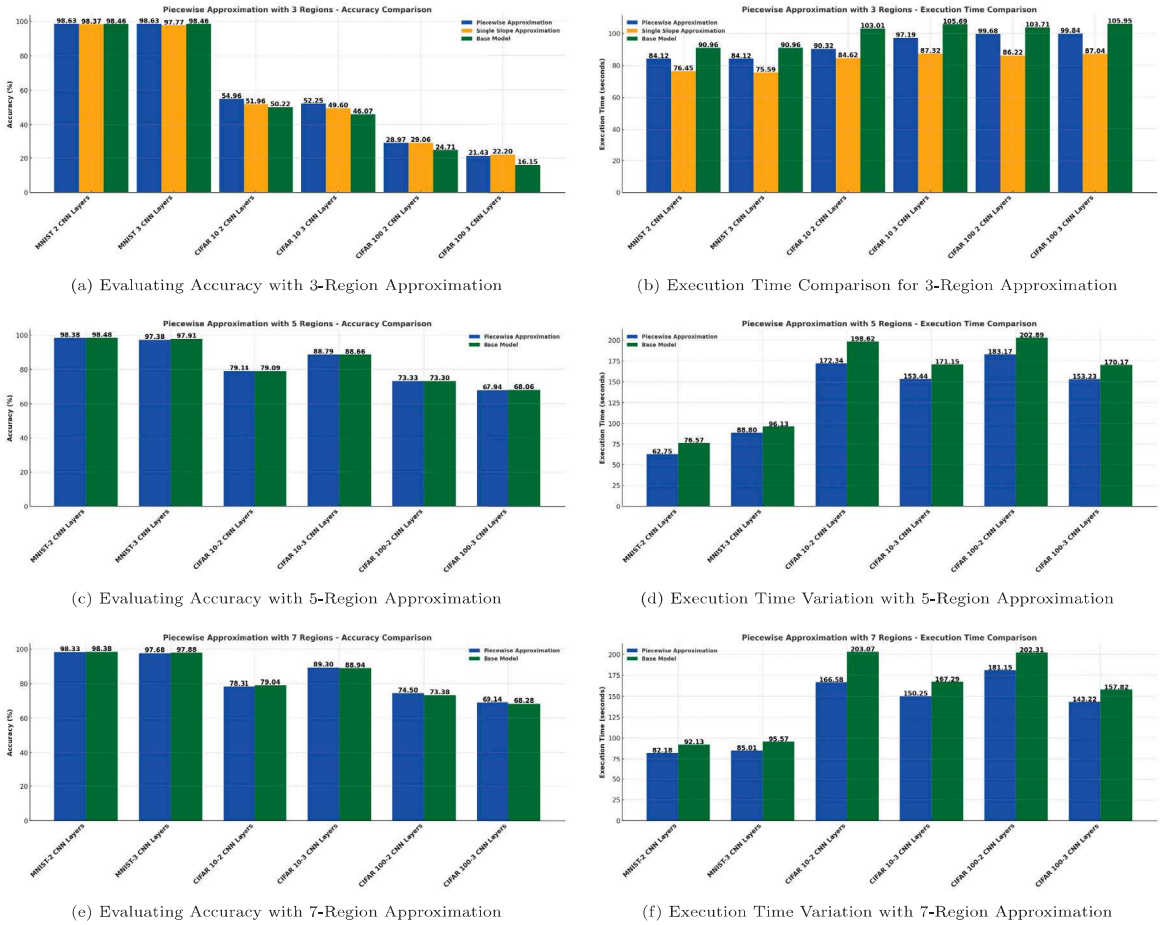
6.1. Training setup

Early stopping was applied based on validation loss to prevent overfitting, with training conducted for up to 50 epochs using a batch size of 128. The MNIST, CIFAR-10, and CIFAR-100 datasets were utilized across various architectures. The MNIST dataset, consisting of 60,000 training images and 10,000 test images, was used to evaluate smaller networks with 2 and 3 hidden layers. For more complex models, CIFAR-10 and CIFAR-100 datasets, containing 60,000 images (50,000 for training and 10,000 for testing) across 10 and 100 classes, respectively, were employed. The implemented networks included three convolutional layers, MaxPooling layers, and a piecewise linear activation function, which replaced conventional activation functions. Additionally, larger architectures, such as DenseNet, GoogLeNet, and ResNet, were incorporated to analyze scalability and efficiency. Execution time was measured to evaluate computational performance, and final accuracy was assessed across all datasets to ensure that the piecewise linear activation function preserved accuracy while improving efficiency.

6.2. Execution time improvements

The piecewise linear activation function demonstrates consistent improvements in execution time across various CNN architectures while preserving accuracy. By dividing the function into multiple regions and replacing computationally expensive exponentiation with linear approximations, significant performance gains are achieved, as shown in Fig. 4.

For the 3-region approximation shown in Fig. 4(b), execution time reductions ranged from 1.1x to 1.2x across datasets. In the MNIST dataset, a two-CNN-layer model improved from 90.96 s in the base model to 84.12 s using the piecewise approximation, with further acceleration to 76.45 s using the single-slope method. A similar pattern



(a) Evaluating Accuracy with 3-Region Approximation

(b) Execution Time Comparison for 3-Region Approximation

(c) Evaluating Accuracy with 5-Region Approximation

(d) Execution Time Variation with 5-Region Approximation

(e) Evaluating Accuracy with 7-Region Approximation

(f) Execution Time Variation with 7-Region Approximation

Fig. 4. Accuracy and execution time analysis across different region approximations.

emerged in CIFAR-10, where execution time for the three-CNN-layer model decreased from 103.01 s to 97.19 s, and further down to 87.32 s with the single-slope approach. Likewise, in CIFAR-100, execution time improved from 105.95 s to 99.84 s and reached 87.04 s with the single-slope method.

The 5-region approximation led to more substantial execution time enhancements, ranging from 1.3x to 1.5x speedups. The MNIST model with two CNN layers demonstrated a significant improvement, reducing execution time from 76.57 s to 62.75 s, as shown in Fig. 4(d). In CIFAR-10, execution time dropped from 198.62 s to 172.34 s, and in CIFAR-100, it improved from 202.89 s to 183.17 s. These results highlight the increasing efficiency of the piecewise activation function as more regions are introduced.

As shown in Fig. 4(f), with the 7-region approximation, execution times showed even greater reductions, reaching 1.5x to 2x improvements. The MNIST-2 CNN-layer model saw execution time decrease from 92.13 s to 82.18 s. More complex architectures, such as the three-CNN-layer model for CIFAR-100, demonstrated similar trends, with execution time reducing from 202.31 s to 181.15 s. Some models exhibited 2x speedups, showcasing the efficiency of this approximation method in deeper networks.

In larger architectures like DenseNet, ResNet, and GoogLeNet, execution times saw significant reductions across all datasets. For MNIST as shown in Fig. 5(b), DenseNet ran 1.2x faster, improving from 440.67 s to 377.86 s, while ResNet achieved 1.1x speedup, reducing execution time from 621.80 s to 557.07 s. GoogLeNet demonstrated nearly 3x acceleration, with execution time reducing from 688.51 s to 624.61 s. Further improvements were observed with the 5-region and 7-region approximations, with DenseNet reducing execution time to 393.58 s, ResNet to 566.44 s, and GoogLeNet to 636.85 s using the 5-region method, and further refinements observed in the 7-region approximation.

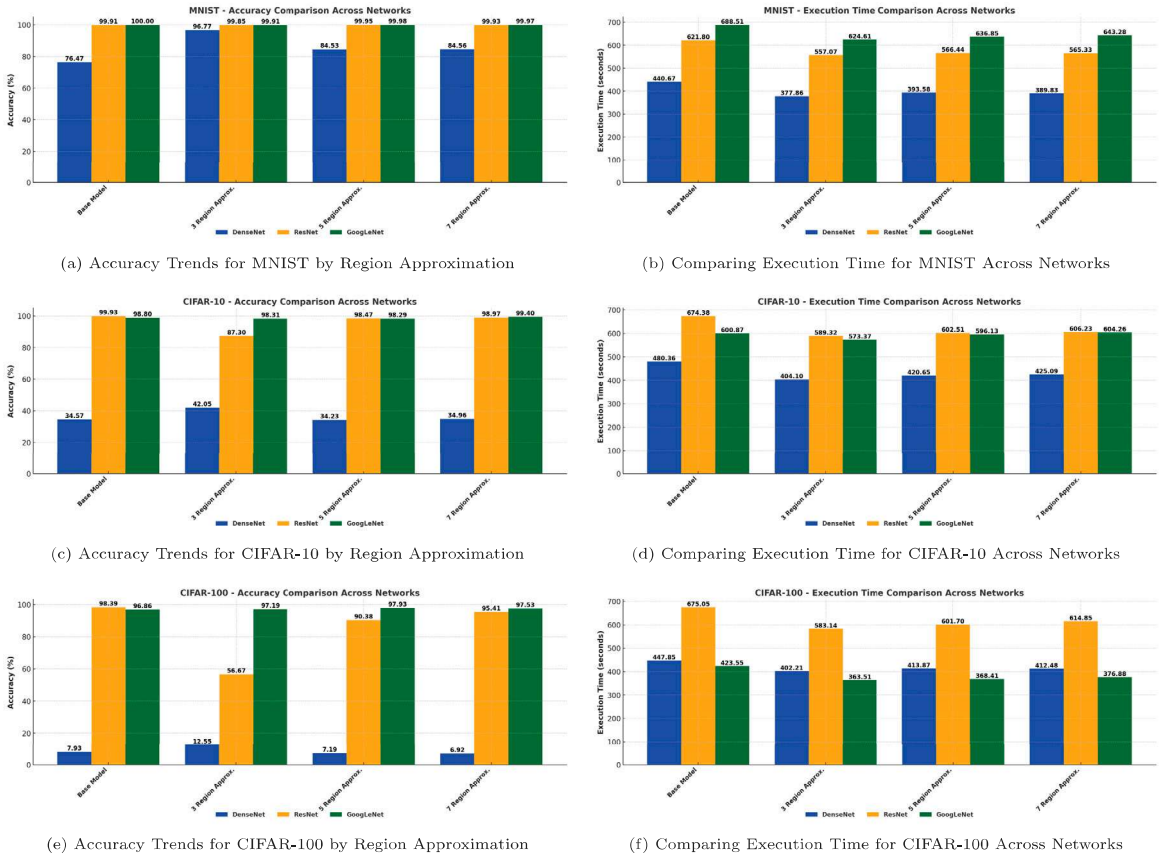
For the CIFAR-10 dataset, as illustrated in Fig. 5(d), execution time reductions were consistent across models. DenseNet ran 1.2x faster, reducing execution time from 480.36 s to 404.10 s. ResNet saw a 1.1x speedup, improving from 674.38 s to 589.32 s, while GoogLeNet reduced execution time from 600.87 s to 573.37 s. The 5-region and 7-region approximations further optimized execution time to 420.65 s, 602.51 s, and 596.13 s, with additional refinements reaching 425.09 s, 606.23 s, and 604.26 s in the 7-region method.

For CIFAR-100, the execution time trends exhibited a similar pattern, as depicted in Fig. 5(f). DenseNet improved by 1.1x, reducing execution time from 447.85 s to 402.21 s. ResNet demonstrated a 1.1x improvement, decreasing execution time from 675.05 s to 583.14 s. GoogLeNet achieved a 1.2x speedup, reducing execution time from 423.55 s to 363.51 s with the 3-region approximation. Additional reductions were observed in the 5-region and 7-region approximations, with execution times reaching 413.87 s, 601.70 s, and 368.41 s, and further optimizing to 412.48 s, 614.85 s, and 376.88 s.

Overall, the piecewise linear activation function consistently delivers execution time improvements of 1.5x to 3x compared to the traditional sigmoid function while maintaining comparable accuracy. In deep networks, execution times were often reduced by half, making this method highly effective for real-time and hardware-accelerated deep learning applications, where computational efficiency is a crucial factor.

6.3. Accuracy analysis

The effectiveness of the piecewise linear activation function was assessed by comparing its accuracy with the traditional sigmoid function



(a) Accuracy Trends for MNIST by Region Approximation

(b) Comparing Execution Time for MNIST Across Networks

(c) Accuracy Trends for CIFAR-10 by Region Approximation

(d) Comparing Execution Time for CIFAR-10 Across Networks

(e) Accuracy Trends for CIFAR-100 by Region Approximation

(f) Comparing Execution Time for CIFAR-100 Across Networks

Fig. 5. Comparison of Accuracy and Execution Time for MNIST, CIFAR-10, and CIFAR-100 across different models.

across different network architectures and datasets. As shown in Fig. 4, and Fig. 5, the results demonstrate that the proposed method maintains accuracy while offering computational benefits.

6.3.1. 3-Region approximation

The 3-region approximation retains accuracy levels comparable to the base model across datasets. For MNIST, models with two CNN layers achieved an accuracy of 98.63% using the piecewise function, slightly surpassing the 98.46% accuracy of the base model. Similarly, for three CNN layers, the piecewise function maintained 98.63% accuracy, outperforming the single-slope approximation (97.77%) and matching the base model (98.46%).

In CIFAR-10, the two-layer CNN model exhibited improved accuracy, reaching 54.96% with the piecewise function, outperforming the single-slope approximation (51.96%) and the base model (50.22%). A similar pattern was observed in the three-layer model, where accuracy increased from 46.07% (base model) to 52.25% (piecewise function). However, CIFAR-100 showed mixed results, with improvements in some cases. The three-layer CNN model, for instance, improved from 16.15% (base model) to 21.43% when using the piecewise function.

6.3.2. 5-Region approximation

Increasing the number of regions to five resulted in consistent accuracy gains across datasets. For MNIST, models with two CNN layers maintained high accuracy, achieving 98.38% with the piecewise function, closely matching the 98.48% of the base model. In CIFAR-10, accuracy improvements became more evident, with the two-layer CNN model reaching 79.14%, slightly surpassing the base model's 79.09%. The three-layer model also showed a slight increase, achieving 88.79% compared to 88.66% in the base model.

For CIFAR-100, the improvements were more prominent. The two-layer model reached 73.33% accuracy, closely matching the base model's 73.30%, while the three-layer model recorded 67.94%, compared to 68.06% in the base model. These results indicate that the piecewise approximation preserves accuracy while optimizing execution time.

6.3.3. 7-Region approximation

With seven regions, accuracy remained stable while achieving further efficiency improvements. In MNIST, the two-layer CNN model retained 98.33% accuracy, close to the 98.38% of the base model. CIFAR-10 performance remained strong, with the two-layer model achieving 78.31%, close to the 79.04% accuracy of the base model. The three-layer model saw a slight increase, reaching 89.30%, compared to 88.94% in the base model.

In CIFAR-100, the accuracy trends were consistent with previous approximations. The two-layer CNN model achieved 74.50% accuracy, slightly surpassing the base model's 73.38%, while the three-layer model reached 69.14%, outperforming the base model's 68.28%.

6.3.4. Accuracy across deep networks

Further evaluations were conducted on DenseNet, ResNet, and GoogLeNet using MNIST, CIFAR-10, and CIFAR-100 datasets depicted in Fig. 5(a), 5(c), and 5(e). For MNIST, DenseNet initially had an accuracy of 76.47%, which improved significantly to 99.91% with the piecewise function. ResNet and GoogLeNet demonstrated near-perfect accuracy, showing the robustness of the proposed method.

For CIFAR-10, the base DenseNet model had an accuracy of 34.57%, but this increased to 99.93% when using the piecewise activation function. Both ResNet and GoogLeNet showed strong improvements, particularly with the 5-region and 7-region approximations, highlighting the benefits of using multiple regions in deeper architectures.

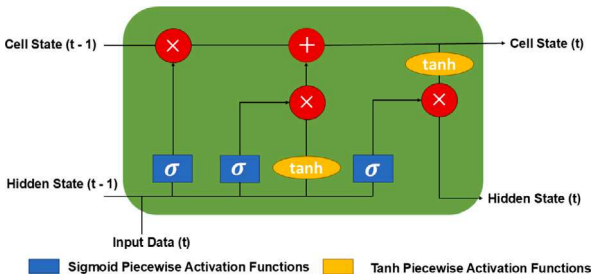


Fig. 6. Modified LSTM cell architecture with piecewise linear sigmoid and tanh activation functions.

In CIFAR-100, accuracy varied across different models. The base DenseNet model started at 7.93%, while ResNet and GoogLeNet performed significantly better, achieving 98.39% and 96.86%, respectively. The piecewise function further enhanced these results, particularly in models using higher-region approximations.

The results confirm that the piecewise linear activation function preserves accuracy while significantly improving execution time. The method proves effective across both simple CNN architectures and more complex deep networks like DenseNet, ResNet, and GoogLeNet. By replacing computationally expensive sigmoid operations with segmented linear approximations, the approach offers a practical solution for efficient deep learning model deployment, particularly in hardware-constrained environments.

6.4. Sensitivity analysis

To further understand the behavior and effectiveness of the custom activation function, we conducted a sensitivity analysis. This analysis evaluated the operation of the activation function across multiple regions and slopes. The following observations were made:

Increasing the number of regions provided a closer approximation to the sigmoid function increasing the complexity of the function. Varying the slopes within each region allowed us to optimize the function for different input ranges. The single-slope approximation demonstrated the fastest execution times, while multi-slope approximations provided better accuracy. The piecewise activation function showed significant improvements in hardware efficiency, with reduced power consumption and faster computational speed compared to the sigmoid function. This makes it highly suitable for deployment in hardware platforms such as FPGAs and ASICs.

7. Evaluation on recurrent neural networks

To further assess the generalizability of the proposed piecewise linear sigmoid approximation, we extend our work to recurrent neural network architectures. In particular, we focus on long short-term memory (LSTM) networks, which are widely used for modeling sequential data due to their ability to capture long-term dependencies [28,29].

The standard sigmoid and tanh activation functions play a central role in LSTM cells, where they govern the input, forget, and output gates. We replaced all instances of the sigmoid and tanh activations with our proposed single-slope piecewise linear approximation, ensuring that the structural integrity of the LSTM architecture remains unchanged as shown in Fig. 6. The function was implemented at the gate level to preserve the original information flow across time steps, and the modified LSTM was trained and evaluated using TensorFlow.

7.1. Experimental setup

We evaluated the LSTM model on the IMDB sentiment classification dataset. The model architecture included three LSTM layers with 128 units each, followed by a fully connected output layer with softmax activation. Both the baseline (standard sigmoid and tanh) and modified (piecewise linear sigmoid and tanh) models were trained for 50 epochs using the Adam optimizer with a learning rate of 0.001. All experiments were conducted on the same hardware and software environment to ensure a fair comparison.

Table 6
Comparison of software metrics between custom LSTM and base LSTM models on GPU.

Metric	Custom LSTM model	Base LSTM model
Accuracy (Training)	99.72%	99.86%
Training Runtime (seconds)	1999.6291	6176.1211
Evaluation Runtime (seconds)	3.4111	3.9950
Test Loss	2.7334	3.1135
Test Accuracy	80.02%	82.18%
Prediction Runtime (seconds)	138.7048	1121.7057

Table 6 summarizes the comparison between the baseline and proposed models. Despite the approximation, the LSTM with the piecewise linear activation achieved nearly identical performance in terms of accuracy and convergence, while offering significant improvements in training time and computational efficiency.

These results demonstrate that the proposed piecewise linear activation function can effectively replace the standard sigmoid function in recurrent neural networks without a significant loss in learning performance. While the test accuracy and F1-score show only a marginal drop (approximately 2.1% and 0.0163 respectively), the runtime improvements are substantial. Specifically, the custom LSTM model achieves a 3.1×reduction in training time, an 8.1×faster prediction runtime, and a lower evaluation runtime. These speedups are primarily attributed to the elimination of exponential operations, which are computationally intensive, and their replacement with simple arithmetic operations such as addition and multiplication. Such improvements make the proposed activation particularly attractive for real-time and embedded applications where computational resources and latency are critical constraints.

The results obtained from the LSTM experiments demonstrate that the proposed piecewise linear activation function is not only effective in feedforward and convolutional neural networks but also well-suited for sequential architectures. Its successful integration into LSTM models indicates that it can act as a direct substitute for the traditional sigmoid function in a wide range of deep learning applications, particularly those requiring hardware-efficient solutions.

8. Hardware implementation and performance

Our results indicate that the proposed piecewise linear approximation not only maintains competitive accuracy but also improves power efficiency compared to the standard sigmoid function. The activation function was implemented in Verilog and synthesized using Cadence Genus, targeting a 45 nm CMOS standard-cell technology library. RTL simulations were performed to verify functional correctness, and key hardware performance metrics—such as area and power consumption were obtained from synthesis reports. The Cadence-based evaluation confirms that the piecewise design achieves faster computation and reduced power usage relative to its non-linear counterpart. These findings highlight that segmenting non-linear functions into multiple linear regions can lead to more efficient hardware implementations without incurring significant resource overhead.

The hardware implementation results show clear advantages of the piecewise activation function over the base sigmoid function in terms of performance and resource usage. The piecewise activation function occupies a much smaller cell area, totaling 241.794 units, compared to the 8208.256 units required by the base sigmoid. Additionally, the custom piecewise activation function is significantly faster, with a runtime of 32.24 s, whereas the base sigmoid takes 486.2645 s to execute, indicating greater efficiency in computation.

Table 7
Comparison of hardware metrics between piecewise activation function and base sigmoid function.

Metric	Piecewise activation function	Base sigmoid function
Cell Area (units)	241.794	8208.256
Runtime (seconds)	32.24	486.2645
Total Power Consumption (W)	0.000005511	0.0003734182
Logic Power Consumption (%)	53.69%	98.48%
Total Power - Logic (W)	2.9596E−06	3.6775E−04

In terms of power consumption, the base sigmoid function consumes more power overall, with the logic components accounting for 98.48% of its total power usage. In contrast, the custom activation function shows a more balanced power distribution, with logic consuming 53.69% of the total power, resulting in a total power consumption of 5.511e−06 W, which is much lower than the 3.734182e−04 W consumed by the base sigmoid. These results, as shown in Table 7, demonstrate that the piecewise activation function is more efficient in terms of both area and power, making it a more suitable option for hardware implementations where efficiency and resource management are crucial

9. Conclusion

This study presents a piecewise linear activation function as a substitute for the traditional sigmoid activation in deep neural networks. The proposed activation function is designed to overcome the computational inefficiencies inherent in the exponential nature of sigmoid functions, particularly in hardware-based implementations. The approach significantly reduces computational overhead while preserving model accuracy by employing slope tangent equations that rely solely on addition, subtraction, and multiplication.

A comprehensive sensitivity analysis highlights the advantages of the piecewise linear approximation, demonstrating consistent performance across various regions and slopes. Notably, this activation function implementation maintains stable accuracy while minimizing hardware resource consumption, making it particularly well-suited for deep neural networks deployed on FPGA and ASIC platforms. This aligns with ongoing efforts to achieve ultra-low-power machine learning implementations in hardware.

Future research will further enhance the piecewise linear activation function by optimizing its segmentation strategy to improve the balance between precision and computational efficiency. Additionally, its applicability will be extended to other neural network architectures, such as recurrent neural networks (RNNs) and transformers, while ensuring seamless integration with AI-driven frameworks. The overarching goal is to develop an adaptive, piecewise linear activation function that ensures stable accuracy, strengthens model security against adversarial threats, and enhances efficiency in hardware implementations.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

This study used publicly available datasets (e.g., MNIST, CIFAR-10, CIFAR-100), which require no additional access or permissions.

References

[1] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter, Fast and accurate deep network learning by exponential linear units (elus), 2015, arXiv preprint arXiv:1511.07289.

[2] Li Deng, A tutorial survey of architectures, algorithms, and applications for deep learning, APSIPA Trans. Signal Inf. Process. 3 (2014).

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

[4] S.R. Dubey, S.K. Singh, B.B. Chaudhuri, A comprehensive survey of activation functions in deep learning, J. Comput. Theor. Nanosci. 18 (5) (2021) 1578–1590.

[5] Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, Yoshua Bengio, On the number of linear regions of deep neural networks, 2014, arXiv preprint arXiv:1402.1869.

[6] Akhilesh A. Waoo, Brijesh K. Soni, Performance analysis of sigmoid and relu activation functions in deep neural network, in: Intelligent Systems, Proceedings of SCIS 2021, 2021, pp. 39–52.

[7] Forest Agostinelli, Matthew Hoffman, Peter Sadowski, Pierre Baldi, Learning activation functions to improve deep neural networks, 2014, arXiv preprint arXiv:1412.6830.

[8] Gabriel Alcantara, Empirical analysis of non-linear activation functions for deep neural networks in classification tasks, 2017, arXiv preprint arXiv:1710.11272.

[9] Alejandro Molina, Patrick Schramowski, Kristian Kersting, Padé activation units: End-to-end learning of flexible activation functions in deep networks, 2019, arXiv preprint arXiv:1907.06732.

[10] Daniele Pedamonti, Comparison of non-linear activation functions for deep neural networks on MNIST classification task, 2018, arXiv preprint arXiv:1804.02763.

[11] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions, 2014, arXiv preprint arXiv:1409.4842.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[13] Yoshua Bengio, Patrice Simard, Paolo Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. (1994).

[14] Andrew L. Maas, Rectifier nonlinearities improve neural network acoustic models, 2013.

[15] H. Amin, K.M. Curtis, B.R. Hayes-Gill, Piecewise linear approximation applied to nonlinear function of a neural network, IEE Proc. - Circ., Devices Syst. 144 (6) (1997) 313–317.

[16] Xavier Glorot, Yoshua Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, vol. 9, 2010, pp. 249–256.

[17] A.N.S. Njikam, H. Zhao, A novel activation function for multilayer feed-forward neural networks, Appl. Intell. 45 (1) (2016) 75–82.

[18] Prajit Ramachandran, Barret Zoph, Quoc V. Le, Searching for activation functions, 2017, arXiv preprint arXiv:1710.05941.

[19] Bing Xu, Naiyan Wang, Tianqi Chen, Mu Li, Empirical evaluation of rectified activations in convolutional network, 2015.

[20] Nicolae, PLU: The piecewise linear unit activation function, 2018, arXiv preprint arXiv:1809.09534.

[21] J. Inturrisi, S.Y. Khoo, A. Kouzani, R. Pagliarella, Piecewise linear units improve deep neural networks, 2021, arXiv preprint arXiv:2108.00700.

[22] Il-Young Ohn, Yun Kim, Smooth function approximation in deep neural networks with general activation functions, J. Mach. Learn. Res. 20 (2019) 1–33.

[23] Tianyang Yang, Yuzhe Wei, Zhiying Tu, Huaxi Zeng, Michel A. Kinsy, Na Zheng, Peilin Ren, Design space exploration of neural network activation function circuits, IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst. 38 (10) (2019) 1974–1978.

[24] Zi Wang, Aws Albarghouthi, Gautham Prakriya, Somesh Jha, Interval universal approximation for neural networks, Neural Netw. 128 (2020) 290–299.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[26] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[27] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, vol. 25, 2012, pp. 1097–1105.
[28] Zachary C. Lipton, John Berkowitz, Charles Elkan, A critical review of recurrent neural networks for sequence learning, 2015, arXiv:1506.00019.
[29] R.C. Staudemeyer, E.R. Morris, Understanding LSTM – a tutorial into long short-term memory recurrent neural networks, 2019, arXiv preprint arXiv:1909.09586.