

# ErfPlus: A Smooth Monotonic Activation Function Utilizing the Error Function

Johan Joby  
Dept. of Mathematics  
Western University  
London, ON, Canada  
jjoby@uwo.ca

Erika Kemp  
Dept. of Computer Science  
Western University  
London, ON, Canada  
ekemp7@uwo.ca

Julian Rochacz  
Dept. of CS and Math  
Western University  
London, ON, Canada  
jrochacz@uwo.ca

Aidan James  
Dept. of Computer Science  
Western University  
London, ON, Canada  
ajame23@uwo.ca

**Abstract**—Smooth monotonic activation functions play a crucial role in the performance, convergence, and replicability of deep neural networks [10]. Error-based activation functions such as ErfAct [5], ErfReLU [11] and GELU [8] have been shown to work very well in deep learning models and tasks, resulting in models with high accuracy. In this paper, we introduce the error plus unit, ErfPlus, which is an error function based activation function that is monotonic, smooth, and centered at 0. We compare ErfPlus to activation functions with similar properties: ReLU, SoftPlus, SquarePlus, GELU and ErfReLU, and test their capability and reproducibility for a Fashion MNIST image classification task, utilizing a ResNet-like model. Performance is assessed across multiple experiments, and measurements are captured through validation scores and prediction difference (PD) values. Results show that ErfPlus achieves the highest mean values and lowest standard deviation (SD) values, indicating both high precision and robustness. Additionally, the error function based activations consistently perform as well as their counterparts in this study. These findings suggest that ErfPlus provides a stable, high-performing alternative activation function suitable for modern deep learning applications.

## I. INTRODUCTION

Activation functions are a key aspect in neural networks. They serve the purpose of introducing non-linearity, in order to transform the output from what a simple linear-regression model would produce, which enables the network to learn relationships between the features. There is a significant interest in finding the optimal activation function for deep learning tasks and what properties these activation functions share. Consequently, breakthroughs in this space will have incredible real-world applications in the medical imaging field, image recognition used for facial recognition, and natural language processing. Current literature suggests that activation functions which have the following mathematical properties may have computational benefits:

- Functions that are **bounded below** and **unbounded above** are able to simulate functions which are *both* unbounded and bounded. Having an activation function that is strictly bounded or unbounded might make it hard to simulate the functions with the other property. Boundedness is important to keep gradients within a certain range, avoiding exploding gradients [14] that result in extremely large weight values, causing the network to diverge instead of learning effectively.

- **Smooth (A)** functions have continuous **(B)** derivatives, that avoid discontinuities that cause singularities and undesired side effects when updating parameters during gradient-based backpropagation optimization [6].
- The **Lipschitz continuous (C)** property states that the absolute value of the derivative never exceeds 1, ensuring there are no large dips or pockets to be computed, and that gradients do not have unstable growths.
- **Monotonic functions** create neural networks that are transparent, interpretable, generalizable, data efficient, and have a higher performance than other functions [13]. Their output landscapes have less bad local minima, compared to landscapes made from non-monotonic functions.
- A **monotonic derivative** will help with smoothing and containing the gradients during backpropagation, as it ensures stable learning and faster convergence.
- In terms of graphs, it is desirable for an activation function to display ReLU-like properties, or to be **centered at 0**. When  $f$  is at 0,  $x$  is approximately 0.
- Additionally, a function should be **linear on the positive side**, its derivative after 0 is approximately equal to 1, and **saturated on the negative**,  $f$  starts to approach 0 with large negative values.
- Activation functions are applied to the outputs of all of the neurons in the models in the common case, so **computation** and execution time are heavily related to one another [14]. Although this is not required for activation functions, being computationally nice is desired.

There exist many examples of commonly used activation functions, such as ReLU (the Rectified Linear Unit), SoftPlus, SquarePlus, GELU (the Gaussian Error Linear Unit), and ErfReLU (the Error Function Linear Unit), which will all be tested and compared against in this paper. We will be exploring more details about the other activation functions being used in our experiments, and introduce our function, **ErfPlus**, in the mathematical analysis. Then, we will describe our dataset, model, and the experiments we ran in order to compare ErfPlus against the 5 other AFs, looking at their validation scores, prediction difference values, and discuss other results and outcomes. Lastly, we will expand upon limitations of our process and design, and go over possible future works.

## II. RELATED WORKS

Since backpropagation became mainstream in multilayer neural networks in the 1980s [15], the rise of activation functions began. The Sigmoid activation function became standard due to its smooth gradient for learning. Historically, we can see ReLU as the dominant activation function, most likely due to its ability to prevent vanishing gradients. However, there are other activation functions such as GELU, Swish, Mish, etc. which are non-monotonic. These activation functions allow for negative values, leading to more stable training, avoiding the **dying neuron phenomenon** [11], and having better gradient flow.

Improvements were then made to ReLU that created variations of the function to address the phenomenon, which happens when a neuron in a neural network gets "stuck" and stops learning entirely. If the neuron's input becomes negative too often, due to bad initialization or large weight updates, it will always output 0. This means its gradient is also 0, it stops updating during backpropagation, and it's effectively "dead." Over time, a large number of such dead neurons can hurt model capacity and accuracy.

The **vanishing gradient problem** [6] occurs when gradients become extremely small as they are propagated backward through the layers of a deep neural network, particularly when using activation functions like Sigmoid or tanh. This causes the early layers to learn very slowly or not at all, since their weight updates diminish to near zero. In contrast, the **exploding gradient problem** arises when gradients become excessively large during backpropagation, often due to poor weight initialization or deep architectures. This leads to unstable updates, causing the model to diverge.

Advancements in activation function research have impacted neural networks as a whole, contributing to more efficient algorithms and improvements in generalization when using deep learning models [1].

### A. ReLU

the Rectified Linear Unit, is defined as:

$$\text{relu}(x) = \max(x, 0) \quad (1)$$

A linear function that maps  $x$  to itself if it is positive, and  $x$  to 0 if it is negative. It is extremely easy to compute, though it falls short in two ways: when  $x$  is negative, its gradient is 0, and ReLU is discontinuous at  $x = 0$  [3]. As  $x$  approaches  $-\infty$ , ReLU is 0, and its range is  $[0, \infty)$ . *Ranges and limits of each function in this analysis can be observed graphically in figure 2.*

### B. SoftPlus

$$\text{softplus}(x) = \ln(1 + e^x) \quad (2)$$

is a smooth approximation that approaches ReLU when the absolute value of  $x$  is large, and is continuous where ReLU is not. Unfortunately, it can be numerically unstable when  $x$  is large [3]. As  $x$  approaches  $-\infty$ , SoftPlus approaches 0, and its range is  $[0, \infty)$ .

### C. SquarePlus

presented by Barron [3],

$$\text{squareplus}(x, b) = \frac{1}{2} \left( x + \sqrt{x^2 + b} \right) \quad (3)$$

where  $b \geq 0$  defines the size of the curved region near  $x = 0$ . SquarePlus and its derivatives are algebraic and straightforward to compute, and do not produce unstable outputs. As  $x$  approaches  $-\infty$ , SquarePlus approaches 0, and its range is  $[0, \infty)$ .

**Error Function (Erf):** The next functions use and are based off of the Error Function, which is also known as the loss or objective function. It quantifies the discrepancy between the predicted output of a model and the actual, or desired output. It is defined as:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \quad (4)$$

### D. GELU

the Gaussian Error Linear Unit,

$$\text{gelu}(x) = x \cdot \frac{1}{2} \left[ 1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right) \right] \quad (5)$$

is smooth, continuously differentiable, and has a well-behaved optimization landscape. It is effective for deep learning applications that focus on boundness, feature space continuity, stationarity, and smoothness [8]. As  $x$  approaches  $-\infty$ , GELU approaches 0, and its approximated range is  $[-0.16997, \infty)$ . Its approximated lowest point is at  $-0.75179$ .

### E. ErfReLU

the Error Function Linear Unit, from Rajanand et al. [11],

$$\text{erfrelu}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha \text{erf}(x), & \text{if } x < 0 \end{cases} \quad (6)$$

is designed with the intention to solve the **dying neuron phenomenon** of ReLU at negative values. It is a piecewise combination of Erf and ReLU.  $\alpha$  is a parameter that controls the slope of the function. It is continuous, bounded below, and preserves a small amount of negative information to prevent the phenomenon. In our testing and implementation of ErfReLU, we have chosen  $\frac{\sqrt{\pi}}{2}$  as our  $\alpha$  parameter, which results in a continuous derivative. The approximated value of  $\frac{\sqrt{\pi}}{2}$  agrees with many of the trained parameters seen in ErfReLU literature [11] because they share the first 3 digits: 0.886. Due to our choice of  $\alpha$ , as  $x$  approaches  $-\infty$ , ErfReLU approaches  $-\frac{2}{\sqrt{\pi}}$ . Its range is  $[-\frac{2}{\sqrt{\pi}}, \infty)$ .

### F. SoftMax

Finally, we have SoftMax, which is a function often used in the final layer of neural networks. We will be using it in the same way in our methodology, against every activation function we're testing. It is defined from  $\mathbb{R}^K \rightarrow \mathbb{R}^K$  as:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}} \quad (7)$$

for  $i = 1 \dots K$ , where  $z = (z_1, \dots, z_K) \in \mathbb{R}^K$ . The exponential function is applied to each element  $z_i$  from the input vector  $z$ , and its resulting values are normalized by dividing the sum of all the exponentials, which guarantees that the output vector  $\text{softmax}(z)$ 's elements sum to 1 [2], making them interpretable as probabilities. Unlike the other activation functions listed above, which are used in hidden layers for binary classification or non-linear transformations, SoftMax is uniquely suited for the output layer in multi-class scenarios.

### III. MATHEMATICAL ANALYSIS

#### Our function: ErfPlus

$$\text{erfplus}(x) = \begin{cases} x, & \text{if } x > 0 \\ -x \frac{\sqrt{\pi}}{2} \text{erf}(x^{-1}), & \text{if } x \leq 0 \end{cases} \quad (8)$$

is a smooth, monotonic piecewise function. Its derivative is:

$$\frac{d}{dx} \text{erfplus}(x) = \begin{cases} 1, & \text{if } x > 0 \\ x^{-1} e^{-x^2} - \frac{\sqrt{\pi}}{2} \text{erf}(x^{-1}), & \text{if } x \leq 0 \end{cases} \quad (9)$$

As  $x$  approaches  $-\infty$ , ErfPlus approaches  $-\frac{\sqrt{\pi}}{2}$ , and its range is  $[-\frac{\sqrt{\pi}}{2}, \infty)$ .

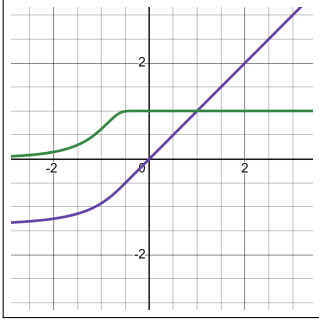


Fig. 1. ErfPlus and its first derivative

Through its piecewise implementation of a linear function and Erf, ErfPlus is bounded below, unbounded above, smooth, Lipschitz continuous, monotonically increasing with a monotonic derivative, centered at 0, and linear on the positive side, as can all be visualized in figure 1. It is less computationally effective than ReLU, but it presents many benefits. Unbounded above and linear on the positive side help to address the **vanishing gradient problem** [6]. We want the negative information to be bounded and to contribute to the model's learning, in order to address the **dying neuron phenomenon**. ReLU will simply zero the information out [6], which is why we chose to use Erf on the negative side. And, unlike ReLU, it is continuous throughout both the entire function and its derivative. It has a smooth profile, which indicates a good gradient flow and will learn more valuable information, obtaining smooth loss landscapes and easing the optimization process [6].

TABLE I  
COMPARISON OF ACTIVATION FUNCTION PROPERTIES

	Monotonic	Smoothness	Lipschitz Continuous	Zero-Centered
ReLU	✓	$C^0$	✓	✓
SoftPlus	✓	$C^\infty$	✓	✗
SquarePlus	✓	$C^\infty$	✓	✗
GELU	✗	$C^\infty$	✗	✓
ErfReLU	✓	$C^2$	✓	✓
*ErfPlus	✓	$C^\infty$	✓	✓

You can see from Table I that we chose functions with similar properties to one another for our tests. ReLU is currently the most popular [11] activation function, and thus is a good benchmark to compare to. **ErfPlus** is most similar to ErfReLU, and where it has an advantage in terms of properties is  $\infty$ -smoothness. SoftPlus and SquarePlus lack the zero-centered property that ErfPlus has. It is also both monotonic and Lipschitz continuous, where GELU, the other error function-based activation function, is not.

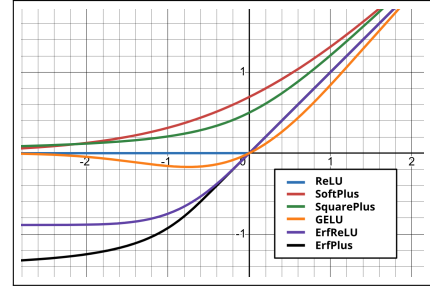


Fig. 2. Comparison of activation functions

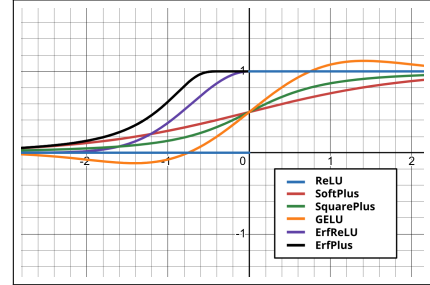












Fig. 3. Comparison of the derivatives of the activation functions

Notable observations from figures 2 and 3, in reference to the desirable properties listed in our introduction are as such:

- SoftPlus and SquarePlus are **not** zero-centered.
- ErfReLU and ErfPlus do not discard negative values, and instead let them contribute to the model.
- Note ReLU's discontinuous gradient compared to the rest.
- GELU is not Lipschitz continuous, while the rest of the absolute values of the gradients remain bounded within 0 and 1.

TABLE II  
FASHION MNIST DATASET EXAMPLES

Label	T-Shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle Boot
Example										

#### IV. METHODS

##### A. Research objectives

The objective of our study is to compare the performance of **ErfPlus** against ReLU, SoftPlus, SquarePlus, GELU, and ErfReLU.

Universally used, ReLU will act as the baseline activation function. SoftPlus and SquarePlus were developed to smoothly approximate ReLU, improve on its shortcomings, and possess the monotonicity, continuity and smoothness properties (see Table I) of activation functions that were discussed in the Introduction. However, they lack the zero-centered property that ReLU has. GELU and ErfReLU are known to perform well compared to other functions as seen in these papers [8], [11]. They utilize the error function the same way **ErfPlus** does, but GELU lacks monotonicity and Lipschitz continuity, and ErfReLU lacks smoothness (see again, Table I).

**Hypothesis:** We propose that since ErfPlus is both monotonic and smooth, unlike GELU and ErfReLU, it will achieve a superior performance in a neural network architecture designed for image classification tasks.

We will run two different experiments. The first experiment compares the 6 activation functions, with 50 trials each, on a fixed 50-layer model. The second will change the number of layers a total of 10 times, running every function with 10 trials on each layer count. The set of layer counts used is: 1, 2, 3, 5, 10, 20, 30, 50, 75, 100.

##### B. Dataset

The dataset we will be using is Fashion MNIST (see examples in table II). It consists of images of clothing articles, including an initial training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes [12]. We split the initial training set of 60,000 examples into 36,000 (60%) training examples, and 24,000 (40%) validation examples. This split is chosen at random, but will remain fixed for each experiment performed. This dataset serves as a standardized benchmark for image classification tasks and is used to assess the generalization performance of neural networks.

##### C. Model architecture

We construct a custom neural network model inspired by ResNet (Residual Networks [7]). ReLU (in figure 4) will be substituted with our activation functions for their respective trials.

The network comprises an encoder block, 50 hidden blocks, and a final decoder block, utilizing SoftMax (see figure 5). The encoder layer transforms the 28 x 28 dimensional input matrix into a 64-dimensional hidden space. The 64-dimensional vectors are then fed into  $m$  hidden blocks, each of which transforms the vectors into another 64-dimensional representation. The decoder block reconstructs the final 64-dimensional vector into a human-interpretable classification output, a 10-output vector summed to a single number from 0-9, corresponding to one of the ten clothing categories in the dataset.

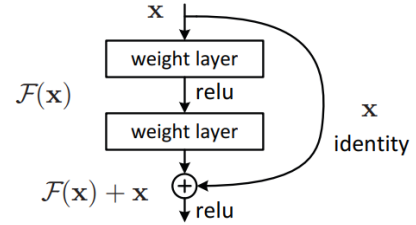


Fig. 4. Layers within a single ResNet block [7]

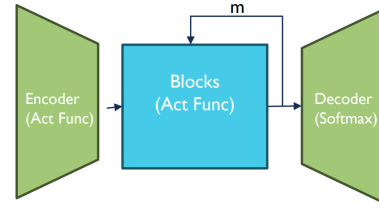


Fig. 5. Diagram of our ResNet-inspired model architecture, incorporating an encoder with activation functions, a block module with residual connections (depth  $m$ ), and a decoder utilizing Softmax for output processing.

- The **encoder block** has three layers: one converts the 28x28 image into a 784-dimensional vector, the next fully connected layer maps it to a 64-dimensional space and in the third, the activation function of choice is applied. Typically ResNet models use convolutional layers, but we have chosen to use fully connected layers for the purpose of computational speedup.
- Each **hidden block** consists of six layers. In the first layer, the data undergoes **batch normalization** to stabilize training by reducing internal co-variate shifts, followed by a fully connected layer where weights and biases are applied to extract hierarchical features. The result is then passed through one of the six activation function layers. The third and fourth layer are then the



second batch normalization, which happens before each activation [7]. This is followed by the fully connected fifth layer that passes the data through a max-pooling operation, down-sampling the feature maps to retain the most significant features and reducing computational complexity. We introduce a skip connection (see figure 4) that adds the original input of the first layer into the output of this layer. Finally, in the sixth layer, the activation function of choice is applied a second time. This sequence is repeated across all 50 hidden blocks, with weights and biases updated iteratively to optimize performance.

- The **decoder block** consists of two main layers, the first layer is a fully connected layer which maps the 64-dimensional hidden space into a human-interpretable classification output, identifying one of ten possible clothing categories. Then, a final SoftMax activation function layer is applied to normalize the results so we can compare them to one another as probabilities.

#### D. Training and optimization

For model training, we employed the **Stochastic Gradient Descent (SGD)** algorithm [9], which is a popular optimization method for training neural networks. The learning rate was varied over the course of training to allow for both rapid convergence initially and finer adjustments as training progressed. Specifically, we used a variable learning rate schedule: the learning rate started at 0.2, and was reduced after every 5 epochs according to the following sequence:

Learning rate schedule = [0.2, 0.1, 0.05, 0.01, 0.005]

This schedule allowed the model to take larger steps at the beginning of training, gradually reducing the step size to enable finer optimization as the model neared convergence.

The models were trained for a total of 25 epochs, with each learning rate applied for 5 epochs. The **cross-entropy loss function**,

$$H(p, q) = - \sum p(x) \log q(x) \quad (10)$$

was used as the objective function for training, which is well-suited for classification tasks involving probabilistic outputs. The true distribution,  $p(x)$ , is usually a one-hot encoded vector where the correct class has probability 1, and the others have 0. The predicted distribution,  $q(x)$ , is the model's predicted probability distribution over the possible outcomes (e.g., from a SoftMax output). Cross entropy effectively measures the divergence between predicted probabilities and actual labels, guiding the weight updates in the network [9]. Additionally, the log score function was utilized to assess model performance on unseen data, providing a robust comparative measure across activation functions.

The SGD optimizer was chosen for its simplicity and efficiency in large-scale machine learning tasks. The network's weights were updated at each mini-batch step based on the gradient of the loss with respect to the model parameters.

#### E. Evaluation metrics

The **score function**,

$$s(z) = -\log(1 - z) \quad (11)$$

transforms the output  $z$  to highlight how far away it is from 1, and helps evaluate how confident the model is in its incorrect predictions. It returns a **score value** that can be compared between functions. This is used to transform the accuracy of a model from the range of  $[0, 1]$  to  $[0, \infty)$ .

The **prediction difference (PD) for two models**,

$$PD_{\{u,v\}} = \frac{1}{n} \sum_{i=1}^n \frac{|f_u(x) - f_v(x)|}{f_u(x) + f_v(x)} \quad (12)$$

measures how different two model's predictions are for the same data, where a higher value means that they are making significantly different decisions [10]. To compare more than two models, we construct a **prediction difference matrix**. Each entry in the matrix represents the PD between two models, one in the row and the other in the column. This creates a symmetric matrix, where the diagonal is always zero (as a model is identical to itself). We compute the average of all values above the diagonal to provide an overall measure of how different the models are from one another.

#### F. Experimental conditions

Performance is assessed using the mean squared error metric, focusing on the error derived from the test sets. While training error indicates how well the model learns from seen data, our primary concern is the *generalization error*, how well the model performs on unseen data. To ensure fair and unbiased evaluation, all models were trained under identical conditions, including **fixed random seeds** for weight initialization, consistent learning rates (with variations examined for additional analysis), and identical network structures. This setup guarantees that the only variable modified throughout the experiments within each of the 50 intermediate layers is the activation function, ensuring a fair comparison.

#### G. Statistical hypothesis testing

To determine whether the differences in performance between activation functions are statistically significant, we apply the Wilcoxon signed-rank test, a non-parametric alternative to the paired t-test. This test is suitable for comparing matched samples that do not necessarily follow a normal distribution, which is often the case with model performance metrics across trials. For each pairwise comparison between functions,

- **Null hypothesis ( $H_0$ ):** There is no significant difference in performance between ErfPlus, and the activation function currently being tested; their performance distributions are identical.
- **Alternative hypothesis ( $H_1$ ):** There is a statistically significant difference in performance between the two activation functions; their performance distributions differ.

TABLE III  
FINAL TEST SCORE VALUES

Function	Mean	Median	Max	SD
ReLU	1.81965	1.92565	1.99731	0.17164
SoftPlus	1.74453	1.72795	2.04330	0.23527
SquarePlus	1.67189	1.65340	2.05729	0.29013
GELU	1.94409	2.00211	2.03256	<b>0.11838</b>
ErfReLU	1.96814	<b>2.03868</b>	<b>2.07864</b>	0.14265
*ErfPlus	<b>1.98329</b>	2.03294	2.06672	0.12570

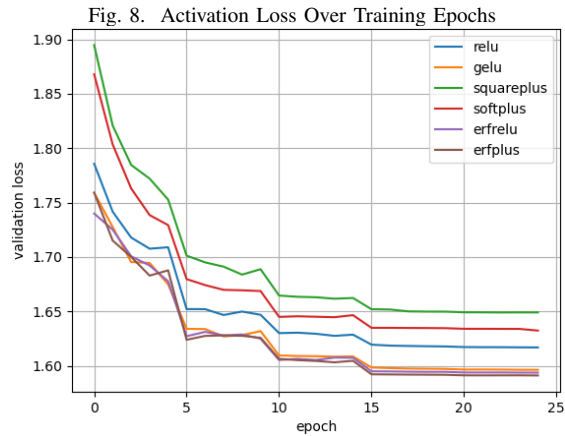
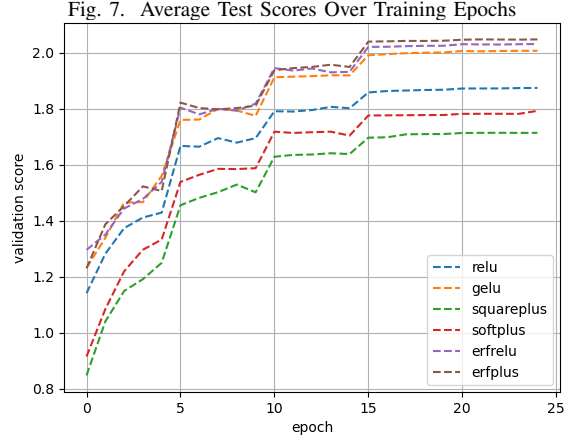
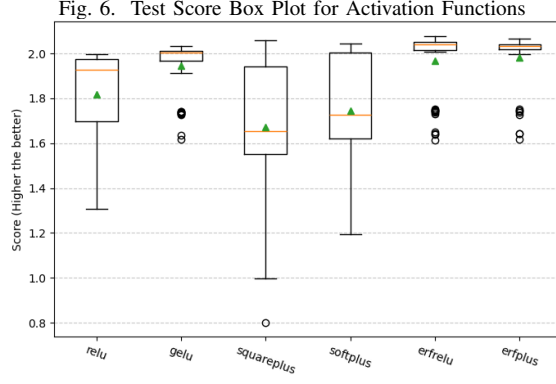


TABLE IV  
PREDICTION DIFFERENCE (PD) VALUES

Function	Mean	Median	Max	SD
ReLU	0.33887	0.27012	<b>0.70986</b>	0.11674
SoftPlus	0.47784	0.48400	0.79128	0.16960
SquarePlus	0.60537	0.53372	1.29313	0.24558
GELU	0.34813	0.27602	0.73277	0.12000
ErfReLU	0.32682	<b>0.25628</b>	0.71649	0.12078
*ErfPlus	<b>0.31518</b>	0.25896	0.71743	<b>0.11373</b>

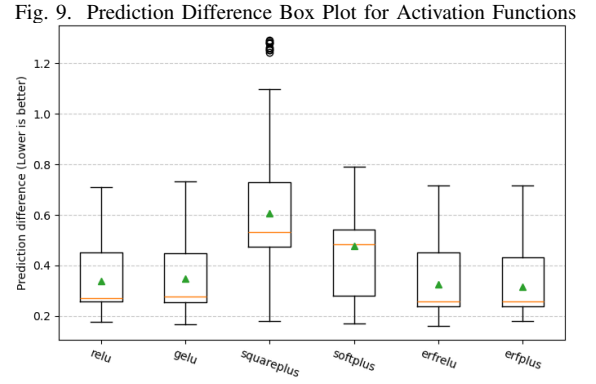


Fig. 10. Experiment 2: Average Test Scores Using Hidden Blocks

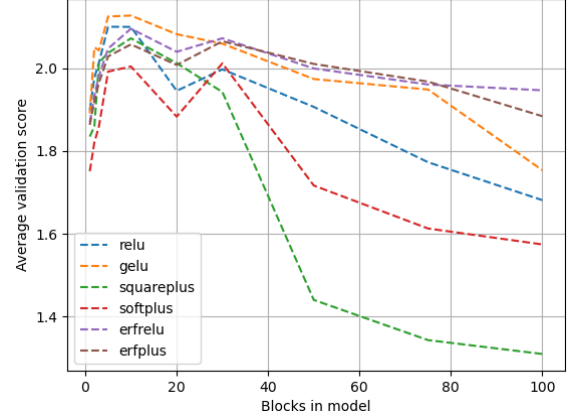


TABLE V  
P-VALUES CALCULATED USING WILCOXON SIGNED-RANK TEST  
AGAINST ERFPLUS

Function	P-Value
ReLU	0.000000082
SoftPlus	0.00026
SquarePlus	0.0000024
GELU	0.000000015
ErfReLU	0.56

The Wilcoxon signed-rank test is applied to the paired performance scores obtained across multiple trials for each function. A significance level of  $\alpha = 0.05$  is used to determine

whether to reject the null hypothesis. P-values (D) below this threshold indicate that the observed differences are unlikely to have occurred by chance, supporting the alternative hypothesis.

## V. RESULTS

Our results showed that ErfPlus has the highest mean test score of 1.98329 and the best mean prediction difference (PD) value of 0.31518. It also has very similar median values to ErfReLU, which recorded a median test score of 2.03868 and a PD median of 0.25628. ErfReLU just barely outperforms ErfPlus here.

When we look closer at the statistics of tables III and IV, ErfPlus performed very similarly to ErfReLU. ErfReLU achieved a better median and maximum test score, 2.03868 and 2.07864, respectively, compared to ErfPlus, which had a median of 2.03294 and a maximum of 2.06672. For PD, it has a better median of 0.25628, whereas ErfPlus had a median of 0.25896. Notably, GELU has the lowest standard deviation (SD) for test scores at 0.11838, and ReLU has the lowest maximum PD value at 0.70986, with and a very low SD at 0.11674. Still, the three error function based functions have similar results to one another. Their SD values for both metrics were also the lowest of the 6 activation functions tested. In the box plots (figures 6 and 9), we can observe high test scores with low variance for GELU, ErfReLU and ErfPlus. ReLU’s box plot for PD is similar to the functions based on the error function, where it is more noteworthy to observe the poor performance of SquarePlus.

See figures 7 and 8, which illustrate how effectively each model learns and adapts over time. Lower loss values are indicative of better generalization and convergence. Here, we can observe the consistently high score performance, low loss metrics, and accuracy of ErfPlus. ErfReLU and GELU also perform well, with SquarePlus notably behind in each measurement. The same trends of the tables and box plots can be seen in these graphs, where the general performance order from lowest to highest is SquarePlus, SoftPlus, ReLU, GELU, ErfReLU, then ErfPlus.

The first experiment evaluates our activation functions by running 50 trials each on a fixed 50-layer model. The second experiment varies the number of layers, using the set  $\{1, 2, 3, 5, 10, 20, 30, 50, 75, 100\}$ , and runs 10 trials for each activation function at every layer count.

Figure 10 is the only plot that uses data from Experiment 2. It displays average test scores, with each score calculated after 25 training epochs. Consistent with trends observed in existing literature [6], performance initially improves as more layers are added, peaking between 5 and 10 blocks. Beyond this point, increasing the number of blocks led to a decline in performance, notably with SquarePlus.

## VI. DISCUSSION

### Experiment discussion

ErfReLU, ErfPlus, and GELU exhibiting low SD values in their final test scores (table III), and low average PD values (table IV) could signify that activation functions that utilize

the error function offer high precision and are less sensitive to random weight initialization. ErfPlus serves as a smooth alternative to ErfReLU. Both yield similarly high accuracy, strong precision, and demonstrate robustness to changes in weight initialization and network depth, outperforming other activation functions in these aspects. In contrast, SquarePlus and SoftPlus perform relatively poorly, despite being smooth and monotonic. A likely reason is that these functions are not centered around zero, a property that helps avoid bias shift and improves gradient flow during training, which may affect learning dynamics and model convergence. Their high PD and SD values further emphasize their lack of reliability across trials. ReLU is centered at zero and performs well, especially in PD metrics (see table IV and figure 9), suggesting that it is robust in terms of prediction consistency, even if its overall accuracy is lower. This aligns with its reputation as a reliable baseline in deep networks.

While ErfReLU has the best maximum validation score and median values, ErfPlus has higher values in mean performance and consistency. This suggests ErfPlus is stable and dependable across different trials and initializations, even if it doesn’t always achieve the highest performance in terms of maximums. GELU performs similarly to the other error function activation functions in terms of validation score, but it tends to have slightly higher PD scores and more variance, especially under deeper architectures in Experiment 2. This could imply that while GELU performs well on average, its predictions are more sensitive to the random seed or architecture depth, and may also signify the importance of Lipschitz continuity and monotonicity that GELU lacks.

Additionally, the contrast between Experiments 1 and 2 is valuable:

- In Experiment 1, most activation functions have enough room (trials  $\times$  layers) to show their true potential, and thus produce consistent results.
- In Experiment 2, when the network depth increases (50–100 layers), only ErfReLU maintains strong, consistent performance. Where ErfReLU have scored evenly in other metrics, this plot shows a more notable delta in favor of ErfReLU. This suggests it may be more scalable and depth-tolerant, which is critical for deeper networks.

Experiment 2 results indicate that deeper architectures tend to overfit when trained for a limited number of epochs. Notably, while most activation functions show a pronounced drop in performance after the peak, ErfReLU and ErfPlus exhibit a slower decline, suggesting enhanced robustness and reduced susceptibility to overfitting, even in deeper networks.

### P-value discussion

Finally, table V presents the p-values obtained through the Wilcoxon signed-rank test, comparing ErfPlus against the other 5 activation functions evaluated in this study. For all activation functions, excluding ErfReLU, the p-values were found to be below the conventional significance threshold of 0.05. This indicates that the performance differences between ErfPlus and these four activation functions (ReLU, SoftPlus,

SquarePlus, and GELU) are statistically significant and are unlikely to have occurred by chance. Consequently, this supports the conclusion that ErfPlus offers a measurable improvement in performance over these alternatives.

In contrast, the p-value obtained from the comparison between ErfPlus and ErfReLU was 0.57, well above the 0.05 threshold. This suggests that the performance difference between these two activation functions is not statistically significant, and any observed variation in results may be attributed to random fluctuation. Therefore, ErfPlus and ErfReLU can be considered to perform similarly under the conditions evaluated in this experiment.

Given its low PD, high average test score, minimal variance, and p-value results with the other functions, **ErfPlus** stands out as a balanced and practical choice across specific scenarios that require both accuracy and stability.

## VII. FUTURE WORK

Our discussion poses a few questions that could be explored in the future:

- We can experiment with other activation functions that look similar, i.e. Leaky ReLU, Swish, Mish, etc., in order to compare different properties with ErfPlus.
- It is possible that some of the activation functions used in this study are not implemented in their optimized form. This discrepancy could lead to an unfair comparison, and it might be of interest to find and compare the optimized versions against each other.
- Should we examine performance under noisy inputs or data augmentation? Would the more stable functions like ErfPlus retain their lead?
- We should explore computational cost vs. performance. Are the small gains from ErfPlus worth it over ReLU, GELU or ErfReLU in resource-constrained scenarios?

The last question came up during our methodology and process, as we chose to put less emphasis on computational complexity when ErfPlus was designed, in order to focus on creating a function that possessed the other properties in table I.

When it came to implementation, we faced a few limitations regarding optimization. Our activation function was coded in a naive approach and while functional, there are ways to optimize computation and improve performance.

- Instead of relying on built-in libraries, it might be of interest to build a polynomial or rational function **approximation** of **ErfPlus** to reduce computation time while maintaining accuracy.
- **Parametric activation functions** have proven to be effective in novel discoveries. They can be customized automatically, result in reliable improvements in performance [4], and could be a very good enhancement to consider for ErfPlus.

### A. Experimentation and robustness

1) *Increase trial count for experiment 1::* In our initial experiments, we conducted 50 trials per activation function across 6 different activation functions, resulting in a total training time of **412 minutes**. Increasing the number of trials would enhance the statistical significance and reduce variance in results and findings. For robustness, it would be effective to use a larger sample size (run more tests over more time) to evaluate performance across different runs.

2) *Expanding trial set per result for experiment 2::* Similarly, in experiment 2, adding more trials would help ensure that performance trends are not due to random chance. Eliminating chance provides stronger empirical evidence for the effectiveness of ErfPlus.

### B. Expanding to more models and tasks

Activation functions can behave differently depending on the network type and the specific problem being solved. Our experiments have been primarily focused on a limited set of neural network architectures and tasks thus far, so to gain a broader understanding of its effectiveness, we propose testing our activation function with a variety of deep learning models:

1) *Convolutional neural networks (CNNs):* CNNs are widely used for image classification and detection tasks. Evaluating ErfPlus in CNNs would show it how handles spatial feature extraction.

2) *Recurrent neural networks (RNNs):* Since RNNs process sequential data, (e.g., time series, speed, and natural language processing tasks), this would help us test ErfPlus' performance in capturing long-term dependencies

3) *Text prediction and natural language processing (NLP) tasks:* Since many NLP models rely on activation functions for non-linear transformations, evaluating ErfPlus in tasks like sentiment analysis, machine translation, and text generation would highlight its performance in handling language data.

4) *Reinforcement learning (RL):* Activation functions play a role in policy learning and value approximation in RL algorithms. Testing ErfPlus in RL tasks could provide insights into stability and convergence behaviour.

## VIII. CONCLUSION

Our study presents a comprehensive evaluation of a novel error function-based activation function, ErfPlus, in comparison to several established alternatives within deep residual neural networks. Across a series of experiments, ErfPlus consistently demonstrated strong performance, achieving the best mean validation score and the lowest prediction difference with minimal variance across trials. In particular, ErfPlus exhibited robust generalization, stability across weight initializations, and resilience when scaling to deeper network architectures.

ErfPlus performed comparably to other error-function-based activations, such as GELU and ErfReLU, all of which outperformed ReLU, SoftPlus, and SquarePlus in both accuracy and consistency. Notably, despite their smooth and monotonic properties, SoftPlus and SquarePlus suffered from high



PD scores and greater variability, likely due to their non-centered outputs. In contrast, the centering, monotonicity and continuous differentiability of ErfPlus contributed to its improved convergence behavior and model precision. These findings suggest that ErfPlus offers a compelling alternative to ErfReLU, and other commonly used activation functions, particularly in deep learning tasks where precision, stability, and resistance to initialization sensitivity are critical.

## IX. CODE AVAILABILITY

An open-source code repository for this work is available on GitHub:

<https://github.com/JJ0BY/Erfplus-Activation-function>

## REFERENCES

- [1] Andrea Apicella, Francesco Donnarumma, Francesco Isgrò, and Roberto Prevete. A survey on modern trainable activation functions. *Neural Networks*, 138:14–32, June 2021.
- [2] Kunal Banerjee, Vishak Prasad C, Rishi Raj Gupta, Karthik Vyas, Anushree H, and Biswajit Mishra. Exploring alternatives to softmax function. *CoRR arXiv*, November 2020.
- [3] Jonathan T. Barron. Squareplus: A softplus-like algebraic rectifier. December 2021.
- [4] Garrett Bingham and Risto Miikkulainen. Discovering parametric activation functions. *Neural Networks*, 148:48–65, April 2022.
- [5] Koushik Biswas, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Erfact and pserf: Non-monotonic smooth trainable activation functions. *CoRR arXiv*, March 2022.
- [6] Mostafa Z. Ali Heba Abdel-Nabi, Ghazi Al-Naymat and Arafat Awajan. Hclsh: A novel non-linear monotonic activation function for deep learning methods. *IEEE Access*, 11:47794–47815, May 2023.
- [7] Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun. Deep residual learning for image recognition. December 2015.
- [8] Minhyeok Lee. Gelu activation function in deep learning: A comprehensive mathematical analysis and performance. August 2023.
- [9] Li Li, Miloš Doroslovački, and Murray H. Loew. Approximating the gradient of cross-entropy loss function. *IEEE Access*, 8:111626–111635, May 2020.
- [10] Dong Lin and Gil I. Shamir. Real world large scale recommendation systems reproducibility and smooth activations. *CoRR arXiv*, February 2022.
- [11] Ashish Rajanand and Pradeep Singh. Erfrelu: adaptive activation function for deep neural network. *Pattern Analysis and Applications*, May 2024.
- [12] Zalando Research. Fashion-mnist, an mnist-like dataset of 70,000 28x28 labeled fashion images, 2017.
- [13] Davor Runje and Sharath M Shankaranarayana. Constrained monotonic neural networks. *Proceedings of the 40th International Conference on Machine Learning*, 2023.
- [14] Mohamed Khalil-Hani Shan Sung Liew and Rabia Bakhteri. Bounded activation functions for enhanced training stability of deep neural networks on visual pattern recognition problems. *Neurocomputing*, 216, December 2016.
- [15] Barry J. Wythoff. Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, 18:115–155, February 1993.

## APPENDIX

### A. Smoothness

A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is **k-smooth** if:

$$f \in C^k(\mathbb{R}) \iff f \text{ has continuous derivatives up to order } k. \quad (13)$$

A function is **smooth** if it is  $\infty$ -smooth, i.e.:

$$f \in C^\infty(\mathbb{R}) \iff \text{all derivatives exist and are continuous.} \quad (14)$$

### B. Continuity

A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is **continuous at a point**  $x_0$  if:

$$\forall \epsilon > 0, \exists \delta > 0 \text{ such that } |x - x_0| < \delta \Rightarrow |f(x) - f(x_0)| < \epsilon. \quad (15)$$

A function is **continuous** if  $\forall x_0 \in \mathbb{R}$ , the function is continuous at  $x_0$ .

### C. Lipschitz Continuity

A function  $f : \mathbb{R} \rightarrow \mathbb{R}$  is **Lipschitz continuous** with constant  $L > 0$  if:

$$\exists L > 0 \text{ such that } \forall x, y \in \mathbb{R}^n, \quad |f(x) - f(y)| \leq L|x - y|. \quad (16)$$

### D. P-Value

A **p-value** is a statistical measure that indicates the probability of obtaining results at least as extreme as the observed results, assuming that the null hypothesis is true. A lower p-value (typically less than 0.05) suggests that the observed difference is statistically significant and unlikely to have occurred by chance.