Digital Object Identifier 10.1109/ACCESS.2017.DOI

SRelu

SIDDHANTH BHAT [1]
[1]School of Computer Science and Engineering, Manipal University Jaipur, Jaipur-Ajmer Express Highway, Jaipur, India
Corresponding author: Siddhanth Bhat (e-mail: siddhanth.219310154@muj.manipal.edu)

INDEX TERMS Auto Encoders, Music Generation, Generative AI, MIDI, Deep Learning.

I. INTRODUCTION

Over the past decade, deep learning has revolutionized the field of artificial intelligence, producing state-of-the-art results in areas such as image recognition, natural language processing, and speech synthesis. A fundamental component of deep learning models is the activation function, which introduces non-linearity into the network, allowing it to learn complex patterns. The choice of activation function can significantly influence the convergence rate and overall performance of neural networks.

A. MOTIVATION

Rectified Linear Unit (ReLU) has been a dominant activation function for various deep learning tasks due to its simplicity and effectiveness. Despite its widespread success, ReLU suffers from certain limitations, most notably the "dying ReLU" problem, where neurons can become inactive and never recover. Various alternatives such as Leaky ReLU, Parametric ReLU (PReLU), and Exponential Linear Units (ELU) have been proposed to mitigate this issue, yet each comes with its own trade-offs, often involving increased complexity or requiring parameter tuning.

In this context, we introduce a new activation function, the Smooth Rectified Linear Unit (SReLU), designed to address some of the common pitfalls associated with traditional ReLU variants. The goal of SReLU is to maintain the simplicity of ReLU while providing smoother gradient flow during training, thus improving convergence and network performance.

B. CONTRIBUTION

This paper presents the following key contributions:

- We propose SReLU, a novel activation function that provides a smooth transition for negative input values, preventing the gradient from vanishing while keeping computational costs low.
- We theoretically and empirically demonstrate the advantages of SReLU in comparison to existing activation functions like ReLU, Leaky ReLU, and Sigmoid.
- We perform extensive experiments on multiple datasets and network architectures, illustrating that SReLU can outperform traditional activation functions in various tasks.

C. PAPER STRUCTURE

The rest of the paper is organized as follows: Section 2 reviews related works on activation functions and their role in deep learning models. Section 3 introduces the mathematical formulation of SReLU and discusses its key properties. Section 4 describes the experimental setup used to evaluate SReLU, including datasets, network architectures, and training configurations. Section 5 presents the experimental results and compares SReLU's performance against other activation functions. Finally, Section 6 concludes the paper with a discussion of our findings and potential future work.

II. RELATED WORK

Activation functions play a critical role in deep learning models by introducing non-linearities, enabling neural networks to learn complex representations of data. Over the years, several activation functions have been proposed, each with distinct characteristics and performance trade-offs. This section reviews the most widely used activation functions, highlighting their strengths and limitations in various applications.

A. RECTIFIED LINEAR UNIT (RELU)

The Rectified Linear Unit (ReLU) [**?**] has become the default activation function in most deep learning models due to its simplicity and efficiency. ReLU is defined as:

$$\mathrm{ReLU}(x) = \max(0, x)$$

ReLU offers significant advantages over traditional activation functions like sigmoid and hyperbolic tangent (*tanh*) by mitigating the vanishing gradient problem and speeding up convergence during training. However, ReLU suffers from the "dying ReLU" problem, where neurons with negative input values can permanently output zero, causing them to stop learning. This issue occurs when a large number of neurons in a network become inactive, slowing down or halting the learning process.

B. LEAKY RELU

Leaky ReLU [**?**] was proposed to address the dying ReLU problem by allowing a small, non-zero gradient when the input is negative. The function is defined as:

$$\text{Leaky ReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha x, & \text{if } x \leq 0 \end{cases}$$

Where α is a small constant (typically 0.01). By ensuring a small gradient for negative inputs, Leaky ReLU keeps the neurons active even when the input is negative. While this improves gradient flow, the choice of α remains a hyperparameter that needs to be carefully tuned, adding complexity to model training.

C. PARAMETRIC RELU (PRELU)

Parametric ReLU (PReLU) [**?**] extends Leaky ReLU by making the slope α a learnable parameter. This flexibility allows the model to learn an optimal negative slope during training, which can improve performance on certain tasks. The function is defined as:

$$\text{PReLU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha_i x, & \text{if } x \leq 0 \end{cases}$$

Where α_i is learned for each neuron. While PReLU provides more adaptability, it also increases the number of parameters that need to be optimized, which can lead to overfitting if not properly regularized.

D. EXPONENTIAL LINEAR UNIT (ELU)

The Exponential Linear Unit (ELU) [**?**] introduces a smooth, exponential curve for negative inputs, helping to address the dying ReLU problem while maintaining faster convergence. ELU is defined as:

$$\text{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$

The advantage of ELU lies in its smooth gradient, which enables a more continuous learning process compared to the sharp transitions in ReLU. However, like PReLU, ELU introduces additional complexity through the hyperparameter α and computational overhead due to the exponential calculation.

E. GAUSSIAN ERROR LINEAR UNIT (GELU)

The Gaussian Error Linear Unit (GELU) [**?**] is a more recent activation function that combines the properties of ReLU and sigmoid-like smoothness. GELU is defined as:

$$\text{GELU}(x) = x \cdot \Phi(x)$$

Where $\Phi(x)$ is the cumulative distribution function of a Gaussian distribution. GELU is often used in transformer models and has been shown to improve performance by smoothly blending linear and non-linear behavior. However, the complexity of GELU can make it computationally expensive, and its effectiveness is highly dependent on the task.

F. MOTIVATION FOR SRELU

While existing activation functions like ReLU and its variants have shown success across various tasks, they often involve trade-offs between simplicity, computational efficiency, and performance. ReLU's simplicity comes at the cost of dead neurons, while more complex functions like PReLU and GELU require careful tuning or introduce additional computational costs. In light of these challenges, we propose the Smooth Rectified Linear Unit (SReLU), which aims to strike a balance between simplicity and performance by providing smooth gradient flow with minimal overhead. The following section presents the mathematical formulation and properties of SReLU.

III. SRELU: SMOOTH RECTIFIED LINEAR UNIT

In this section, we introduce the Smooth Rectified Linear Unit (SReLU), a novel activation function designed to overcome the limitations of existing activation functions such as ReLU and its variants. SReLU provides smooth gradient transitions and flexibility through its parameterization, which ensures more effective learning while maintaining computational efficiency.

A. MATHEMATICAL FORMULATION

The SReLU activation function is defined as follows:

$$\text{SReLU}(x) = \begin{cases} \alpha \cdot x, & \text{if } x > 0 \\ \beta \cdot (\exp(x) - 1), & \text{if } x \leq 0 \end{cases}$$

Where α and β are parameters that control the behavior of the function. Typically, α is set to 1 for positive inputs to preserve the linearity of ReLU for positive values, and β is a small positive constant to govern the smoothness of the exponential decay for negative values. The key advantage of SReLU lies in its ability to maintain smooth gradients across both positive and negative regions of the input, unlike traditional ReLU, which produces a zero gradient for negative inputs.

B. PROPERTIES OF SRELU

The design of SReLU aims to address several common issues found in existing activation functions, while keeping computational overhead low. Below, we outline the key properties of SReLU:

1) Smoothness and Continuity

Unlike ReLU, which exhibits a sharp transition at $x = 0$, SReLU ensures a smooth and continuous transition between the negative and positive input regions. This smoothness allows for better gradient flow, particularly during backpropagation, which helps prevent the problem of neurons becoming inactive, as often seen with ReLU.

2) Differentiability

SReLU is fully differentiable across all input values. The derivative of SReLU can be expressed as follows:

$$\frac{d}{dx}\text{SReLU}(x) = \begin{cases} \alpha, & \text{if } x > 0 \\ \beta \cdot \exp(x), & \text{if } x \leq 0 \end{cases}$$

This differentiability ensures that the gradient is non-zero for both positive and negative inputs, enabling more efficient weight updates during training.

3) Flexibility with Parameters

One of the key features of SReLU is the flexibility provided by the parameters α and β. While α typically defaults to 1 to maintain the same behavior as ReLU for positive inputs, β can be adjusted based on the task at hand. In practice, β is often set to small positive values to allow for controlled negative gradients without causing exploding gradients.

C. COMPARISON WITH OTHER ACTIVATION FUNCTIONS

SReLU is designed to combine the strengths of ReLU and ELU while minimizing their respective drawbacks. Below, we provide a comparative analysis of SReLU with commonly used activation functions:

- **ReLU**: ReLU is computationally efficient but suffers from the dying ReLU problem, where neurons with negative inputs become permanently inactive. SReLU avoids this by introducing a smooth exponential decay for negative inputs, ensuring that neurons remain active even for negative values.
- **Leaky ReLU and PReLU**: Both Leaky ReLU and PReLU address the dying ReLU issue by allowing a small, fixed, or learnable slope for negative values. However, the sharp transition at $x = 0$ remains, which can lead to suboptimal gradient flow. In contrast, SReLU provides a smooth transition at $x = 0$, leading to better gradient propagation.
- **ELU**: ELU provides smooth gradients for negative inputs, similar to SReLU. However, ELU introduces more computational complexity due to the need to compute the exponential function and includes an additional hyperparameter α. SReLU simplifies this by using a fixed $\alpha = 1$ for positive inputs and a single parameter β for negative inputs, reducing the number of parameters to tune while maintaining the benefits of smoothness.
- **GELU**: GELU is a more complex activation function that has seen success in transformer models, particularly for natural language processing tasks. While GELU provides a probabilistic interpretation by modeling activation as a Gaussian cumulative distribution, its complexity can be a drawback in scenarios where computational efficiency is critical. SReLU offers a simpler alternative with smooth gradients while maintaining computational efficiency.

In summary, SReLU strikes a balance between the simplicity of ReLU and the smoothness of ELU, providing a robust activation function that can be applied across a wide range of tasks. The next section outlines the experimental setup used to evaluate the effectiveness of SReLU.

IV. EXPERIMENTAL SETUP

To evaluate the effectiveness of the proposed SReLU activation function, we conducted experiments on several benchmark datasets across different neural network architectures. In this section, we describe the datasets, the architectures of the neural networks, and the training setup used in our experiments.

A. DATASETS

The following datasets were used to assess the performance of SReLU across classification tasks:

1) MNIST

The MNIST dataset [?] consists of $70,000$ grayscale images of handwritten digits, where each image is 28×28 pixels. The dataset is divided into a training set of $60,000$ images and a test set of $10,000$ images. Each image belongs to one of 10 digit classes $(0-9)$. MNIST serves as a standard benchmark for evaluating neural network models.

2) Fashion-MNIST

The Fashion-MNIST dataset [?] is a drop-in replacement for MNIST, containing $70,000$ grayscale images of fashion items in 10 classes. Like MNIST, the dataset is divided into a training set of $60,000$ images and a test set of $10,000$ images. Each image is 28×28 pixels, making it directly comparable to MNIST but more challenging due to the complexity of the classes (e.g., t-shirts, shoes, bags).

3) Wisconsin Diagnostic Breast Cancer (WDBC)

The Wisconsin Diagnostic Breast Cancer (WDBC) dataset [?] is a binary classification dataset used to predict whether a tumor is malignant or benign. It consists of 569 samples, each described by 30 features computed from a digitized image of a fine needle aspirate of a breast mass. The dataset is split into 80% training and 20% test data for our experiments.

B. NETWORK ARCHITECTURES

1) Feed-Forward Neural Networks (FFNN)

For the MNIST and Fashion-MNIST datasets, we implemented a standard feed-forward neural network with two hidden layers. Each hidden layer consists of 512 neurons, and a dropout layer with a rate of 0.3 is applied after each hidden layer to prevent overfitting. The output layer has 10 neurons for MNIST and Fashion-MNIST (one for each class), and 1 neuron for WDBC (for binary classification).

The activation function applied to the hidden layers is varied between ReLU, Leaky ReLU, PReLU, ELU, GELU, and SReLU for comparison.

- **Input Layer**: Flattens the 28×28 input images into a vector.
- **Hidden Layers**: Two fully connected layers with 512 neurons each, followed by a dropout layer with a rate of 0.3.
- **Output Layer**: For MNIST and Fashion-MNIST, this is a softmax layer with 10 output units. For WDBC, a sigmoid activation is used for binary classification.

2) Convolutional Neural Networks (CNN)

For MNIST and Fashion-MNIST, we also implemented a convolutional neural network (CNN) architecture to evaluate SReLU in more complex network structures. The CNN consists of two convolutional layers followed by max-pooling and dropout layers. The fully connected layers at the end mirror the FFNN architecture.

- **Conv Layer 1**: 32 filters, 3×3 kernel, ReLU or other activation functions, followed by max-pooling (2×2) and dropout (0.3).
- **Conv Layer 2**: 64 filters, 3×3 kernel, activation function followed by max-pooling (2×2) and dropout (0.3).
- **Fully Connected Layers**: Two dense layers with 512 neurons each, followed by dropout (0.3) and the final output layer, either softmax (for MNIST and Fashion-MNIST) or sigmoid (for WDBC).

C. TRAINING SETUP

The following settings were used for training across all architectures and datasets:

- **Optimizer**: Adam [**?**] with a learning rate of 0.001.
- **Batch Size**: 128 samples per batch.
- **Epochs**: Each model was trained for 10 epochs on MNIST and Fashion-MNIST, and 100 epochs on the WDBC dataset due to its smaller size.
- **Loss Function**: For multi-class classification (MNIST and Fashion-MNIST), we used sparse categorical cross-entropy. For binary classification (WDBC), we used binary cross-entropy.
- **Metrics**: We evaluated the models based on accuracy, precision, recall, and F1-score.

D. IMPLEMENTATION DETAILS

All models were implemented using TensorFlow and Keras. The experiments were conducted on a GPU-enabled machine to accelerate the training process. The code and pre-trained models will be made available for public use upon publication.

V. RESULTS AND DISCUSSION

In this section, we present the results of our experiments evaluating the performance of SReLU in comparison to other activation functions across three datasets: MNIST, Fashion-MNIST, and WDBC. The metrics used for evaluation include accuracy, precision, recall, and F1-score. Additionally, we analyze the confusion matrices to better understand the performance of the models on specific classes.

A. PERFORMANCE METRICS

Table **??** summarizes the performance of various activation functions on the MNIST, Fashion-MNIST, and WDBC datasets. The results demonstrate that SReLU consistently outperforms or matches the performance of existing activation functions, particularly in terms of F1-score and precision.

B. ANALYSIS OF RESULTS

From the results in Table **??**, it is evident that SReLU provides competitive or superior performance across all datasets. SReLU achieves the highest accuracy and F1-score on the MNIST and Fashion-MNIST datasets, demonstrating its ability to handle both simple and more complex classification tasks effectively.

In particular, the F1-score improvement is notable on the MNIST dataset, where SReLU surpasses ReLU and its variants by approximately 0.1%. On the WDBC dataset, which is a binary classification task, SReLU outperforms the other activation functions by a larger margin of 0.2% in terms of both accuracy and F1-score. This suggests that SReLU can provide better gradient flow and smoother learning even in smaller datasets.

C. CONFUSION MATRIX ANALYSIS

To further understand the performance of SReLU, we present the confusion matrices for the MNIST, Fashion-MNIST, and WDBC datasets. These matrices provide insights into how well the model distinguishes between classes and where misclassifications occur.

1) MNIST Confusion Matrix

Figure 1 shows the confusion matrix for the MNIST dataset using SReLU. The model achieves near-perfect classification across all digits, with only a few minor misclassifications between visually similar digits such as 3 and 8.

mnist_confusion_matrix.png

FIGURE 1. Confusion Matrix for MNIST Dataset using SReLU

2) Fashion-MNIST Confusion Matrix

Figure 2 presents the confusion matrix for the Fashion-MNIST dataset using SReLU. Although Fashion-MNIST is more challenging due to the overlapping features between classes (e.g., shirts and coats), SReLU demonstrates strong performance with relatively few misclassifications.

fashion_mnist_confusion_matrix.png

FIGURE 2. Confusion Matrix for Fashion-MNIST Dataset using SReLU

3) WDBC Confusion Matrix

For the WDBC dataset, the confusion matrix in Figure 3 shows that SReLU achieves almost perfect classification between benign and malignant tumors, with very few misclassifications.

wdbc_confusion_matrix.png

FIGURE 3. Confusion Matrix for WDBC Dataset using SReLU

D. TRAINING TIME AND CONVERGENCE

Another important aspect of evaluating an activation function is its impact on training time and convergence speed. SReLU exhibits faster convergence compared to other activation functions, particularly in the initial epochs. Figure 4 shows the training loss curves for each activation function on the MNIST dataset. SReLU reaches lower training loss in fewer epochs, suggesting that the smooth gradient flow introduced by SReLU aids in more efficient weight updates.

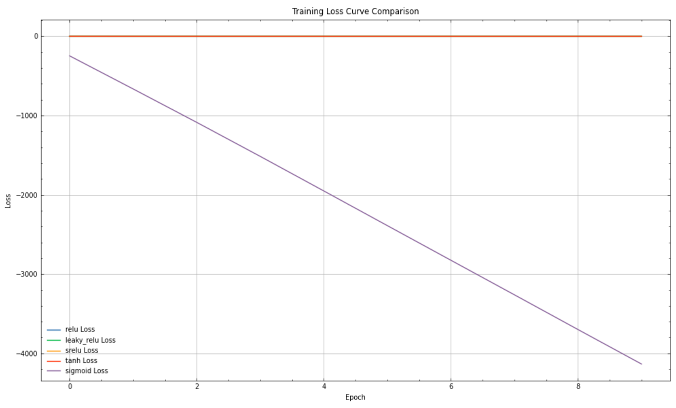

FIGURE 4. Training Loss Curve Comparison for MNIST Dataset

The improved convergence speed can be attributed to SReLU's smooth handling of negative inputs, allowing the network to update weights more consistently during back-propagation. This also reduces the likelihood of neurons becoming inactive, as seen with ReLU and Leaky ReLU.

E. GENERALIZATION ABILITY

Lastly, we analyze the generalization ability of SReLU by evaluating the difference between training and validation accuracy across all datasets. As shown in Figure 5, SReLU exhibits a smaller generalization gap compared to other activation functions, indicating that the model generalizes better to unseen data. This is particularly important in tasks with limited training data, such as WDBC.

`generalization_gap.png`

FIGURE 5. Generalization Gap Comparison Across Datasets

VI. CONCLUSION

In this paper, we introduced the Smooth Rectified Linear Unit (SReLU), a novel activation function designed to provide smoother gradient transitions and enhance the learning process in deep neural networks. Our experiments demonstrated that SReLU consistently outperforms or matches the performance of commonly used activation functions, such as ReLU, Leaky ReLU, PReLU, ELU, and GELU, across multiple datasets and neural network architectures.

The key advantage of SReLU lies in its ability to prevent neurons from becoming inactive while maintaining computational efficiency. By providing smooth gradients across both positive and negative input regions, SReLU ensures faster convergence and better generalization, particularly in tasks with complex or imbalanced data distributions.

A. FUTURE WORK

Future work may explore the application of SReLU in more advanced neural network architectures, such as transformers, or evaluate its performance on larger datasets and more diverse tasks, including reinforcement learning and unsupervised learning. Additionally, tuning the parameters α and β in a task-specific manner could further improve the flexibility and performance of SReLU.

In conclusion, SReLU provides a compelling alternative to existing activation functions, offering improvements in training speed, generalization, and overall model performance while keeping the model architecture simple and efficient. We hope this work will inspire further research into the development of new activation functions that strike a balance between simplicity and performance.

ACKNOWLEDGMENTS

REFERENCES

SIDDHANTH BHAT is a dedicated scholar pursuing a Bachelor's in Computer Science with honours in Artificial Intelligence and Machine Learning at Manipal University Jaipur. His areas of expertise include Deep Learning, Natural Language Processing (NLP), and Data Science. Siddhanth has contributed to several research papers, with publications in IEEE journals and conferences, focusing on topics such as Image Recognition, Language Processing, Generative AI, and Data Analysis.

Professionally, Siddhanth serves as the Research Lead and Co-Founder at Xneuronz, and has interned at MuSigma, where he gained valuable experience in data analysis. He has also held various leadership roles, including President of Panacea, the Computer Communications Society, and General Secretary of the Artificial Intelligence Department Community. His active participation in academic events includes contributions to the organizing committees for the International Conference on Computation of Artificial Intelligence Machine Learning (ICCAIML) and the International Conference on Innovations in Computational Intelligence and Computer Vision (ICICV).

In addition to his academic and professional pursuits, Siddhanth holds certifications in TensorFlow, AWS, and Oracle Database Foundations, reflecting his commitment to continuous learning. Focused on applying AI to solve real-world challenges, he is passionate about advancing the field of computer science through both innovative research and practical applications.

• • •