JAM CM Journal of Applied Mathematics and Computational Mechanics 2026, 25(3), 47-55
www.amcm.pcz.pl p-ISSN 2299-9965
DOI: 10.17512/jamcm.2026.3.03 e-ISSN 2353-0588

FROM SYMMETRY TO SPARSITY: A DUAL FRAMEWORK OF LiSHT AND ReXTanh ACTIVATION FUNCTIONS FOR DEEP LEARNING

Emil Barciś*, Krzysztof Rojek

Department of Computer Science, Faculty of Computer Science and Artificial Intelligence
Czestochowa University of Technology
Częstochowa, Poland
emil.barcis@pcz.pl, krzysztof.rojek@icis.pcz.pl

Received: 16 March 2026; Accepted: 1 August 2026

Abstract. Activation functions are one of the fundamental concepts in the field of artificial neural networks used to optimize their performance and feature extraction. In this paper, we introduce a dual approach to those non-linear transformations, focusing on the balance between keeping information and enforcing sparsity. We start by analyzing the already established Linearly Scaled Hyperbolic Tangent (LiSHT) function. Our mathematical proofs demonstrate that it provides a smooth and differentiable approximation of the absolute value function. This makes LiSHT a differentiable L1-like regularizer that does not discard negative signals. Next, for models that need strict sparsity, we propose ReXTanh, a smooth-sparse hybrid function that provides C^1 continuity while remaining asymmetric. We tested both functions extensively on HPC clusters using mixed-precision distributed training, and the results highlight a clear split in use cases. While ReXTanh works best as a sparse operator for standard image datasets, yielding higher accuracy, LiSHT's symmetric shape handles complex and noisy data much better, showing that keeping negative magnitudes can outmatch forced sparsity in harder recognition problems.

MSC 2010: *68T05, 68T10, 65K05*
Keywords: *deep learning, activation functions, neural sparsity, LiSHT, ReXTanh*

1. Introduction

The accuracy and training time of a neural network are heavily dependent on the choice of an activation function. Over the years, researchers have moved away from saturating functions like Sigmoid or Tanh to unbounded, piecewise linear options such as ReLU, which quickly became the standard choice. The reason for this was its computational simplicity and setting all negative values to zero, which forced sparsity. Despite this, researchers are still trying to find out why exactly some non-linear transformations work better than others in different scenarios.

*Corresponding author

Strict sparsity is not always the best solution, because it sometimes leads to the "dying ReLU" problem, where neurons stop updating completely due to zeroing gradients. As a solution, many researchers have suggested smooth, non-monotonic functions that allow some negative gradients to pass through.

Beyond standard image recognition tasks, activation functions are used in applied computational mechanics. For example, Physics-Informed Neural Networks (PINNs) often rely on smooth functions, like tanh, to calculate spatial and temporal derivatives for partial differential equations (PDEs) [1]. Moreover, studying C^∞-continuous functions gives engineers more tools to use in models requiring smooth higher-order derivatives. On the other hand, automated visual defect detection in industrial settings requires very efficient convolutional networks for fast feature extraction [2]. In these systems, using sparse activations like ReLU helps keep classification accuracy high while lowering computation time. This leads to the conclusion that testing both smooth and sparse transformations is valuable in modern engineering applications.

In this paper, we look closely at the mathematical properties of the Linearly Scaled Hyperbolic Tangent (LiSHT) function introduced in 2019 by Roy et al. [3]. Specifically, we show that LiSHT belongs to a sequence of smooth approximations of the absolute value function. In other words, it serves as a continuously differentiable regularizer. On the other hand, some network architectures still depend on sparsity to maintain efficiency. To satisfy this need, we propose a new activation function called ReXTanh that serves as a C^1-differentiable ReLU equivalent.

Our main contributions are:

- Providing mathematical proof showing that the LiSHT-based sequence uniformly converges to the absolute value function.
- Introducing ReXTanh as a smooth-sparse hybrid together with its derivative and boundary analysis.
- Conducting distributed learning experiments on NVIDIA A100 GPUs shows that ReXTanh performs reliably on standard vision tasks, whereas LiSHT achieves superior results on highly complex, high-variance data.

The remainder of this paper is organized as follows. Section 2 reviews the related literature on modern activation functions. Section 3 outlines the theoretical framework and mathematical proofs for both LiSHT and ReXTanh. Section 4 describes the experimental methodology and distributed training configuration. The evaluation results are presented in Section 5, followed by a detailed discussion in Section 6. Finally, Section 7 concludes the paper.

2. Related work

Developing a new activation function usually involves efforts to balance simplicity of its formula with better gradient flow. On the one hand, ReLU [4] showed

that non-saturating functions speed up training much better than older Sigmoid units. On the other, it had problems with "dying neurons", when their gradients got zeroed. To solve the issue, researchers suggested functions like Leaky ReLU and Exponential Linear Units (ELU) that used small negative values instead of zeroing negative inputs. Later, Klambauer et al. brought in Self-Normalizing Neural Networks along with SELU [5], which normalizes network activations automatically using fixed-point dynamics.

In the late 2010s, the researchers focused mostly on smooth, non-monotonic activations. Functions like Swish [6] or the Gaussian Error Linear Unit (GELU) [7] showed that allowing a smooth dip into negative numbers can improve the network's ability to generalize. Soon after, Misra [8] improved self-regularized non-monotonicity even further with the Mish function, which uses a smooth, softplus-mediated tanh curve to preserve small negative gradients. A comprehensive taxonomy of these earlier advancements can be found in the survey by Dubey et al. [9]. More recently, this direction has been extended further by architectural variations such as Swish-T [10], which enhance the standard Swish design by integrating a tanh-based modification designed to improve the stability and flexibility of the activation. While these contemporary functions achieve state-of-the-art results by relying on smooth, asymmetric non-monotonicity, our framework offers a distinct dual alternative. Instead of enforcing a slight negative dip, we explore the two logical extremes of this design space: utilizing a fully symmetric formulation to preserve magnitude information via an already established function, and introducing a new counterpart to enforce strict neural sparsity while maintaining C^1 continuity at the origin.

In 2019, Roy et al. introduced LiSHT [3]. The paper mostly focused on how well it performs on popular datasets. But currently, the literature lacks a deep mathematical explanation of why LiSHT's symmetric shape works so well. In this paper, we fill that gap by connecting its mechanics to the L1 norm and proposing ReXTanh as its sparse counterpart.

3. Theoretical framework

In this section, we establish the mathematical foundations of the evaluated activation functions, examining their analytic properties, boundaries, and convergence behaviors.

3.1. Revisiting LiSHT: The smooth L1 proxy

The LiSHT activation function, originally introduced by Roy et al. [3], is defined as $f(x) = x\tanh(x)$. It maps $\mathbb{R} \to [0,\infty)$, is strictly even as a product of two odd functions, and belongs to the class C^∞ (meaning it is infinitely continuously differentiable).

Its first derivative is bounded, approximately $-1.2 < \mathrm{LiSHT}'(x) < 1.2$, with horizontal asymptotes at $y = 1$ (as $x \to \infty$) and $y = -1$ (as $x \to -\infty$). The second derivative is bounded by $-0.147 \lesssim \mathrm{LiSHT}''(x) \leq 2$.

3.1.1. Approximation of the absolute value

A key contribution of this theoretical section is proving that LiSHT can serve as a differentiable surrogate for the absolute value function $|x|$.

It is easy to observe that the sequence of functions $(x\tanh(nx))_{n\in\mathbb{N}}$ converges point-wise to $|x|$ for all $x \in \mathbb{R}$. From this fact, we can proceed to establishing the stronger property of uniform convergence.

Theorem 1 *The sequence of functions* $(x\tanh(nx))_{n\in\mathbb{N}}$ *converges uniformly to* $|x|$. □

PROOF We must show that

$$\forall \varepsilon > 0 \;\; \exists n_0 \in \mathbb{N} \;\; \forall n \geq n_0 \;\; \forall x \in \mathbb{R} \quad ||x| - x\tanh(nx)| < \varepsilon.$$

Assume $x \geq 0$ and $n \geq 1$. Then:

$$x - x\tanh(nx) = x(1 - \tanh(nx)) = \frac{nx(1 - \tanh(nx))}{n}.$$

Substituting $u = nx$, the function $u(1 - \tanh(u))$ reaches its maximal value of approximately 0.2785 at $u_0 = \frac{1}{2}\left(W\left(\frac{1}{e}\right) + 1\right) \approx 0.6392$, where W is the Lambert W function. Let us take $M = 0.28$. Then:

$$\frac{nx(1 - \tanh(nx))}{n} = \frac{u(1 - \tanh(u))}{n} < \frac{M}{n}.$$

To ensure the convergence condition is satisfied, we set $n_0 = \left\lfloor \frac{M}{\varepsilon} \right\rfloor + 1$. Symmetric bounds hold for $x < 0$, ensuring uniform convergence across $\mathbb{R}$. ■

3.2. Introducing ReXTanh: The sparse variant

While LiSHT's symmetry keeps information about negative signal strength, many networks need sparsity to work properly. We proposed ReXTanh to combine the smooth hyperbolic curve of LiSHT with the zero-cutoff behavior of ReLU.

We define ReXTanh as:

$$\mathrm{ReXTanh}(x) = \mathrm{ReLU}(x) \times \tanh(x) = \mathbb{1}_{x>0} \times (x\tanh(x)) \tag{1}$$

where $\mathbb{1}_{x>0}$ denotes the indicator function that outputs 1 if the condition $x > 0$ is satisfied, and 0 otherwise.

This function maps $\mathbb{R} \to [0, \infty)$. Standard ReLU is C^0, but ReXTanh is C^1 differentiable at the origin. We can check the limits at $x = 0$:

$$\lim_{h \to 0^+} \frac{h \tanh(h) - 0}{h} = \lim_{h \to 0^+} \tanh(h) = 0$$

$$\lim_{h \to 0^-} \frac{0 - 0}{h} = 0$$

Because both limits match, the derivative exists at $x = 0$ and equals 0.

The first derivative looks like this:

$$\text{ReXTanh}'(x) = \mathbb{1}_{x>0} \times \left[\tanh(x) + x(1 - \tanh^2(x))\right]$$

It is strictly bounded within $[0, 1.2)$, which prevents exploding gradients while training.

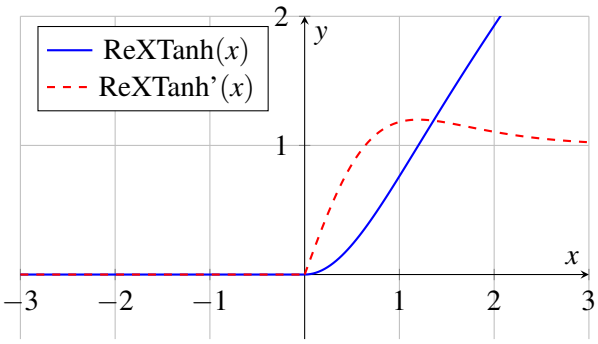

Fig. 1. Plot of ReXTanh and its first derivative. The function drops to zero for all negative inputs, ensuring strict sparsity

4. Methodology

We set up a large training pipeline to see how these functions compare against well established activations like ReLU, SELU, and Tanh.

4.1. Network architecture and datasets

We used the same Convolutional Neural Network (CNN) architecture for all tests. The network has two main convolutional blocks. Each block consists of two 3×3 Conv2D layers with 64 channels, batch normalization, the chosen activation function, 2×2 Max Pooling, and a 25 % Dropout. After those blocks, there is a fully connected layer with 16 neurons before the final classification head.

We ran tests on three datasets with different difficulty levels: MNIST [11], Fashion--MNIST [12], and EMNIST-Balanced [13].

4.2. Distributed training setup

The experiments were performed on the PLGrid supercomputing infrastructure using computing nodes equipped with NVIDIA A100-SXM4-40GB GPUs and AMD EPYC 7742 64-Core Processors, interconnected via a 200 Gb/s InfiniBand network. To handle the tests, we chose PyTorch's Distributed Data Parallel (DDP). The tests scaled up to 4 GPUs per node with global batch sizes from 64 all the way to 1024.

We chose the Adam optimizer along with a ReduceLROnPlateau scheduler (using a factor of 0.5 and patience of 5 epochs). Network weights used standard PyTorch initializations, like Kaiming uniform for convolutional layers. Every model trained for 100 epochs using Cross-Entropy loss. We also logged performance and speed under different precision settings: float32, mixed_float16, and mixed_bfloat16.

5. Results

In total, we gathered data from 135 different training setups, which varied precision levels, batch sizes, datasets, and GPU counts.

5.1. General performance averages

The test results show stable performance across all configurations. Out of 135 cases, ReXTanh reached the absolute highest accuracy 50 times. This confirms it works well for standard recognition problems. Table 1 presents the average classification accuracies across the three datasets.

Table 1. Average classification accuracy [%] across all tested training setups

Dataset	ReXTanh	LiSHT	ReLU	SELU	Tanh
MNIST	**99.52**	99.51	99.50	99.38	99.20
Fashion-MNIST	93.04	92.88	**93.12**	92.78	92.17
EMNIST-Balanced	88.89	**89.01**	88.71	88.67	85.65

ReXTanh achieves the best average accuracy on MNIST. Standard ReLU does slightly better on average for Fashion-MNIST (93.12 % vs. 93.04 %), but ReXTanh still generally performs better than LiSHT and SELU. On the other hand, the EMNIST--Balanced dataset, with 47 classes that look very similar to each other, shows that the fully symmetric LiSHT function has a slight advantage over the sparse activations.

5.2. Distributed scaling and training time

To show how these functions handle parallel environments, we chose a highly optimized 4-GPU setup with mixed_float16 precision and a global batch size of 128, which results in a local batch size of 32 per GPU.

As we can see in Table 2, with this specific parameter set, ReXTanh achieves the highest accuracy across all three datasets. It slightly outperforms both ReLU and LiSHT. The results also show that adding multiplication by ReLU to LiSHT to create ReXTanh causes a small drop in training speed. In this specific case, it takes about 4 % - 8 % longer to train when compared to standard ReLU, which, considering the accuracy increase, seems acceptable.

Table 2. Accuracy [%] and time [s] for a 4-GPU setup (Global Batch 128)

Dataset	**Activation**	**Accuracy [%]**	**Time [s]**
MNIST	**ReXTanh**	**99.58**	282.43
	LiSHT	99.52	271.46
	ReLU	99.50	**260.68**
	SELU	99.33	261.19
	Tanh	99.15	261.58
Fashion-MNIST	**ReXTanh**	**93.16**	280.24
	ReLU	93.01	260.00
	LiSHT	92.91	273.49
	SELU	92.66	260.44
	Tanh	91.57	**259.48**
EMNIST-Balanced	**ReXTanh**	**89.01**	477.67
	LiSHT	89.00	467.02
	ReLU	88.58	447.91
	SELU	88.40	449.27
	Tanh	85.13	**445.06**

6. Discussion

The tests show that choosing the right non-linear transformation for a given dataset and parameters can improve network performance.

The evaluation of our framework under properly tuned environments indicates that combining the smoothness of hyperbolic curves with ReLU's sparsity is highly effective. The results in Table 2 prove that by cutting out negative noise right away, ReXTanh acts as an effective filter, slightly outperforming standard functions on MNIST and Fashion-MNIST. Moreover, on the more challenging EMNIST-Balanced dataset, both ReXTanh and LiSHT scored higher than ReLU and SELU. Considering SELU was built to keep variance stable in tough network topologies [5], we can see that ReXTanh can be an accurate replacement for standard tasks, and our hyperbolic approach can compete with advanced self-normalizing functions.

When shifting focus to complex data domains, the limitations of strict sparsity become apparent. Although ReXTanh might give the highest peak accuracy under perfect conditions, Table 1 highlights LiSHT's main advantage: its ability to handle high-variance data very well without heavy tuning. On EMNIST-Balanced, trying to force sparsity often lowers performance. Since LiSHT acts as a smooth L1 regularizer (as proven in Theorem 1), it maps negative values to their positive counterparts.

This keeps important structural details intact, and this is why LiSHT maintained the highest average accuracy on EMNIST across all 135 tests. It is the safer choice when you cannot spend much time on hyperparameter tuning.

Regarding the magnitude of the performance gains, it is worth addressing both their practical significance and potential statistical relevance. Although the absolute improvements in classification accuracy are relatively modest (typically ranging between 0.1 % and 0.43 % depending on the dataset), such increments are practically meaningful in high-throughput automated processing pipelines (e.g., industrial defect inspection or high-volume document recognition), where fraction-of-a-percent reductions in error rates correspond to thousands of fewer misclassified instances at scale. Furthermore, with respect to statistical relevance, the reported gains are remarkably consistent across a comprehensive suite of 135 distinct experimental configurations – spanning varied numerical precisions, batch sizes, and multi-GPU distributed nodes. This empirical consistency indicates that the performance edge of ReXTanh and LiSHT stems from a fundamental structural property of their activation dynamics rather than random seed fluctuations or localized hyperparameter alignment.

Finally, to analyze the computing costs, we must point out that modern Tensor Cores process simple piecewise functions very quickly. Because ReXTanh combines $\mathrm{ReLU}(x)$ and $\tanh(x)$, it requires more memory access steps. In our 4-GPU runs, this resulted in a slightly longer training time than native ReLU. The choice for developers is straightforward: use ReXTanh for better classification accuracy if you have the computing time, or stick with LiSHT for stable training on more difficult datasets.

7. Conclusions

In this paper, we described a dual approach to neural network activation functions. We examined the properties of LiSHT, including its symmetry. By formally proving LiSHT-based sequence convergence to the absolute value function, we showed the function itself acts as a smooth L1 regularizer. To fulfill the need for sparsity, we proposed ReXTanh, which zeroes out negative inputs like ReLU but also maintains C^1 continuity.

Our multi-GPU experiments point to distinct use cases. LiSHT preserves signal magnitudes, helping it maintain consistent performance on complex problems. ReXTanh relies on sparsity instead. This makes it a reliable choice for simpler tasks. These results show that relaxing strict sparsity to allow symmetric smoothing provides a clear advantage on difficult visual datasets.

Acknowledgments

We gratefully acknowledge the Polish high-performance computing infrastructure PLGrid (HPC Center: ACK Cyfronet AGH) for providing computer facilities and support within computational grant no. PLG/2026/019154.

References

[1] Raissi, M., Perdikaris, P., & Karniadakis, G.E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, *378*, 686-707.

[2] Weimer, D., Scholz-Reiter, B., & Shpitalni, M. (2016). Design of deep convolutional neural network architectures for automated feature extraction in industrial inspection. *CIRP Annals*, *65*(1), 417-420.

[3] Roy, S.K., Manna, S., Dubey, S.R., & Chaudhuri, B.B. (2019). LiSHT: Non-parametric linearly scaled hyperbolic tangent activation function for neural networks. *arXiv preprint arXiv:1901.05894*.

[4] Nair, V., & Hinton, G.E. (2010). Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 807-814.

[5] Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 971-980.

[6] Ramachandran, P., Zoph, B., & Le, Q.V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[7] Hendrycks, D., & Gimpel, K. (2016). Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*.

[8] Misra, D. (2020). Mish: A self regularized non-monotonic activation function. *British Machine Vision Conference (BMVC)*.

[9] Dubey, S.R., Singh, S.K., & Chaudhuri, B.B. (2022). Activation functions in deep learning: A comprehensive survey and benchmark. *Neurocomputing*, 503, 92-108.

[10] Seo, Y., Kim, J., & Park, U. (2024). Swish-T: Enhancing swish activation with tanh bias for improved neural network performance. *arXiv preprint arXiv:2407.01012*.

[11] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[12] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

[13] Cohen, G., Afshar, S., Tapson, J., & van Schaik, A. (2017). EMNIST: Extending MNIST to handwritten letters. *2017 International Joint Conference on Neural Networks (IJCNN)*, 2921-2926.