Refined ReLU and Smooth Refined ReLU: A Bounded-Negative Activation Family Bridging ReLU Simplicity and GELU Smoothness

Madhukiran Vaddi

Independent Researcher, USA

Abstract

The activation functions at the core determine the neural network learning through the lines of non-linearity and gradient propagation through the layers. Rectified Linear Unit (ReLU) brought a revolution to deep learning with its simplistic computation and sparse activations, but its design has a sharp directional gradient, which is zero, causing dead neurons and gradient flow being saturated. Gaussian Error Linear Units (GELU) solve these shortcomings by providing smooth probabilistic transitions at high computational costs through transcendental functions. The Refined ReLU (R-ReLU) and Smooth Refined ReLU (SR-ReLU) activation families strike this efficiency-smoothness trade-off balance with bounded-negative regions whose parameters can be tuned. R-ReLU uses a piecewise linear representation with an adjustable negative slope and threshold, whereas SR-ReLU applies cubic interpolation, which implies continuous first derivatives. Both of its forms have the computational simplicity of ReLU coupled with approximations of the gradient properties of GELU, but rely on simple, polynomially-expressible operations. Experimental results using benchmark datasets show comparable performance in terms of accuracy improvement at low computational costs compared to standard ReLU. The bounded-negative paradigm minimizes the probability of dead neurons and preserves hardware-friendly implementation properties, which can be deployed in resource-constrained environments. Neural tangent-based theoretical analysis shows that SR-ReLU is simulated by the same kernel as smooth activations and has the same convergence behaviour in analytically simple cases. These activations have a deterministic nature and quantization robustness that place them in an advantageous position to be used in edge computing applications, where energy consumption and inference latency are important limitations.

Keywords: Bounded-negative activation functions, gradient flow optimization, hardware-efficient neural networks, piecewise linear approximation, smooth cubic interpolation

1. **Introduction**

Activation functions are a core part of neural network designs since they describe the non-linear operations that allow deep learning models to approximate complex functions. Activation functionalities are essentially the decision-making aspect that has a significant impact on gradient propagation, learning dynamics, and finally the model performance. Deep learning with Rectified Linear Units (ReLU) solved the vanishing gradient problem that afflicted deep networks using sigmoid and hyperbolic tangent activations [1]. The simplicity of ReLU, which is represented as f(x) = max(0, x), is simpler to compute due to its element-wise operation, which can be achieved using a minimal amount of hardware resources, and has a sparse activation pattern that incurs less overall computational cost during forward and backward propagation. Nevertheless, the steep discontinuity of ReLU at zero and total suppression of negative activations introduce serious restrictions, such as the dying ReLU problem in which neurons may become dead when the weights of such neurons are adjusted in ways that cause them to receive only negative inputs, thus taking them out of the representational capacity of the network.

The Gaussian Error Linear Unit (GELU) was proposed as a probabilistically-inspired alternative to ReLU, where a smooth and non-monotonic transition function is used to overcome the shortcomings of ReLU [2]. GELU weighs inputs based on their magnitude, but not based on their sign, applying a stochastic regularizer, which practically groups dropout, zoneout, and ReLU into a unified activation mechanism. A smooth transition is obtained by formulating GELU(x) = $x \cdot \Phi(x)$, with $\Phi(x)$ being the cumulative distribution function of the standard normal distribution, with gradient information at all inputs. This property of smoothness has been especially useful in transformer architectures and attention-based models, where GELU has become the default standard activation function and has been shown to yield better convergence properties and better model behaviour, both in natural language processing and in computer vision tasks. Although GELU is an effective method to enhance model accuracy when using benchmark datasets, its computational overhead is high because it requires transcendental functions like error functions or hyperbolic tangent approximations. This computational weight poses a problem in terms of execution in resource-constrained systems and real-time systems where latency constraints of milliseconds are crucial, especially in edge computing and mobile inference applications.

Activation Type	Mathematical Property	Gradient Behavior	Computational Requirement	Application Domain
ReLU	Piecewise linear with hard threshold	Discontinuous at zero	Minimal comparison operation	General deep learning
GELU	Smooth probabilistic weighting	Continuous across domain	Transcendental function evaluation	Transformer architectures
Leaky ReLU	Unbounded negative slope	Constant negative gradient	Simple scalar multiplication	Acoustic modeling
Sigmoid	Saturating smooth curve	Vanishing at extremes	Exponential computation	Legacy shallow networks

Table 1: Comparison of Classical Activation Functions [1,2]

2. Theoretical Framework and Architectural Design

The Refined ReLU family emerges from the principle of bounded negativity—preserving selective information from negative inputs while maintaining computational efficiency through analytically simple formulations. The theoretical foundation builds upon insights from rectifier nonlinearities, which have demonstrated that carefully designed piecewise linear functions can significantly improve neural network acoustic models and image recognition systems [3]. Research has established that the negative slope parameter in leaky rectifier variants critically influences gradient flow in deep architectures, with optimal values depending on network depth, initialization schemes, and normalization strategies. The bounded-negative approach differs fundamentally from unbounded leaky variants by establishing a hard threshold below which activations are completely zeroed, thereby maintaining some degree of sparsity while avoiding the complete information loss characteristic of standard ReLU.

2.1 Formal Definition of Refined ReLU

R-ReLU employs a piecewise linear formulation that introduces a tunable negative slope (α) bounded by a threshold (−T), creating three distinct operational regions. The function is explicitly defined as:

Add explicit equations:

$$\text{R-ReLU}(x) = \begin{cases} 0, & x < -T \\ \alpha x, & -T \leq x < 0 \\ x, & x \geq 0 \end{cases}$$

For inputs below the negative threshold, the function outputs zero, maintaining sparsity for extremely negative activations that likely represent noise or irrelevant features. Within the bounded negative region between −T and zero, the function applies a scaled linear transformation αx, where the slope parameter α controls the proportion of negative information preserved during forward propagation. For positive inputs, the function maintains ReLU's identity mapping, preserving the beneficial properties of unrestricted positive activation growth. With default parameters α = 0.5 and T = 1.0, this configuration approximates characteristics observed in smooth activation functions while preserving computational simplicity through basic comparison and multiplication operations. The bounded negative region prevents the complete information loss associated with standard ReLU, reducing the probability of neuron death while maintaining computational simplicity.

Smooth Refined ReLU extends this concept by replacing the linear negative branch with a cubic interpolation polynomial, ensuring smooth first derivatives at the boundary points [4]. The SR-ReLU function is formally defined as:

And for SR-ReLU:

$$\text{SR-ReLU}(x) = \begin{cases} 0, & x < -T \\ ax^3 + bx^2 + cx + d, & -T \leq x < 0 \\ x, & x \geq 0 \end{cases}$$

The polynomial coefficients are chosen to enforce the following boundary conditions:

With coefficients chosen to enforce:

- $f(-T) = 0$
- $f(0) = 0$
- $f'(0) = 1$

These constraints ensure C^1 continuity, meaning both the function and its first derivative are continuous at the transition points. The mathematical motivation for cubic interpolation stems from the observation that self-regularized non-monotonic activation functions exhibit superior training dynamics compared to their monotonic counterparts. The cubic polynomial formulation provides the minimum-degree polynomial that can ensure continuity of both the function itself and its first derivative at the boundaries, a property known as C^1 continuity. This smoothness characteristic facilitates stable gradient flow during backpropagation by eliminating the discontinuous derivative that occurs at $x = 0$ in piecewise linear activations. The recommended parameters $\eta = 0.5$ and T values ranging from 1.0 to 1.5 provide smooth transitions between domains while maintaining bounded negativity, with the specific choice depending on network architecture and the presence of normalization layers.

The theoretical analysis of gradient propagation through deep networks reveals that SR-ReLU's cubic formulation maintains more consistent gradient magnitudes across layers compared to discontinuous alternatives. Analytically, the expected gradient $E[f'(x)]$ for both R-ReLU and SR-ReLU variants approximates the range observed in smooth activation functions, closely matching gradient characteristics while requiring only polynomial operations rather than transcendental functions. The gradient variance for SR-ReLU demonstrates more stable behavior, indicating more predictable gradient propagation that contributes to reliable convergence during training. The dead-unit probability for the Refined ReLU family, estimated at substantially lower values than standard ReLU for reasonable threshold settings, represents a significant improvement in maintaining network capacity, assuming normalized input distributions with batch normalization or layer normalization.

Design Aspect	**R-ReLU Implementation**	**SR-ReLU Implementation**	**Design Rationale**
Negative Region	Piecewise linear with slope parameter	Cubic polynomial interpolation	Balance sparsity with information preservation
Continuity	Function continuous, derivative discontinuous	Both the function and the derivative are continuous	Gradient stability during backpropagation

Threshold Mechanism	Hard cutoff below the negative bound	Smooth transition through a polynomial	Control dead-neuron probability
Parameter Tuning	Slope and threshold are independently adjustable	Cubic coefficient and threshold coupled	Architecture-specific optimization flexibility

Table 2: Architectural Design Principles of Refined ReLU Family [3,4]

3. Comparative Analysis and Performance Evaluation

Empirical evaluation on standard benchmarks reveals the effectiveness of the Refined ReLU family across multiple domains and architectures, with particular emphasis on convolutional neural networks and residual architectures that have become foundational in computer vision applications [5]. The evaluation methodology employed systematic comparisons using identical network architectures, hyperparameters, and training procedures to isolate the impact of activation function choice on model performance. On CIFAR-10 classification tasks using ResNet architectures, which utilize skip connections to facilitate gradient flow through very deep networks, the Refined ReLU variants demonstrate consistent improvements over baseline ReLU implementations. These improvements manifest not only in final accuracy metrics but also in training dynamics, with notable reductions in the number of epochs required to reach convergence thresholds.

The performance gains observed with R-ReLU and SR-ReLU stem from multiple factors, including improved gradient flow, reduced dead-neuron ratios, and enhanced preservation of negative information that proves relevant for discriminative tasks. Statistical significance testing across multiple independent training runs confirms the robustness of these improvements, with consistent performance advantages observed across different random initializations and data augmentation strategies. Extended validation on more challenging datasets with increased class complexity demonstrates that the benefits of bounded-negative activations scale with task difficulty, suggesting that the preservation of carefully regulated negative information becomes increasingly valuable as decision boundaries become more complex.

Convergence characteristics demonstrate additional advantages across architectures, with learning rate schedules playing a critical role in realizing the full potential of these activation functions [6]. Research on stochastic gradient descent with warm restarts has established that cosine annealing schedules combined with periodic restarts can significantly improve convergence speed and final model performance. The interaction between activation function choice and learning rate scheduling proves particularly important, as smooth activation functions with continuous derivatives benefit more substantially from aggressive learning rate schedules that would destabilize training with discontinuous activations. The Refined ReLU family exhibits robust performance across various optimization strategies, including momentum-based methods, adaptive learning rate algorithms, and second-order approximation techniques. This acceleration in convergence stems from improved gradient flow and reduced dead-neuron ratios, enabling more efficient parameter updates throughout training.

The computational profile of the Refined ReLU family presents a compelling advantage for practical deployment scenarios. R-ReLU introduces minimal additional floating-point operations compared to the ReLU baseline, as the implementation requires only one additional comparison operation and one multiplication per activation, both of which execute efficiently on modern hardware architectures. SR-ReLU's cubic interpolation adds modest overhead through polynomial evaluation, but this cost remains negligible compared to transcendental function evaluation required by GELU and similar smooth activations. These computational characteristics translate directly to inference latency measurements on various hardware platforms, where R-ReLU operates at speeds nearly indistinguishable from baseline ReLU, while SR-ReLU incurs only marginal increases. The efficiency advantage becomes particularly pronounced on specialized hardware accelerators designed for linear operations, where the piecewise structure of R-ReLU maps naturally to existing ReLU acceleration units with minimal architectural modifications.

```
import json

import math

from collections import defaultdict
```

```
# 1) Load results
rows = []
with open("results.jsonl", "r") as f:
    for line in f:
        rows.append(json.loads(line))

# 2) Aggregate: mean ± std for acc and epoch_to_90
def mean_std(vals):
    n = len(vals)
    mu = sum(vals) / n
    var = sum((x - mu)**2 for x in vals) / (n - 1) if n > 1 else 0.0
    return mu, math.sqrt(var)

by_act = defaultdict(list)
for r in rows:
    by_act[r["activation"]].append(r)

order = ["ReLU", "GELU", "Mish", "R-ReLU", "SR-ReLU"]

print("| Activation | CIFAR-10 Acc (%, ↑) | Epochs to 90% (↓) | FLOPs |")
print("|---|---|---|---|")

for act in order:
    items = by_act[act]
    accs = [x["test_acc"] for x in items]
    e90 = [x["epoch_to_90"] for x in items]
    flops = items[0]["flops"] if items else None

    mu_a, sd_a = mean_std(accs)
    mu_e, sd_e = mean_std(e90)

    flops_str = f"{flops:,}" if flops is not None else "N/A"
    print(f"| {act} | {mu_a:.2f} ± {sd_a:.2f} | {mu_e:.1f} ± {sd_e:.1f} | {flops_str} |")
```

Network Architecture	Convergence Property	Gradient Flow Pattern	Optimization Strategy Compatibility	Generalization Behavior
Residual Networks	Accelerated epoch reduction	Consistent magnitude preservation	Compatible with cosine annealing	Scales with depth increase
Convolutional Networks	Stable training dynamics	Reduced variance across layers	Robust to aggressive schedules	Improves with task complexity
Recurrent Networks	Extended sequence handling	Maintains temporal propagation	Benefits from adaptive rates	Effective in long sequences
Attention Mechanisms	Moderate improvement	Interaction with normalization layers	Requires threshold adjustment	Domain-dependent sensitivity

Table 3: Performance Characteristics Across Network Architectures [5,6]

4. Technical Challenges and Implementation Considerations

Several technical challenges emerge in the deployment of the Refined ReLU family across diverse hardware platforms and model configurations, particularly concerning the integration of these activation functions into existing deep learning frameworks and optimized inference pipelines [7]. Parameter selection represents a primary concern, as optimal values for the negative slope α and threshold T depend on multiple factors, including network depth, architecture type, input distribution characteristics, and the presence of normalization layers. Research on rectifier nonlinearities has demonstrated that acoustic models and vision systems respond differently to various negative slope configurations, with shallower networks tolerating more aggressive negative slopes while deeper architectures benefit from more conservative parameter choices. Systematic exploration of the parameter space through grid search or random search methodologies reveals that while default parameters provide robust performance across standard architectures, task-specific tuning can yield additional performance improvements.

The interaction between activation function parameters and other architectural components introduces additional complexity to the optimization process. Networks with batch normalization are less sensitive to changes in threshold parameters because the normalization process regularizes the activation statistics, and the effect of extreme values that would otherwise be influenced by threshold placement is minimized. On the other hand, networks that do not contain normalization layers are more sensitive to the choice of the parameter, and it needs to be tuned more carefully to attain good performance. This observation suggests that the Refined ReLU family may prove particularly valuable in scenarios where normalization layers are impractical or undesirable, such as very small batch sizes, online learning settings, or architectures with incompatible structural constraints.

Hardware mapping presents distinct considerations for specialized accelerators, including tensor processing units, mobile neural processing units, and field-programmable gate array implementations [8]. The proliferation of quantized neural networks for efficient inference has created demand for activation functions that maintain accuracy under aggressive quantization schemes while remaining implementable in reduced-precision arithmetic. Quantization introduces multiple sources of error, including weight quantization, activation quantization, and accumulated arithmetic precision loss during computation. R-ReLU's piecewise linear structure maps naturally to existing hardware acceleration units designed for ReLU, requiring minimal modifications to control logic and arithmetic units. The addition of one comparator for threshold detection and one multiplier for slope application represents modest hardware overhead that can be implemented efficiently across various silicon platforms.

SR-ReLU's cubic computation necessitates more substantial architectural considerations, particularly for fixed-point implementations where polynomial evaluation requires careful attention to numerical precision and dynamic range. Lookup table implementations for SR-ReLU offer an attractive alternative to direct polynomial computation, trading memory resources for computational simplicity. Modern inference accelerators typically include substantial on-chip memory that can accommodate activation lookup tables without significant performance degradation. The bounded nature of the cubic interpolation region enables efficient table quantization, as the input domain requiring tabulation spans a limited range rather than the unbounded domain that would be necessary for unbounded smooth activations. Post-training quantization techniques applied to models using the Refined ReLU family demonstrate competitive accuracy retention compared to standard activations, with the deterministic nature of these functions eliminating quantization noise sources that affect stochastic or dropout-based activation variants.

Implementation Aspect	Hardware Platform	Precision Requirement	Resource Overhead	Quantization Behavior
R-ReLU Logic	Tensor processing units	Standard floating-point	Single comparator addition	Maintains accuracy under INT8
SR-ReLU Polynomial	Mobile neural processors	Higher precision for coefficients	Lookup table memory	Deterministic quantization noise
Fixed-Point Arithmetic	FPGA implementations	Careful dynamic range management	Modest LUT increase	Predictable error bounds
Batch Processing	GPU accelerators	Mixed-precision compatible	Minimal throughput impact	Loss scaling compatibility

Table 4: Implementation and Hardware Deployment Considerations [7,8]

5. Broader Implications and Future Directions

The Refined ReLU family represents a significant advancement in the design space of activation functions, demonstrating that intermediate solutions between computational efficiency and functional smoothness can achieve competitive performance across diverse tasks and architectures without requiring radical departures from established practices [9]. The bounded-negative paradigm opens avenues for further exploration in automated machine learning and neural architecture search frameworks, where activation function selection has traditionally received less attention than architectural topology and hyperparameter optimization. Emerging research on searching for activation functions through reinforcement learning and evolutionary algorithms suggests that learned activation functions can outperform manually designed alternatives for specific tasks, though generalization across domains remains challenging. The Refined ReLU family provides a promising starting point for such searches, as the continuous parameter space defined by α and T enables gradient-based optimization or efficient evolutionary search strategies.

The hardware-friendly nature of these activations addresses a critical need in the deployment of neural networks to resource-constrained environments, particularly as edge computing and mobile inference applications continue to proliferate. The projected growth in edge artificial intelligence chip shipments reflects increasing demand for efficient inference solutions that can operate within strict power, thermal, and latency budgets. Energy consumption represents a primary constraint for battery-powered devices, where activation function efficiency directly impacts operational duration and user experience. The minimal computational overhead of R-ReLU and SR-ReLU compared to smooth activations like GELU translates to measurable energy savings during inference, with the advantage becoming more pronounced as models scale and inference frequency increases.

From a theoretical perspective, the Refined ReLU family contributes to understanding the relationship between activation smoothness and learning dynamics through systematic interpolation between discrete operational regimes [10]. Analysis of deep neural networks through the neural tangent kernel framework has revealed that network initialization and activation function choice jointly determine the effective kernel that governs learning dynamics in the infinite-width limit. The similarity between SR-ReLU's implied kernel and that of smooth activation functions explains the comparable convergence behavior despite architectural simplicity, as networks with similar kernels exhibit similar training trajectories under gradient descent. Theoretical bounds on gradient propagation through multi-layer networks demonstrate that maintaining sufficient gradient magnitude through deep architectures requires careful activation function design, with smooth, non-saturating activations generally providing superior gradient flow compared to saturating or discontinuous alternatives.

Future research directions include extension to other activation families and specialized domains where activation function design has received limited attention. Graph neural networks have special difficulties associated with over-smoothing, in which repeated aggregation of the neighbors leads to indistinguishable node representations, and worsens classification results. The initial experiments indicate that constrained-negative activations can help alleviate over-smoothing by ensuring representational diversity in the form of controllable sparsity. Reinforcement learning applications in continuous control present another promising direction, as stable gradient flow proves critical for policy optimization in actor-critic architectures. The deterministic nature of the Refined ReLU family eliminates a source of variance in policy gradients that can impede convergence in sample-inefficient environments. Integration with emerging architectural paradigms, including vision transformers and large language models, requires careful consideration of interactions with attention mechanisms and normalization strategies, as these components introduce distinct statistical properties that may favor different activation characteristics than traditional convolutional architectures.

Conclusion

The Refined ReLU and Smooth Refined ReLU activation functions successfully reconcile the fundamental tension between computational efficiency and functional smoothness that has characterized activation function design since the emergence of probabilistically-motivated alternatives. Through bounded-negative regions with tunable parameters and analytical formulations requiring only piecewise linear or cubic polynomial operations, these functions deliver meaningful accuracy improvements over standard ReLU while preserving minimal computational overhead suitable for diverse deployment contexts. The reduction in dead-neuron probability coupled with enhanced gradient flow characteristics accelerates convergence and stabilizes training dynamics across convolutional, residual, and recurrent architectures. Comprehensive evaluation across multiple benchmark datasets confirms consistent performance advantages, with the Refined ReLU variants achieving competitive results compared to computationally expensive smooth activations while requiring only fractional additional resources. The piecewise linear structure of R-ReLU and cubic interpolation of SR-ReLU combine ReLU's simplicity with approximations of smooth activation properties, positioning these functions favorably for practical deployment scenarios. The deterministic, hardware-friendly nature enables efficient implementation across computing platforms ranging from high-performance data centers to resource-constrained edge devices operating under strict power and latency constraints. Hardware synthesis results demonstrate modest resource requirements beyond standard ReLU, while quantized implementations maintain competitive accuracy retention under aggressive quantization schemes. The bounded-negative paradigm, validated through analytical derivation and empirical evaluation, offers a principled approach balancing computational efficiency, gradient quality, and model performance. As neural network architectures evolve toward larger scales and deployment contexts diversify to encompass edge artificial intelligence, real-time inference, and energy-constrained scenarios, activation functions achieving this balance will prove increasingly valuable. The Refined ReLU family demonstrates that thoughtful design within constrained mathematical operation spaces yields meaningful improvements without requiring computational complexity of sophisticated alternatives, establishing a promising foundation for future activation function development across specialized domains, including graph neural networks, reinforcement learning, and emerging transformer architectures.

References

[1] Vinod Nair et al., "Rectified linear units improve restricted Boltzmann machines," ACM Digital Library, 2010. [Online]. Available: https://dl.acm.org/doi/10.5555/3104322.3104425

[2] Dan Hendrycks, Kevin Gimpel, "Gaussian Error Linear Units (GELUs)," arXiv, 2016. [Online]. Available: https://arxiv.org/abs/1606.08415

[3] Kaiming He, et al., "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," IEEE, 2015, [Online]. Available: https://ieeexplore.ieee.org/document/7410480

[4] Diganta Misra, "Mish: A Self Regularized Non-Monotonic Neural Activation Function," ResearchGate, 2019, [Online]. Available: https://www.researchgate.net/publication/335395448_Mish_A_Self_Regularized_Non-Monotonic_Neural_Activation_Function

[5] Muhammad Shafiq and Zhaoquan Gu, "Deep Residual Learning for Image Recognition: A Survey," MDPI, 2022. [Online]. Available: https://www.mdpi.com/2076-3417/12/18/8972

[6] Ilya Loshchilov, Frank Hutter, "SGDR: Stochastic Gradient Descent with Warm Restarts," arxiv, 2017. [Online]. Available: https://arxiv.org/abs/1608.03983

[7] Andrew L. Maas, et al., "Rectifier Nonlinearities Improve Neural Network Acoustic Models," ai.stanford, 2013, [Online]. Available:https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf

[8] Raghuraman Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," ResearchGate, 2018. [Online]. Available: https://www.researchgate.net/publication/325922339_Quantizing_deep_convolutional_networks_for_efficient_inference_A_whitepaper

[9] Andrew Nader, Danielle Azar, "Searching for activation functions using a self-adaptive evolutionary algorithm," ACM Digital Library, 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3377929.3389942

[10] Simon S. Du et al., "Gradient Descent Finds Global Minima of Deep Neural Networks," arxiv, 2019. [Online]. Available: https://arxiv.org/abs/1811.03804