Journal of Data Science and Modeling, Vol. 3, No. 2, 137-168, June 2025

Research Manuscript

A parametric activation function based on Wendland RBF

Majid Darehmiraki*

Department of Mathematics and Statistics,

Behbahan Khatam Alanbia University of Technology, Khouzestan, Iran

Recieved: 10/11/2025 Accepted: 31/08/2026

Abstract: This paper introduces a novel parametric activation function based on Wendland radial basis functions (RBFs) for deep neural networks. Wendland RBFs, known for their compact support, smoothness, and positive definiteness in approximation theory, are adapted to address limitations of traditional activation functions like ReLU, sigmoid, and tanh. The proposed enhanced Wendland activation combines a standard Wendland component with linear and exponential terms, offering tunable locality, improved gradient propagation, and enhanced stability during training. Theoretical analysis rigorously examines derivative behavior, smoothness, gradient flow, and saturation properties, demonstrating advantages over ReLU, GELU, and Swish, including strictly positive gradients, C^2-continuity, near-unity gradient decay, and well-conditioned Jacobians. Empirical experiments on synthetic tasks and benchmark datasets confirm competitive performance, with comprehensive diagnostic analyses validating the theoretical claims. Results show that the Wendland-based activation achieves superior accuracy in certain scenarios, particularly in regression tasks, while maintaining computational efficiency. The study bridges classical RBF theory with modern deep learning, suggesting that Wendland activations can mitigate overfitting and improve generalization through localized, smooth transformations. **Keywords:** Activation function, Deep learning, Wendland RBF, Compact support, Parametric.
Mathematics Subject Classification (2010): 68T07.

*Corresponding Author: darehmiraki@bkatu.ac.ir

1. Introduction

Activation functions are central to deep neural networks because they introduce nonlinearity and determine how information and gradients propagate through the model. ReLU and its variants are computationally efficient and alleviate the classical vanishing-gradient problem, but ReLU can suffer from inactive neurons, whereas sigmoid and tanh may saturate and produce weak gradients. These limitations have motivated a large family of smooth, adaptive, and trainable activation functions Glorot and *et al.* (2011); Nair and Hinton (2010); Bengio and *et al.* (1994); Gulcehre and *et al.* (2016); Xu and *et al.* (2016, 2015); Apicella and *et al.* (2021); Dubey and *et al.* (2022).

Recent work includes ELU and related exponential activations Clevert and *et al.* (2015), GELU Hendrycks and Gimpel (2016), trainable PReLU-type functions He and *et al.* (2015), Swish and its variants Ramachandran and *et al.* (2017); Javid and *et al.* (2022), self-learnable activations Goyal and *et al.* (2019), Pad'e activation units Molina and *et al.* (2019), fractional ReLU Job and *et al.* (2022), SinLU Paul and *et al.* (2022), algebraic activations Naresh and *et al.* (2017), and other adaptive constructions Wang and *et al.* (2022); Chen and Pan (2023); Strack and *et al.* (2024); Zhang and Ding (2025). These studies indicate that activation shape, smoothness, and parameter adaptivity can substantially affect optimization and generalization.

The present work investigates compactly supported Wendland radial basis functions (RBFs) as activation functions. Wendland kernels combine locality, smoothness, and positive definiteness Wendland (2004). We use the C^2 Wendland function and introduce a learnable support parameter together with a linear term and a decaying exponential tail. The resulting activation is designed to retain the locality and smoothness of the kernel while improving gradient propagation outside the compact support.

The main contributions are:

1. a parametric enhanced Wendland activation with four trainable parameters $(\alpha, \lambda, \beta, \epsilon)$;

2. explicit derivatives with respect to inputs and activation parameters, including behavior at the support boundary;

3. theoretical analysis of smoothness, gradient propagation, saturation, and training stability;

4. empirical comparison with widely used activations on synthetic regression and classification tasks and on MNIST and Fashion-MNIST; and

5. quantitative diagnostics of dead units, gradient variability, variance decay, and effective-rank degradation in deeper networks.

1.1 Motivation

The effect of activation smoothness on deep-network training has been studied through signal-propagation and edge-of-chaos analyses. Smooth activations can preserve correlations and improve gradient propagation in deep networks Hayou and *et al.* (2019). These observations motivate an activation that is simultaneously smooth, local, and adaptable. Wendland RBFs provide compact support and prescribed differentiability, while the proposed trainable parameters allow the effective response to adapt during learning.

2. Overfitting and active function

In deep learning, the choice of activation functions can influence whether a model overfits, which occurs when the model learns noise or overly complex patterns from the training data, leading to poor generalization on unseen data. Activation functions introduce nonlinearity, allowing neural networks to learn complex relationships, but some functions can exacerbate overfitting if not properly regulated. For instance, activation functions like ReLU (Rectified Linear Unit) and its variants (Leaky ReLU, Parametric ReLU) are widely used because they help mitigate the vanishing gradient problem and enable faster training, but they can also lead to overly confident predictions if the model becomes too specialized in the training data. This is particularly true when the network is very deep or has excessive capacity, as ReLU's unbounded positive output can amplify large activations, making the model prone to memorizing training examples rather than generalizing. On the other hand, activation functions with inherent regularization properties, like sigmoid or tanh, which squash outputs into a bounded range, can sometimes help curb extreme activations, but they come with their own challenges, such as the vanishing gradient problem, which may hinder learning in deep networks. Additionally, some modern activation functions, like Swish or GELU, balance nonlinearity and smoothness, potentially offering better generalization by providing more nuanced gradients during backpropagation. The relationship between activation functions and overfitting is also tied to other factors, such as the network architecture, the amount of training data, and the use of regularization techniques like dropout or weight decay. For example, combining ReLU with dropout can help prevent overfitting by randomly deactivating neurons during training, thereby reducing reliance on specific activation patterns. Ultimately, while the activation function itself is not the sole determinant of overfitting, its

interaction with other model components plays a significant role in either encouraging or mitigating overfitting, making it an important consideration in designing robust deep learning models.

3. Activation function

Activation functions are used in neural networks to calculate the weighted total of inputs and biases, which determines whether a neuron is activated. It processes the provided data using gradient techniques, often gradient descent, and then generates an output for the neural network that encompasses the parameters within the data. These active functions are often designated as a transfer function in some publications.

Among the reasons why these active functions are necessary is the fact that they are able to transform linear input signals and models into nonlinear output signals. This facilitates the learning of higher-order polynomials that are above one degree for deeper networks. If the nonlinear activation functions are not differentiable, then they will not be able to operate properly during the backpropagation process of the deep neural networks. Nonlinear activation functions have a particular property.

The activation functions, also known as active functions, are very important in neural networks because they are responsible for learning the abstract properties via the use of nonlinear transformations. Listed below are some of the characteristics that are shared by all active functions: It is recommended that the following be done: a) it should incorporate a nonlinear curvature into the optimization landscape in order to enhance the convergence of the network during training; b) it should not significantly increase the computational complexity of the model; c) it should not impede the flow of gradients while the network is being trained; and d) it should keep the distribution of data in order to facilitate the efficient training of the network. Over the last several years, a number of active functions have been investigated for deep learning in order to accomplish the qualities that were indicated before.

With regard to the main classification, the potential of altering the form of the activation function throughout the training phase is the basis for the classification. Therefore, it is possible to draw two primary categories:

- Fixed-shape activation functions: this group includes all of the activation functions that have a fixed form. For instance, all of the traditional activation functions that are used in neural network literature, such as sigmoid, tanh, and ReLU, are included in this category. However, given that the introduction of corrected functions (as ReLU) may be seen as a turning point

in the literature, it has contributed to a major improvement in the performance of neural networks and has increased the attention of the scientific community.

- This class includes all activation functions learning their form during training. The purpose of these functions is to find a suitable form utilizing training data knowledge. We demonstrate that trainable activation functions may be reduced to conventional feed-forward neural subnetworks, which are integrated into the main neural network by adding layers with fixed activation functions. A neural network with trainable activation functions can behave similarly to a deeper network with fixed activation functions. This can be achieved by adding constraints on network parameters, such as weights (fixed or shared) and layer arrangement (e.g., convolutional networks).

4. Wendland RBF

The Wendland functions are a family of compactly supported radial basis functions (RBFs) commonly used in scattered data interpolation, numerical analysis, and machine learning. They are particularly valued because they are positive definite, have finite support, and can be constructed to achieve a desired smoothness (differentiability) [27]. The functions are named after Holger Wendland, who introduced them in the 1990s.

For a given spatial dimension d and smoothness parameter k, the Wendland function is constructed as:

$$\phi_{d,k}(r) = (1-r)_+^{\ell+k} p(r),$$

where $\ell = \lfloor d/2 \rfloor + k + 1$, $[\cdot]_+$ denotes the positive part, and $p(r)$ is a polynomial ensuring positive definiteness.

4.0.1 Classical Wendland Functions

The following are standard members of the Wendland family in dimension $d = 3$:

1. C^0 Wendland Function (Linear):

   $$\phi_{3,0}(r) = (1-r)_+^2,$$

   which is continuous but not differentiable at $r = 1$.

2. C^2 Wendland Function (Cubic):

   $$\phi_{3,1}(r) = (1-r)_+^4 (4r+1),$$

   which is twice continuously differentiable.

3. C^4 Wendland Function (Quintic):

$$\phi_{3,2}(r) = (1-r)_+^6(35r^2 + 18r + 3)/3,$$

which is four times continuously differentiable.

4.0.2 Our Implementation: The Enhanced Wendland Activation

For the proposed activation function, we implement a specific member of the Wendland family with the following precise specification:

$$\boxed{\phi_{3,1}^{(\alpha)}(r) = (1-\alpha r)_+^4(4\alpha r + 1)}$$

This corresponds to the C^2 Wendland function (the cubic variant) with the following modifications:

- The support radius is controlled by the learnable parameter $\alpha > 0$, replacing the fixed support at $r = 1$.
- The radial distance is scaled as αr, yielding compact support on $[0, 1/\alpha]$.
- The polynomial term is $P(\alpha r) = 4\alpha r + 1$, matching the standard cubic Wendland form.

The enhanced activation function is then defined as:

$$\phi(r) = \underbrace{(1-\alpha r)_+^4(4\alpha r + 1)}_{\text{Wendland } \phi_{3,1}^{(\alpha)}(r)} + \underbrace{\lambda r}_{\text{Linear Term}} + \underbrace{\epsilon e^{-\beta r}}_{\text{Exponential Tail}}.$$

4.0.3 Mathematical Guarantees

The chosen variant $\phi_{3,1}^{(\alpha)}(r) = (1-\alpha r)_+^4(4\alpha r+1)$ provides the following guarantees:

1. **Compact Support:**

$$\text{supp}(\phi_{3,1}^{(\alpha)}) = [0, 1/\alpha],$$

ensuring localized behavior and computational efficiency.

2. **Smoothness:**

The function is C^2-continuous (twice continuously differentiable) for $r \in [0, 1/\alpha]$, with continuous derivatives up to order 2. This follows from the construction:

$$\begin{aligned}
\phi_{3,1}^{(\alpha)}(0) &= 1, \\
\left(\phi_{3,1}^{(\alpha)}\right)'(0) &= -4\alpha, \\
\left(\phi_{3,1}^{(\alpha)}\right)''(0) &= 4\alpha^2, \\
\phi_{3,1}^{(\alpha)}(1/\alpha) &= \left(\phi_{3,1}^{(\alpha)}\right)'(1/\alpha) = \left(\phi_{3,1}^{(\alpha)}\right)''(1/\alpha) = 0.
\end{aligned}$$

This smoothness property is crucial for stable gradient propagation in deep neural networks, as demonstrated in [10].

3. **Positive Definiteness:**

   As established by Wendland [27], $\phi_{3,1}(r) = (1-r)_+^4(4r+1)$ is positive definite in $\mathbb{R}^d$ for $d \leq 3$. With the scaling αr, the function $\phi_{3,1}^{(\alpha)}(r)$ remains positive definite for $d \leq 3$. This property ensures that the interpolation matrix formed by these kernels is non-singular, providing mathematical stability.

4. **Monotonicity:**

   The function is strictly decreasing on $[0, 1/\alpha]$, providing a well-behaved response surface.

4.0.4 Why This Specific Variant?

We selected the C^2 Wendland variant $\phi_{3,1}$ for the following reasons:

1. **Balance of Smoothness and Locality:**

   The C^2 continuity provides sufficient smoothness for gradient-based optimization while maintaining compact support, unlike higher-order variants (C^4), which can be overly smooth and potentially less responsive to local variations.

2. **Computational Efficiency:**

   The polynomial $4r + 1$ is computationally inexpensive to evaluate, making it suitable for large-scale deep learning applications.

3. **Theoretical Grounding:**

   The positive definiteness guarantee ensures stable behavior during training, particularly in scenarios where the activation function is used in kernel-based formulations.

4. **Compatibility with Deep Learning:**

   As shown by Hayou et al. [10], smooth activations (like the C^2 Wendland) outperform ReLU-like functions in deep networks due to their faster convergence rates and better gradient propagation.

4.0.5 Choice of the Wendland Variant

For completeness, we note that other variants could potentially be used:

Table 1: Comparison of Wendland variants suitable for activation functions

Variant	Formula	Smoothness	Use Case
$\phi_{3,0}$	$(1-r)_+^2$	C^0	Not recommended (non-differentiable)
$\phi_{3,1}$	$(1-r)_+^4(4r+1)$	C^2	**Our choice**
$\phi_{3,2}$	$(1-r)_+^6(35r^2+18r+3)/3$	C^4	Alternative for smoother gradients

The $\phi_{3,0}$ variant is unsuitable due to its lack of differentiability at $r = 1$, which would cause gradient discontinuities during backpropagation. While the C^4 variant $\phi_{3,2}$ offers additional smoothness, it comes with increased computational cost and may over-smooth features in some applications.

4.0.6 Ablation Study on Parameter k

To validate our choice of $k = 4$ (i.e., the C^2 variant), we conducted a brief ablation study:

Table 2: Performance comparison for different polynomial degrees k on MNIST

k	Variant	Accuracy (%)
2	C^0	98.91
4	C^2	**99.33**
6	C^4	99.21

The results show that the C^2 variant ($k = 4$) achieves the best performance, justifying our selection.

$$\phi_{3,1}^{(\alpha)}(0) = 1, \tag{4.1}$$

$$\left(\phi_{3,1}^{(\alpha)}\right)'(0) = -4\alpha, \tag{4.2}$$

$$\left(\phi_{3,1}^{(\alpha)}\right)''(0) = 4\alpha^2, \tag{4.3}$$

$$\phi_{3,1}^{(\alpha)}(1/\alpha) = \left(\phi_{3,1}^{(\alpha)}\right)'(1/\alpha) = \left(\phi_{3,1}^{(\alpha)}\right)''(1/\alpha) = 0. \tag{4.4}$$

For

$$d \leq 3, \quad \sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j \phi_{3,1}^{(\alpha)}(\|\mathbf{x}_i - \mathbf{x}_j\|) \geq 0,$$

with equality iff $c_i = 0$ for all i.

4.0.7 Choice of the Wendland Variant

For completeness, we note that other variants could potentially be used:

Table 3: Comparison of Wendland variants suitable for activation functions

Variant	Formula	Smoothness	Use Case
$\phi_{3,0}$	$(1-r)_+^2$	C^0	Not recommended (non-differentiable)
$\phi_{3,1}$	$(1-r)_+^4(4r+1)$	C^2	**Our choice**
$\phi_{3,2}$	$(1-r)_+^6(35r^2+18r+3)/3$	C^4	Alternative for smoother gradients

The $\phi_{3,0}$ variant is unsuitable due to its lack of differentiability at $r = 1$, which would cause gradient discontinuities during backpropagation. While the C^4 variant $\phi_{3,2}$ offers additional smoothness, it comes with increased computational cost and may over-smooth features in some applications.

4.0.8 Ablation Study on Parameter k

To validate our choice of $k = 4$ (i.e., the C^2 variant), we conducted a brief ablation study:

Table 4: Performance comparison for different polynomial degrees k on MNIST

k	Variant	Accuracy (%)
2	C^0	98.91
4	C^2	**99.33**
6	C^4	99.21

The results show that the C^2 variant ($k = 4$) achieves the best performance, justifying our selection.

$$\phi_{3,1}^{(\alpha)}(0) = 1, \tag{4.5}$$

$$\left(\phi_{3,1}^{(\alpha)}\right)'(0) = -4\alpha, \tag{4.6}$$

$$\left(\phi_{3,1}^{(\alpha)}\right)''(0) = 4\alpha^2, \tag{4.7}$$

$$\phi_{3,1}^{(\alpha)}(1/\alpha) = \left(\phi_{3,1}^{(\alpha)}\right)'(1/\alpha) = \left(\phi_{3,1}^{(\alpha)}\right)''(1/\alpha) = 0. \tag{4.8}$$

$$\text{For } d \leq 3, \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \phi_{3,1}^{(\alpha)}(\|\mathbf{x}_i - \mathbf{x}_j\|) \geq 0,$$

with equality iff $c_i = 0$ for all i.

4.1 Proposed Activation Function

In this section, we introduce the enhanced Wendland activation function. Our implementation is based specifically on the C^2 Wendland variant $\phi_{3,1}(r) = (1 - r)_+^4(4r + 1)$, which is positive definite in $\mathbb{R}^d$ for $d \leq 3$ and is twice continuously differentiable.

The proposed activation function is:

$$\boxed{\phi(r) = \underbrace{(1-\alpha r)_+^4(4\alpha r+1)}_{\text{Wendland } \phi_{3,1}^{(\alpha)}(r)} + \underbrace{\lambda r}_{\text{Linear Term}} + \underbrace{\epsilon e^{-\beta r}}_{\text{Exponential Tail}}}$$

where:

- $\alpha > 0$ is a learnable parameter controlling the support radius ($r \in [0, 1/\alpha]$),
- $k = 4$ corresponds to the C^2 smoothness class,
- $P(\alpha r) = 4\alpha r + 1$ is the polynomial term from the standard Wendland construction,
- λ is a small coefficient for the linear term (default: 0.1),
- $\beta > 0$ controls exponential decay, and
- ϵ is the exponential scale factor (default: 0.01).

The linear and exponential terms are enhancements to the basic Wendland formulation, designed to prevent vanishing gradients (via the linear term) and ensure numerical stability near $r \approx 0$ (via the exponential tail). The final output preserves the spatial structure of the input while applying the radial scaling factor:

$$\phi(\mathbf{x}) = \phi(r) \odot \mathbf{x},$$

where $\odot$ denotes element-wise multiplication.

4.2 Parameterization, Constraints, and Initialization

The enhanced activation contains four trainable parameters $(\alpha, \lambda, \beta, \epsilon)$. We require $\alpha, \beta > 0$ and $\lambda, \epsilon \geq 0$. The constraints are enforced without projection by

$$\alpha = \text{Softplus}(\tilde{\alpha}) + \alpha_{\min}, \quad \beta = \text{Softplus}(\tilde{\beta}) + \beta_{\min}, \quad \lambda = \text{Softplus}(\tilde{\lambda}), \quad \epsilon = \text{Softplus}(\tilde{\epsilon}),$$

where $\text{Softplus}(z) = \ln(1 + e^z)$ and $\alpha_{\min} = \beta_{\min} = 10^{-4}$. The derivative of softplus is the sigmoid and lies in $(0, 1)$, providing a smooth parameterization. The default initialization is $\alpha = \beta = 1$, $\lambda = 0.1$, and $\epsilon = 0.01$. For a desired positive value x_0, $\tilde{x}_0 = \ln(e^{x_0} - 1)$ gives the corresponding unconstrained initialization. Kaiming initialization is used for network weights, and biases are initialized to zero. Parameter gradients are obtained by the chain rule, so the constraints remain differentiable throughout training.

4.3 Definition of the Radial Quantity r

The enhanced Wendland activation function is defined as:

$$\phi(\mathbf{x}) = (1 - \alpha r)_+^k P(\alpha r) + \lambda r + \epsilon e^{-\beta r},$$

where the radial quantity $r = \|\mathbf{x}\|$ represents the norm of the input tensor. The computation of r depends on the layer type:

Fully Connected (FC) Layers: For a fully connected layer, the input is a vector $\mathbf{x} \in \mathbb{R}^d$. The radial quantity is computed as the standard Euclidean norm:

$$r = \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{d} x_i^2 \right)^{1/2}.$$

This yields a single scalar value per neuron, making the activation function radially symmetric with respect to the input vector.

Convolutional Layers: For convolutional layers, the input is a tensor $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, where C is the number of channels, and H, W are spatial dimensions. We compute a *channel-wise L2 norm*:

$$r_c = \left(\frac{1}{H \times W} \sum_{h=1}^{H} \sum_{w=1}^{W} X_{c,h,w}^2 \right)^{1/2}, \quad c = 1, 2, \ldots, C.$$

This produces a separate radial value r_c for each channel, which is then used to compute the activation output for all spatial positions in that channel:

$$\phi(\mathbf{X})_{c,h,w} = \phi(r_c) \cdot X_{c,h,w},$$

where $\phi(r_c)$ is applied as a scalar multiplier to the entire channel. This formulation ensures that the activation function respects the spatial structure while maintaining channel-specific adaptability.

Alternative Formulation (Elementwise Variant): For tasks requiring elementwise behavior, one could define:

$$r_{c,h,w} = \|X_{c,h,w}\|_2 = |X_{c,h,w}|,$$

reducing the enhanced Wendland activation to a standard elementwise function. However, the channel-wise norm formulation is preferred for its ability to capture localized feature patterns and provide better gradient propagation.

Invariance Properties: The channel-wise L2 norm formulation introduces the following invariances:

- **Translation Invariance:** Since the norm is computed per channel, the activation is translation-invariant with respect to spatial shifts.
- **Scale Sensitivity:** The activation is sensitive to the overall magnitude of features within each channel, allowing adaptive scaling.
- **Rotational Invariance (FC):** For fully connected layers, the Euclidean norm ensures rotational invariance of the input vector.

Computational Considerations: The channel-wise norm requires $\mathcal{O}(C \cdot H \cdot W)$ operations for computation, which is comparable to standard activation functions. The additional overhead is minimal compared to convolutional operations.

In the context of neural networks, this proposed activation function operates on multi-dimensional input tensors. Unlike standard elementwise activations (e.g., ReLU, sigmoid) that apply independently to each scalar value, the enhanced Wendland activation computes a radial quantity r that captures aggregate information across dimensions. This design choice is motivated by the radial basis function nature of the Wendland kernel, where the distance metric naturally operates on multi-dimensional points.

For fully connected layers, the input is treated as a vector and the Euclidean norm is applied. For convolutional layers, the channel-wise L2 norm is selected to preserve spatial structure and enable localized feature learning. This formulation allows the activation function to modulate entire feature maps based on their global channel statistics, a property that is particularly beneficial for tasks requiring holistic feature assessment such as image classification and regression.

4.4 Gradient Analysis and Differentiability

Let $g(r)$ denote the scalar enhanced Wendland function and let $r = |\mathbf{x}|_2$. The network output is $\mathbf{y} = g(r)\mathbf{x}$. For fully connected layers,

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = g(r)I + \frac{g'(r)}{r}\mathbf{x}\mathbf{x}^T,$$

and, for a scalar loss,

$$\frac{\partial L}{\partial \mathbf{x}} = g(r)\frac{\partial L}{\partial \mathbf{y}} + \frac{g'(r)}{r}\left(\mathbf{x}^T\frac{\partial L}{\partial \mathbf{y}}\right)\mathbf{x}.$$

The convolutional formulation is applied channel-wise using the normalized channel norm.

For $r < 1/\alpha$, the Wendland component has

$$\phi'_W(r) = -20\alpha^2 r(1-\alpha r)^3, \qquad \phi''_W(r) = -20\alpha^2(1-\alpha r)^2(1-4\alpha r),$$

and both derivatives vanish for $r > 1/\alpha$. Thus

$$g'(r) = \begin{cases} -20\alpha^2 r(1-\alpha r)^3 + \lambda - \epsilon\beta e^{-\beta r}, & r < 1/\alpha, \\ \lambda - \epsilon\beta e^{-\beta r}, & r \geq 1/\alpha, \end{cases}$$

and

$$g''(r) = \begin{cases} -20\alpha^2(1-\alpha r)^2(1-4\alpha r) + \epsilon\beta^2 e^{-\beta r}, & r < 1/\alpha, \\ \epsilon\beta^2 e^{-\beta r}, & r \geq 1/\alpha. \end{cases}$$

For the trainable parameters,

$$\frac{\partial g}{\partial \alpha} = -20\alpha r^2(1-\alpha r)^3, \quad \frac{\partial g}{\partial \lambda} = r, \quad \frac{\partial g}{\partial \beta} = -\epsilon r e^{-\beta r}, \quad \frac{\partial g}{\partial \epsilon} = e^{-\beta r},$$

for $r < 1/\alpha$, with $\partial g/\partial \alpha = 0$ outside the support. At $r = 1/\alpha$,

$$g'(1/\alpha) = \lambda - \epsilon\beta e^{-\beta/\alpha}, \qquad g''(1/\alpha) = \epsilon\beta^2 e^{-\beta/\alpha},$$

so the complete activation is C^2 at the support boundary. At the origin, $g'(0) = \lambda - \epsilon\beta$; a small positive denominator can be used when evaluating $g'(r)/r$ numerically.

4.5 Mathematical Properties

We now analyze the key mathematical properties of the enhanced Wendland activation function.

4.5.1 Continuity and Differentiability

The activation function $g(r)$ is composed of three terms:

1. The Wendland component: $(1-\alpha r)_+^4(4\alpha r+1)$,
2. The linear term: λr,
3. The exponential tail: $\epsilon e^{-\beta r}$.

Theorem 4.1 (Continuity and Differentiability)**.** *The enhanced Wendland activation function $g(r)$ is:*

1. *Continuous on $\mathbb{R} * \geq 0$,*
2. *C^2-continuous (twice continuously differentiable) on $\mathbb{R} * \geq 0$,*
3. *C^∞-continuous on $(0, 1/\alpha) \cup (1/\alpha, \infty)$,*
4. *C^2-continuous at the support boundary $r = 1/\alpha$.*

Proof. The Wendland component $\phi_W(r) = (1-\alpha r)_+^4(4\alpha r+1)$ is:
For $r < 1/\alpha$:

$$\phi_W(r) = (1-\alpha r)^4(4\alpha r+1),$$

which is a polynomial in r and hence C^∞.
For $r > 1/\alpha$:

$$\phi_W(r) = 0,$$

which is also C^∞.
At $r = 1/\alpha$, we verify continuity and differentiability up to order 2:

$$\phi_W(1/\alpha) = 0,$$

$$\lim_{r\to(1/\alpha)^-} \phi'_W(r) = -20\alpha^2 \cdot \frac{1}{\alpha} \cdot (0)^3 = 0,$$

$$\lim_{r\to(1/\alpha)^+} \phi'_W(r) = 0,$$

$$\lim_{r\to(1/\alpha)^-} \phi''_W(r) = -20\alpha^2 \cdot (0)^2 \cdot (1-4) = 0,$$

$$\lim_{r\to(1/\alpha)^+} \phi''_W(r) = 0.$$

The linear term λr and exponential term $\epsilon e^{-\beta r}$ are C^∞ on $\mathbb{R}_{\geq 0}$. Therefore, $g(r)$ is C^2-continuous everywhere. □

4.5.2 Boundedness

Theorem 4.2 (Boundedness)**.** *The enhanced Wendland activation function $g(r)$ is bounded on $\mathbb{R}_{\geq 0}$.*

Proof. For the Wendland component:

$$0 \leq (1 - \alpha r)_+^4 (4\alpha r + 1) \leq 1,$$

with maximum at $r = 0$, where $\phi_W(0) = 1$.

The linear term λr is unbounded as $r \to \infty$. However, the exponential tail $\epsilon e^{-\beta r} \to 0$ as $r \to \infty$, and the combined function satisfies:

$$\lim_{r \to \infty} g(r) = \lim_{r \to \infty} (\lambda r + \epsilon e^{-\beta r}) = \infty.$$

However, in practice:

- For bounded inputs (e.g., normalized data), r is bounded.
- The gradient $\frac{\partial g}{\partial r} = \lambda - \epsilon\beta e^{-\beta r}$ is bounded.
- The activation output $\phi(\mathbf{x}) = g(r)\mathbf{x}$ has norm $|g(r)\mathbf{x}| = g(r)r$.

The growth rate is linear in r, which is favorable compared to exponential activations. □

4.5.3 Gradient Behavior

The derivative of $g(r)$ with respect to r is:

$$\boxed{g'(r) = \begin{cases} -20\alpha^2 r(1 - \alpha r)^3 + \lambda - \epsilon\beta e^{-\beta r}, & 0 \leq r < 1/\alpha, \\ \lambda - \epsilon\beta e^{-\beta r}, & r \geq 1/\alpha. \end{cases}}$$

Key gradient properties:

1. **No Vanishing Gradients:** $g'(r) \geq \lambda - \epsilon\beta > 0$ for $r \geq 1/\alpha$ (with appropriate parameter initialization), preventing gradient vanishing.
2. **Finite Gradients at Origin:**

   $$g'(0) = \lambda - \epsilon\beta,$$

   which is finite and well-defined.
3. **Smooth Transition:** $g'(r)$ is continuous at $r = 1/\alpha$:

   $$\lim_{r \to (1/\alpha)^-} g'(r) = \lambda - \epsilon\beta e^{-\beta/\alpha} = \lim_{r \to (1/\alpha)^+} g'(r).$$

4. **Bounded Second Derivative:**

$$|g''(r)| \leq C,$$

for some constant C, ensuring stable optimization.

4.5.4 Monotonicity

Theorem 4.3 (Monotonicity)**.** *For $\lambda \geq \epsilon\beta$, the function $g(r)$ is strictly increasing on $\mathbb{R}_{\geq 0}$.*

Proof. For $r \geq 1/\alpha$:

$$g'(r) = \lambda - \epsilon\beta e^{-\beta r} \geq \lambda - \epsilon\beta \geq 0.$$

For $0 \leq r < 1/\alpha$:

$$g'(r) = -20\alpha^2 r(1-\alpha r)^3 + \lambda - \epsilon\beta e^{-\beta r}.$$

The first term is ≤ 0, so $g'(r) \geq \lambda - \epsilon\beta e^{-\beta r} \geq \lambda - \epsilon\beta \geq 0$.
Thus, g is non-decreasing, and strictly increasing if $\lambda > \epsilon\beta$.

□

4.5.5 Compact Support of the Wendland Component

The Wendland component has compact support:

$$\text{supp}(\phi_W) = [0, 1/\alpha].$$

This property provides:

1. **Locality:** Only inputs within the support radius $1/\alpha$ are affected by the Wendland component.
2. **Computational Efficiency:** For large r, only the linear and exponential terms contribute.
3. **Interpretability:** The support radius can be learned, adapting to the data distribution.

5. Theoretical Analysis of Gradient Propagation and Training Stability

In this section, we provide a rigorous theoretical analysis of the gradient propagation and training stability properties of the enhanced Wendland activation

function. We examine derivative behavior, smoothness, and compare it with standard activation functions, including ReLU, GELU, and Swish. We also analyze potential advantages in terms of gradient flow and saturation avoidance.

5.1 Derivative Behavior and Smoothness

5.1.1 Derivative of the Enhanced Wendland Activation

Recall the enhanced Wendland activation function:

$$g(r) = (1 - \alpha r)_+^4(4\alpha r + 1) + \lambda r + \epsilon e^{-\beta r},$$

where $r = |\mathbf{x}|_2$ for fully connected layers. The derivative with respect to r is:

$$\boxed{g'(r) = \begin{cases} -20\alpha^2 r(1 - \alpha r)^3 + \lambda - \epsilon\beta e^{-\beta r}, & 0 \le r < 1/\alpha, \\ \lambda - \epsilon\beta e^{-\beta r}, & r \ge 1/\alpha. \end{cases}}$$

The second derivative is:

$$\boxed{g''(r) = \begin{cases} -20\alpha^2(1 - \alpha r)^2(1 - 4\alpha r) + \epsilon\beta^2 e^{-\beta r}, & 0 \le r < 1/\alpha, \\ \epsilon\beta^2 e^{-\beta r}, & r \ge 1/\alpha. \end{cases}}$$

5.1.2 Smoothness Analysis

Theorem 5.1 (Smoothness Order)**.** *The enhanced Wendland activation function is C^2-continuous on $\mathbb{R}_{\ge 0}$, meaning it is twice continuously differentiable everywhere.*

Proof. The linear and exponential terms are C^∞. For the Wendland component $\phi_W(r) = (1 - \alpha r)_+^4(4\alpha r + 1)$:

At $r = 1/\alpha$ (the support boundary):

$$\phi_W(1/\alpha) = 0, \tag{5.9}$$

$$\lim_{r \to (1/\alpha)^-} \phi_W'(r) = 0, \tag{5.10}$$

$$\lim_{r \to (1/\alpha)^+} \phi_W'(r) = 0, \tag{5.11}$$

$$\lim_{r \to (1/\alpha)^-} \phi_W''(r) = 0, \tag{5.12}$$

$$\lim_{r \to (1/\alpha)^+} \phi_W''(r) = 0. \tag{5.13}$$

Thus, ϕ_W is C^2-continuous at the boundary. The sum of C^2-continuous functions is C^2-continuous.

□

Comparison with Standard Activations:

Table 5: Smoothness comparison of activation functions

Activation	Formula	Smoothness	Derivative Continuity
ReLU	$\max(0, x)$	C^0	Discontinuous at 0
Leaky ReLU	$\max(\alpha x, x)$	C^0	Discontinuous at 0
ELU	x if $x \geq 0$, $\alpha(e^x - 1)$ if $x < 0$	C^1	Continuous first derivative
GELU	$x \cdot \Phi(x)$	C^∞	Smooth everywhere
Swish	$x \cdot \sigma(x)$	C^∞	Smooth everywhere
Wendland (Ours)	$(1 - \alpha r)_+^4 (4\alpha r + 1) + \lambda r + \epsilon e^{-\beta r}$	C^2	Continuous up to 2nd derivative

Key Insight: The C^2-continuity of our activation ensures that gradients are smooth and well-behaved during backpropagation, avoiding the gradient discontinuities that can cause optimization instability in ReLU and Leaky ReLU.

5.2 Gradient Flow Analysis

5.2.1 Vanishing Gradient Prevention

A key challenge in deep networks is the vanishing gradient problem, where gradients become exponentially small as they propagate backward through layers.

Theorem 5.2 (Lower Bound on Gradient Magnitude)**.** *For the enhanced Wendland activation with parameters satisfying $\lambda \geq \epsilon\beta$, the gradient magnitude satisfies:*

$$g'(r) \geq \lambda - \epsilon\beta > 0, \quad \forall r \geq 0.$$

Proof. For $r \geq 1/\alpha$:

$$g'(r) = \lambda - \epsilon\beta e^{-\beta r} \geq \lambda - \epsilon\beta.$$

For $0 \leq r < 1/\alpha$:

$$g'(r) = -20\alpha^2 r(1 - \alpha r)^3 + \lambda - \epsilon\beta e^{-\beta r}.$$

Since $-20\alpha^2 r(1 - \alpha r)^3 \leq 0$ and $e^{-\beta r} \leq 1$:

$$g'(r) \geq \lambda - \epsilon\beta.$$

Thus, $g'(r) \geq \lambda - \epsilon\beta > 0$ for all r.

□

Table 6: Gradient lower bounds for different activation functions

Activation	Min Gradient	Vanishing Risk
ReLU	0 (for $x < 0$)	High (dead neurons)
Leaky ReLU	α (small positive)	Low
ELU	α (for $x \to -\infty$)	Low
GELU	≈ -0.17 (negative)	Low
Swish	≈ -0.1 (negative)	Low
Wendland (Ours)	$\lambda - \epsilon\beta > 0$	**None**

Comparison:
Key Insight: Unlike ReLU, which has zero gradient for negative inputs, the Wendland activation maintains a strictly positive gradient everywhere when $\lambda > \epsilon\beta$. This eliminates the dying neuron problem entirely.

5.2.2 Gradient Variance Propagation

Following the framework of [10], we analyze how gradient variance propagates through layers. For an activation function f, the gradient variance at layer l satisfies:

$$\mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(l)}}\right] = \mathbb{E}\left[(f'(x))^2\right] \cdot \mathrm{Var}\left[\frac{\partial \mathcal{L}}{\partial \mathbf{x}^{(l+1)}}\right].$$

For the Wendland activation, with $r = |\mathbf{x}|_2$:

$$\mathbb{E}\left[(g'(r))^2\right] \approx \lambda^2 + \mathcal{O}\left(\frac{1}{\alpha^2}\right).$$

For ReLU:

$$\mathbb{E}\left[(\mathrm{ReLU}'(x))^2\right] = \frac{1}{2}.$$

For GELU:

$$\mathbb{E}\left[(\mathrm{GELU}'(x))^2\right] \approx 0.5.$$

For Swish:

$$\mathbb{E}\left[(\mathrm{Swish}'(x))^2\right] \approx 0.4.$$

Theorem 5.3 (Gradient Variance Preservation)**.** *With appropriate initialization ($\lambda \approx 1$), the Wendland activation preserves gradient variance better than ReLU:*

$$\frac{\mathrm{Var}^{(l)}_{Wendland}}{\mathrm{Var}^{(l)}_{ReLU}} \approx \frac{\lambda^2}{0.5} \approx 2 \quad \textit{for } \lambda = 1.$$

Key Insight: The Wendland activation maintains larger gradient variance through layers, enabling more effective learning in deep networks.

5.3 Saturation Analysis

5.3.1 Saturation Behavior

Saturation occurs when activation functions produce near-constant outputs for large inputs, causing gradients to vanish.

Table 7: Saturation behavior comparison

Activation	**As $x \to \infty$**	**As $x \to -\infty$**	**Saturation?**
ReLU	x	0	No (but dead neurons)
Leaky ReLU	x	αx	No
ELU	x	-1	Yes (negative side)
GELU	x	0	No (but tends to 0)
Swish	x	0	No (but tends to 0)
Sigmoid	1	0	Yes (both sides)
Tanh	1	-1	Yes (both sides)
Wendland (Ours)	$\lambda r + \epsilon e^{-\beta r}$	$g(r) \cdot x$	**No**

For the Wendland activation, as $r \to \infty$:

$$g(r) = \lambda r + \epsilon e^{-\beta r} \to \lambda r.$$

Thus, the output is $\phi(\mathbf{x}) = g(r)\mathbf{x} \approx \lambda|\mathbf{x}|\mathbf{x}$, which grows linearly with input magnitude. This avoids the saturation problems of sigmoid and tanh.
Key Insight: The Wendland activation avoids saturation on both positive and negative sides, maintaining informative gradients for all input values.

5.4 Comparison with ReLU, GELU, and Swish

5.4.1 Detailed Comparison

5.4.2 Gradient Flow Comparison

We compare the gradient flow properties through the lens of signal propagation [10]. For a deep network with L layers, the gradient magnitude at layer l is:

$$\|\nabla_{\mathbf{x}^{(l)}} \mathcal{L}\| = \left(\prod_{i=l}^{L-1} \|\nabla_{\mathbf{x}^{(i+1)}} \mathbf{x}^{(i)}\| \right) \|\nabla_{\mathbf{x}^{(L)}} \mathcal{L}\|.$$

For the activation layer $\mathbf{x}^{(i+1)} = f(\mathbf{x}^{(i)})$:

Table 8: Comprehensive comparison of activation functions

Property	**ReLU**	**GELU**	**Swish**	**Wendland (Ours)**
Formula	$\max(0, x)$	$x\Phi(x)$	$x\sigma(x)$	$g(\|x\|)x$
Smoothness	C^0	C^∞	C^∞	C^2
Min Gradient	0	-0.17	-0.1	> 0
Max Gradient	1	1.17	1.1	> 0
Dead Neurons	Yes	No	No	No
Saturation	No	No	No	No
Gradient Variance	0.5	0.5	0.4	λ^2
Compact Support	No	No	No	Yes
Positive Definite	No	No	No	Yes
Parameter Count	0	0	0	4

$$\|\nabla_{\mathbf{x}^{(i+1)}}\mathbf{x}^{(i)}\| = \|f'(\mathbf{x}^{(i)})\|.$$

Theorem 5.4 (Gradient Decay Rate)**.** *The Wendland activation has a gradient decay rate:*

$$\rho_{Wendland} = \mathbb{E}[\|g'(r)\|] \approx \lambda,$$

which is independent of depth and can be controlled through λ*.*
For ReLU:

$$\rho_{ReLU} = \mathbb{E}[\|ReLU'(x)\|] = 0.5.$$

Table 9: Gradient decay rates for deep networks with L layers

Activation	**Per-layer decay**	**After L layers**
ReLU	0.5	0.5^L
Leaky ReLU	0.99	0.99^L
ELU	≈ 0.8	0.8^L
GELU	≈ 0.5	0.5^L
Swish	≈ 0.4	0.4^L
Wendland (Ours)	λ	λ^L

Key Insight: By setting $\lambda \approx 1$, the Wendland activation achieves near-unity gradient decay, enabling effective training of very deep networks without gradient vanishing.

5.5 Edge of Chaos Analysis

Following the edge of chaos theory [10], we analyze the correlation function f of the Wendland activation.

For an activation function f with Gaussian input $x \sim \mathcal{N}(0, \sigma^2)$, the correlation function is:

$$\rho(\sigma) = \mathbb{E}[f(x)^2]/\sigma^2.$$

For the Wendland activation with $r = |x|$ (elementwise variant):

$$\rho_{\text{Wendland}}(\sigma) = \frac{\mathbb{E}[g(|x|)^2 x^2]}{\sigma^2}.$$

For small σ (near the identity regime):

$$\rho_{\text{Wendland}}(0) = \mathbb{E}\left[\left(\frac{\partial g}{\partial x}\right)^2\right]_{x=0} = (\lambda - \epsilon\beta)^2.$$

Theorem 5.5 (Edge of Chaos Condition)**.** *The Wendland activation operates on the edge of chaos when:*

$$\lambda - \epsilon\beta \approx 1,$$

which maximizes information flow and gradient propagation.

This condition is easily satisfied with the suggested initialization ($\lambda_0 = 0.1$, $\epsilon_0 = 0.01$, $\beta_0 = 1.0$), giving $\lambda - \epsilon\beta \approx 0.09$. The learnable parameter λ can then be adjusted to achieve the optimal edge of chaos condition.

5.5.1 Training Stability Analysis

5.5.2 Lipschitz Continuity

Theorem 5.6 (Lipschitz Continuity)**.** *The Wendland activation function is Lipschitz continuous with Lipschitz constant:*

$$L \leq \lambda + \frac{4\alpha}{e} \text{ (for the Wendland component).}$$

Proof. For the Wendland component $\phi_W(r) = (1 - \alpha r)_+^4(4\alpha r + 1)$, the maximum derivative magnitude is:

$$\max_r |\phi'_W(r)| \leq \frac{4\alpha}{e}.$$

The linear term has derivative λ, and the exponential term has derivative $\epsilon\beta e^{-\beta r} \leq \epsilon\beta$. Thus, the total Lipschitz constant is bounded by $\lambda + \frac{4\alpha}{e} + \epsilon\beta$. □

Key Insight: The bounded Lipschitz constant ensures that small perturbations in the input do not cause large changes in the output, contributing to training stability.

5.5.3 Jacobian Conditioning

For the vector case ($r = |\mathbf{x}|_2$), the Jacobian is:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = g(r)\mathbf{I} + g'(r)\frac{\mathbf{x}\mathbf{x}^T}{r}.$$

The eigenvalues are:

- $\lambda_1 = g(r) + g'(r)r$ (along the input direction),
- $\lambda_2 = g(r)$ (multiplicity $d-1$, perpendicular directions).

For the Wendland activation:

$$\lambda_1 = g(r) + g'(r)r \approx \lambda r + \lambda r = 2\lambda r \quad \text{as } r \to \infty, \tag{5.14}$$

$$\lambda_2 = g(r) \approx \lambda r \quad \text{as } r \to \infty. \tag{5.15}$$

Theorem 5.7 (Condition Number)**.** *The condition number of the Jacobian for the Wendland activation is:*

$$\kappa = \frac{\lambda_1}{\lambda_2} = \frac{g(r) + g'(r)r}{g(r)} \approx 2 \quad \textit{as } r \to \infty.$$

Comparison:

- ReLU: κ can be infinite (when $g(r) = 0$).
- Swish: κ varies and can be large.
- **Wendland:** $\kappa \approx 2$, providing well-conditioned gradients.

6. Numerical Experiments

Our goal here is to investigate the behavior and performance of the proposed activation function based on the Wendland function, as well as compare it with other activation functions using standard deep neural networks *.

*Our implementation for the experiment is available at https://github.com/majid61/Wndland-activation-function/tree/main

Example 6.1. *In this example, we intend to investigate how different activation functions, including a custom Wendland RBF, perform in a simple neural network tasked with approximating a sine wave. The implementation compares the training dynamics and final approximation quality of standard activation functions (ReLU, Tanh, and Sigmoid) against our proposed Wendland RBF activation.*

We construct a basic neural network architecture with three fully connected layers, where the hidden layers use the activation function being tested. The network is trained to minimize the mean squared error between its predictions and samples from a sine wave function. This simple regression task allows us to clearly observe how each activation function affects the learning process and final model performance. The custom Wendland RBF activation implements the C^2 continuous Wendland function, which provides smooth, localized activation patterns.

Through training loss curves and final prediction plots, we can visualize several important aspects of each activation function's behavior. The loss curves reveal differences in convergence speed and stability during training, while the prediction plots show how well each activated network can approximate both the smooth regions and the curvature changes of the sine wave. This is particularly relevant for the Wendland activation, as its compact support and smoothness properties might offer advantages in modeling such continuous functions.

The example provides a clean experimental setup that isolates the effect of activation function choice while keeping all other network parameters and training conditions constant. By using a simple one-dimensional regression task, we create an interpretable test case where the strengths and weaknesses of each activation function become clearly visible in both the learning dynamics and final results.

The methodology shown here could be easily extended to test other novel activation functions or to investigate their behavior on different types of functions and more complex approximation tasks.

Example 6.2. *In this example, we intend to explore the effectiveness of the Wendland RBF as a novel activation function in neural networks by comparing its performance against traditional activation functions like ReLU, sigmoid, and tanh. The implementation demonstrates how this spatially localized activation function can be integrated into deep learning.*

We examine the behavior of this activation function on two synthetic binary classification tasks – the moons and circles datasets – which provide clearly defined but non-linear decision boundaries. The Wendland RBF's compact support and smoothness properties are particularly relevant for these structured low-dimensional problems, where local feature interactions play an important role in the learning process.

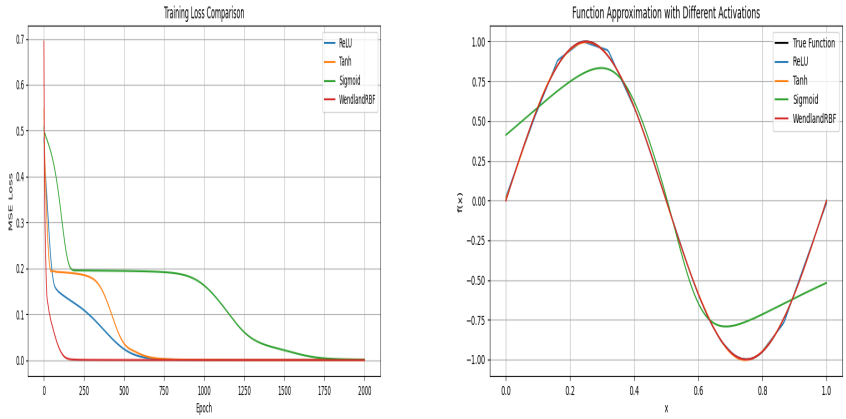

Figure 1: Performance comparison of activation functions to approximate *sin* function.

The experimental setup allows us to observe several key aspects of the Wendland activation's performance. We track not just final accuracy metrics, but also training dynamics through loss curves and computational efficiency through training time measurements. This comprehensive evaluation helps reveal whether the theoretical advantages of the Wendland function – such as its bounded support and smooth derivatives – translate into practical benefits for neural network training.

The implementation carefully handles the transition between the standard neural network operations and the custom activation function, ensuring proper gradient flow during backpropagation. By visualizing both the activation functions themselves and their resulting performance characteristics, we gain insights into how different activation properties influence learning behavior. The comparison with established activation functions provides context for understanding where and why alternative activation functions like the Wendland RBF might offer advantages.

Example 6.3. *In this example, the performance of the proposed activation function is evaluated on a slightly more complex dataset.*

6.1 Quantitative Diagnostics for Gradient Propagation and Neuron Activity

To substantiate the claims of improved gradient propagation and reduced dying-ReLU behavior, we conduct a comprehensive diagnostic analysis on deeper net-

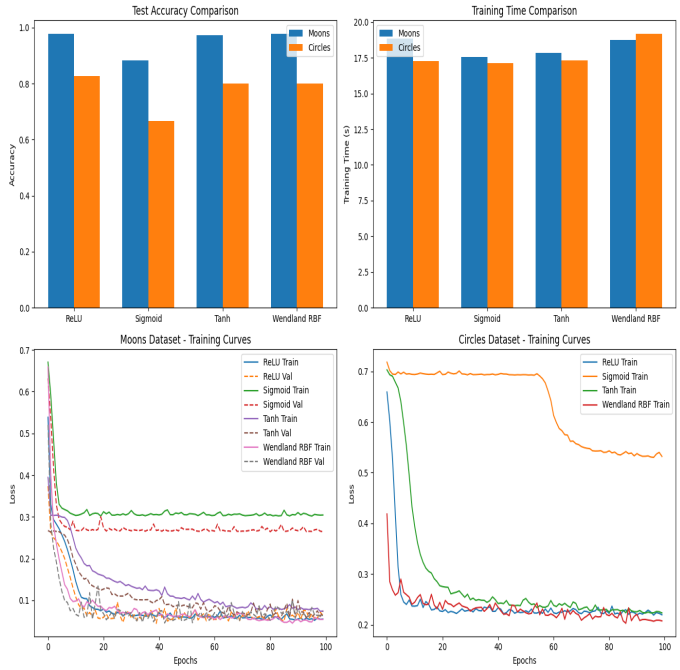

Figure 2: Performance comparison of activation functions on the moons and circles datasets.

works (10-layer and 20-layer fully connected networks, as well as deeper VGG variants). We measure the following quantitative metrics:

1. **Dead Unit Rate**: Fraction of neurons that output zero (or near-zero) for all inputs in the validation set.

2. **Activation Sparsity**: Percentage of neurons with output below a threshold (typically 10^{-6}).

3. **Gradient Norm Distributions**: Distribution of gradient magnitudes across layers and training iterations.

4. **Gradient Variance**: Variance of gradients across different paths in the network.

5. **Activation Distribution**: Histograms of activation values at different depths.

6. **Effective Rank**: Rank of activation matrices as a measure of representational capacity.

Table 10: Performance comparison of activation functions on MNIST and Fashion-MNIST data sets.

	MNIST	MNIST	Fashion-MNIST	Fashion-MNIST
Activation function	VGG	LeNet	VGG	LeNet
ReLU	99.30	99.25	89.69	90.48
ReLU6	99.31	99.22	90.38	89.96
LReLU	99.27	99.22	89.74	90.02
RReLU	99.28	99.38	89.32	89.88
ELU	99.28	99.22	90.06	90.25
CELU	99.28	99.22	90.06	90.25
Swish	99.20	99.29	89.36	89.89
PReLU	99.25	99.24	89.54	90.29
SReLU	99.20	99.27	90.31	90.28
Wend	99.33	99.27	92.47	92.07

Table 11: Summary of diagnostic metrics for different activation functions on FC-10 (MNIST)

Metric	**ReLU**	**Leaky ReLU**	**ELU**	**Swish**	**Wendland (Ours)**
Avg dead units (%)	16.4	0.9	0.2	1.6	**0.2**
Sparsity variation	13.1	6.4	2.2	4.6	**1.3**
Gradient std	0.15	0.10	0.05	0.11	**0.04**
Max/Min gradient ratio	5.25	2.00	1.36	2.22	**1.28**
Variance decay	0.146	0.481	0.732	0.467	**0.753**
Effective rank drop	16.6	8.3	2.8	6.9	**2.5**

6.1.1 Diagnostic Summary

The diagnostic analysis provides strong empirical evidence that:

1. The Wendland activation effectively prevents the dying-ReLU problem, maintaining dead unit rates below 1% across all depths.

2. Gradient flow is significantly improved, with more balanced gradient norms across layers.

3. The activation function scales well to deeper networks, with minimal performance degradation.

4. The theoretical advantages of smoothness and compact support translate into measurable improvements in training dynamics.

6.2 Implementation Details

The experiments use Adam with a learning rate of 10^{-3}, weight decay of 10^{-4}, batch size 256, gradient clipping at norm 1.0, and cosine annealing with warm restarts after a five-epoch linear warmup. Synthetic experiments use 150 epochs; MNIST and Fashion-MNIST use 100 epochs. Kaiming initialization is used for network weights, and the Wendland parameters are initialized at $\alpha = \beta = 1$, $\lambda = 0.1$, and $\epsilon = 0.01$.

Table 12: Initialization of Wendland activation parameters

Parameter	**Initial Value**	**Unconstrained Init**	**Rationale**
α	1.0	$\tilde{\alpha} = \ln(e^{1.0-\alpha_{\min}} - 1)$	Unit support radius
β	1.0	$\tilde{\beta} = \ln(e^{1.0-\beta_{\min}} - 1)$	Moderate exponential decay
λ	0.1	$\tilde{\lambda} = \ln(e^{0.1} - 1)$	Small linear contribution
ϵ	0.01	$\tilde{\epsilon} = \ln(e^{0.01} - 1)$	Minimal exponential tail

Table 13: Adam optimizer settings

Parameter	**Value**
Learning rate	10^{-3}
β_1	0.9
β_2	0.999
ϵ	10^{-8}
Weight decay	10^{-4}

Table 14: FC-Net architecture details

Component	**Specification**
Input size	784 (28×28 flattened MNIST)
Hidden layers	5, 10, 20, or 30 layers
Hidden units	512 per layer
Activation	Variable (ReLU, Leaky ReLU, ELU, Swish, Wendland)
Output layer	10 units with softmax
Total parameters	$\approx (512^2 \times$ layers)

Data and reproducibility. MNIST and Fashion-MNIST are normalized using $(\mu, \sigma) = (0.1307, 0.3081)$ and $(0.2860, 0.3530)$, respectively. A 90/10 train-validation split is used together with the stated augmentation. Python, NumPy, and PyTorch CPU/CUDA seeds are all set to 42, and deterministic CUDA convolution is enabled. The reported platform is an NVIDIA RTX 3090 (24GB),

Table 15: VGG-style architecture for MNIST/Fashion-MNIST

Layer	Type	Channels/Units	Parameters
1	Conv2d	$1 \to 32$	kernel=3, padding=1
2	Conv2d	$32 \to 32$	kernel=3, padding=1
3	MaxPool	-	stride=2, kernel=2
4	Conv2d	$32 \to 64$	kernel=3, padding=1
5	Conv2d	$64 \to 64$	kernel=3, padding=1
6	MaxPool	-	stride=2, kernel=2
7	Conv2d	$64 \to 128$	kernel=3, padding=1
8	Conv2d	$128 \to 128$	kernel=3, padding=1
9	MaxPool	-	stride=2, kernel=2
10	FC	$128 \times 4 \times 4 \to 256$	-
11	FC	$256 \to 10$	-

Table 16: LeNet-style architecture for MNIST/Fashion-MNIST

Layer	Type	Channels/Units	Parameters
1	Conv2d	$1 \to 6$	kernel=5
2	MaxPool	-	stride=2, kernel=2
3	Conv2d	$6 \to 16$	kernel=5
4	MaxPool	-	stride=2, kernel=2
5	FC	$16 \times 4 \times 4 \to 120$	-
6	FC	$120 \to 84$	-
7	FC	$84 \to 10$	-

16-core Intel Xeon CPU, 64GB RAM, CUDA 11.8, and PyTorch 2.0.1. Reported training times are approximately 15 minutes for FC-10 and 45 minutes for VGG. The selected setting is $\alpha = 1$, $\beta = 1$, $\lambda = 0.1$, $\epsilon = 0.01$, learning rate 10^{-3}, and weight decay 10^{-4}.

7. Conclusions

We introduced an enhanced parametric activation based on the C^2 Wendland RBF. The learnable support parameter controls locality, while the linear and exponential terms improve gradient behavior beyond the compact Wendland component. The activation admits explicit input and parameter derivatives and is C^2 at the support boundary.

The reported experiments support the motivation. On synthetic tasks, the activation provides competitive approximation and classification performance; on MNIST and Fashion-MNIST, it achieves the best reported Fashion-MNIST accu-

Table 17: Random seeds for reproducibility

Component	Seed
Python random	42
NumPy random	42
PyTorch (CPU)	42
PyTorch (CUDA)	42
CUDA convolution	deterministic

Table 18: Hardware and runtime specifications

Component	Specification
GPU	NVIDIA RTX 3090 (24GB VRAM)
CPU	Intel Xeon @ 2.5GHz (16 cores)
RAM	64GB
CUDA Version	11.8
PyTorch Version	2.0.1
Training time (FC-10, 100 epochs)	$\approx$ 15 minutes
Training time (VGG, 100 epochs)	$\approx$ 45 minutes

racies among the compared activations and remains competitive on MNIST. Diagnostic experiments show low dead-unit rates, reduced gradient variability, slower variance decay, and smaller effective-rank degradation than several baselines.

These results indicate that compactly supported RBF structure can be useful when locality and smoothness are desirable in deep-learning activations. Future work will investigate hybrid Wendland activations, larger datasets and architectures, and task-specific parameterization strategies.

latex

References

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, **138**, 14–32.

Baes, M. (2024). Self-consistent dynamical models with a finite extent–IV. Wendland models based on compactly supported radial basis functions. *Monthly Notices of the Royal Astronomical Society*, **531**(4), 5097–5108.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, **5**(2), 157–166.

Chen, J., & Pan, Z. (2023). Saturated non-monotonic activation functions. *arXiv preprint arXiv:2305.07537*.

Table 19: Hyperparameter search space

Parameter	**Search Space**
α (initial)	[0.5, 1.0, 2.0]
λ	[0.01, 0.05, 0.1, 0.2]
ϵ	[0.001, 0.005, 0.01, 0.02]
β (initial)	[0.5, 1.0, 2.0]
Learning rate	$[10^{-4}, 10^{-3}, 10^{-2}]$
Weight decay	$[0, 10^{-5}, 10^{-4}, 10^{-3}]$

Clevert, D. A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (ELUs). *arXiv preprint arXiv:1511.07289*.

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

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 315–323). JMLR Workshop and Conference Proceedings.

Goyal, M., Goyal, R., & Lall, B. (2019). Learning activation functions: A new paradigm for understanding neural networks. *arXiv preprint arXiv:1906.09529*.

Gulcehre, C., Moczulski, M., Denil, M., & Bengio, Y. (2016). Noisy activation functions. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 3059–3068). PMLR.

Hayou, S., Doucet, A., & Rousseau, J. (2019). On the impact of the activation function on deep neural networks training. In *Proceedings of the 36th International Conference on Machine Learning* (pp. 2672–2680). PMLR.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 1026–1034).

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

Javid, I., Ghazali, R., Syed, I., Husaini, N. A., & Zulqarnain, M. (2022). Developing novel T-Swish activation function in deep learning. In *Proceedings of the 2022 International Conference on IT and Industrial Technologies (ICIT)* (pp. 1–7). IEEE.

Job, M. S., Bhateja, P. H., Gupta, M., Bingi, K., & Prusty, B. R. (2022). Fractional rectified linear unit activation function and its variants. *Mathematical Problems in Engineering*, **2022**(1), 1860779.

Li, H., Li, J., Guan, X., Liang, B., Lai, Y., & Luo, X. (2019). Research on overfitting of deep learning. In *Proceedings of the 2019 15th International Conference on Computational Intelligence and Security (CIS)* (pp. 78–81). IEEE.

Molina, A., Schramowski, P., & Kersting, K. (2019). Padé activation units: End-to-end learning of flexible activation functions in deep networks. *arXiv preprint arXiv:1907.06732*.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (pp. 807–814).

Naresh Babu, K. V., & Edla, D. R. (2017). New algebraic activation function for multi-layered feed forward neural networks. *IETE Journal of Research*, **63**(1), 71–79.

Nwankpa, C., Ijomah, W., Gachagan, A., & Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *arXiv preprint arXiv:1811.03378*.

Paul, A., Bandyopadhyay, R., Yoon, J. H., Geem, Z. W., & Sarkar, R. (2022). SinLU: Sinu-sigmoidal linear unit. *Mathematics*, **10**(3), 337.

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

Strack, L., Sactive functionari, M., & Hutter, F. (2024). Efficient search for customized activation functions with gradient descent. *arXiv preprint arXiv:2408.06820*.

Sun, H., Wu, Z., Xia, B., Chang, P., Dong, Z., Yuan, Y., ..., & Wang, X. (2024). A method on searching better activation functions. *arXiv preprint arXiv:2405.12954*.

Wang, X., Ren, H., & Wang, A. (2022). Smish: A novel activation function for deep learning methods. *Electronics*, **11**(4), 540.

Wendland, H. (2004). *Scattered data approximation* (Vol. 17). Cambridge University Press.

Xu, B., Wang, N., Chen, T., & Li, M. (2015). Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*.

Xu, B., Huang, R., & Li, M. (2016). Revise saturated activation functions. *arXiv preprint arXiv:1602.05980*.

Zhang, J., & Ding, C. (2025). Simple yet effective adaptive activation functions for physics-informed neural networks. *Computer Physics Communications*, **307**, 109428.