INTERNATIONAL JOURNAL OF LATEST TECHNOLOGY IN ENGINEERING, MANAGEMENT & APPLIED SCIENCE (IJLTEMAS)

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

HyperNova++: A Novel Adaptive Activation Function for High-Accuracy Neural Learning on Nonlinear Synthetic Decision Manifolds

Sourish Dey, Sunil Kumar Sawant, Arunima Dutta, Abhradeep Hazra

KIIT University, Bhubaneswar, Odisha, India

DOI : https://doi.org/10.51583/IJLTEMAS.2025.1412000109

Received: 27 December 2025; Accepted: 01 January 2026; Published: 10 January 2026

ABSTRACT

Activation functions are at the heart of how deep neural networks perform non-linear transformations. The use of an activation function allows a neural network to approximate highly complex functions, train using a gradient-based optimization technique and generalize to new data. However, existing activation functions, such as ReLU, GELU, and Swish, have limitations that restrict their use in practice. Specifically, they can saturate gradients during training due to their inherent structure, cause vanishing gradients on deeply stacked architectures, and are inefficient at learning periodic dependency relationships while performing poorly at modeling highly heterogeneous non-linear interactions. These limitations are of particular importance for scientific, financial, and engineering use cases where data represent polynomial, periodic, saturating, and exponential shapes on the same data manifold.

This paper introduces HyperNova++, a smooth, adaptive, parameterized activation function that unifies bounded saturation, periodic oscillation, and unbounded growth into a single learnable formula. HyperNova++ is architectured and designed to overcome the expressive constraints of existing activations which enables dynamic, data-driven modulation of curvature, frequency, and growth behavior using three trainable parameters (α,β,γ). These above mentioned parameters respectively govern contributions from the hyperbolic tangent (tanh) for bounded saturation, sine (sin) for periodic oscillations, and Softplus ($\log(1+e^x)$) for getting a smooth monotonic growth all thorughout. The resulting function obtained ensures non-vanishing gradients, smooth transitions, and controlled Lipschitz continuity, along with maintaining computational efficiency comparable to contemporary activations and other counterparts.

After doing a rigorous, large-scale evaluation on a meticulously crafted synthetic dataset with a known ground-truth decision boundary that stimulates reall life linear, polynomial, and periodic interactions. This controlled environment enables precise, unbiased comparisons against various functions including ReLU, GELU, and Swish under identical architectural, optimization, and hyperparameter settings. HyperNova++ achieves statistically significant superior performance compared to all , exceeding 99% accuracy (0.9903) compared to 98.34% for ReLU, 98.08% for GELU, and 97.60% for Swish, while also attaining the highest F1-score (0.9906) and ROC-AUC (0.9997). Gradient analyses obtained confirm stable, non-vanishing gradients and accelerated convergence.

We supplement empirical results obtained during testing with comprehensive theoretical analysis, thus establishing HyperNova++'s universal approximation guarantee, Lipschitz properties, gradient bounds, and optimization landscape characteristics. Practical implementation guidelines, computational complexity dissections, and prospective applications in scientific machine learning, time-series analysis, and multimodal inference are being discussed. Collectively, this work positions HyperNova++ as a potent, versatile activation function for advanced deep learning architectures confronting intricate nonlinear manifolds in upcoming future.

Index Terms—Activation Function, Deep Learning, Hyper-Nova++, Neural Networks, Nonlinear Modeling Synthetic Dataset, ROC-AUC Curve, Optimization, Adaptive Activation, Mixed Nonlinearities, Universal Approximation, Lipschitz Continuity

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

INTRODUCTION

A. The Central Role of Activation Functions in Deep Learning

There is a paradigm shift in computational science, machine learning, and artificial intelligence has been sparked by deep neural networks. The success is essentially based on the combination of nonlinear activation functions and affine transformations, which allow complex function approximation and hierarchical feature extraction. Although fully connected, convolutional, and attention-based linear layers offer structural scaffolding, it is the activation functions that give networks their nonlinear expressive power, which is formally embodied in universal approximation theorems.

Activation functions serve four intertwined roles:

1. Nonlinear Transformation: They break linearity, allowing networks to model intricate, non-additive interactions between input features.

2. Gradient Flow Regulation: During backpropagation, activation derivatives modulate gradient magnitudes, directly influencing optimization stability and convergence rates.

3. Output Range Control: They determine neuron output bounds, affecting regularization dynamics (e.g., sparsity, saturation) and generalization.

4. Loss Landscape Geometry: Activation functions shape the curvature and topology of the loss surface, guiding optimization trajectories toward desirable minima.

Thus, the choice of activation function is a pivotal architectural decision with profound implications for model capacity, trainability, and task performance.

B. Historical Evolution and Current Limitations

The evolution ofthe activation functions projects the trajectory of neural network research. Early networks employed sigmoid ($\sigma(x) = 1/(1 + e^{-x})$) and hyperbolic tangent ($\tanh(x) = (e^{x} - e^{-x})/(e^{x} + e^{-x})$), which offered smooth, bounded nonlinearities. Even though , these saturating functions suffer from the vanishing gradient problem: as $|x| \rightarrow$

∞, derivatives approach zero, stalling learning in deep layers. The introduction of the Rectified Linear Unit (ReLU), $f(x) = \max(0,x)$, marked a watershed. Its piecewise linearity provided:

- Non-saturating behavior for $x > 0$, mitigating gradient vanishing.
- Sparse activations, inducing implicit regularization.
- Computational efficiency (comparison and multiplication only).

Yet, ReLU's limitations are well-documented:

- Dying ReLU: Neurons with consistently negative inputs become permanently inactive.
- Unbounded Growth: Positive inputs yield linear, potentially explosive activations.
- Non-differentiability at zero (handled via subgradients).
- Lack of negative values, limiting representational symmetry.

Subsequent variants sought to address these issues:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- Leaky ReLU ($f(x) = \max(\alpha x, x)$, $\alpha > 0$) prevents dead neurons.
- Parametric ReLU (PReLU) makes the negative slope learnable.
- Exponential Linear Unit (ELU) smooths negative saturation.
- Scaled Exponential Linear Unit (SELU) enables selfnormalizing networks.
- Swish ($f(x) = x \cdot \sigma(x)$), discovered via automated search, offers smooth non-monotonicity.
- Gaussian Error Linear Unit (GELU) ($f(x) = x \cdot \Phi(x)$, Φ Gaussian CDF) incorporates stochastic regularization.

Despite these advances, contemporary activations remain fundamentally limited:

1. Limited Periodic Expressiveness: Most functions lack explicit periodic components, rendering them ill-suited for oscillatory patterns ubiquitous in time-series (seasonality), spatial data (textures, waves), and scientific phenomena (harmonic motion).
2. Inflexible Nonlinear Regimes: Existing activations specialize in specific behaviors (ReLU: piecewise linearity; tanh: saturation; Softplus: smooth growth) but cannot adaptively combine multiple regimes.
3. Static Formulations: With few exceptions (e.g., PReLU), activation functions have fixed functional forms throughout training, unable to evolve nonlinear characteristics in response to learned representations.
4. Suboptimal Gradient Dynamics: Many functions still exhibit problematic gradient properties in very deep or complex architectures, particularly when learning highly nonlinear decision manifolds.

C. The Challenge of Mixed Nonlinear Manifolds

Real-world data distributions frequently exhibit heterogeneous nonlinear characteristics that challenge monolithic activation paradigms. Consider:

- Scientific Computing: Physical systems obey equations combining polynomial terms (Newtonian mechanics), periodic components (wave equations), saturation effects (material yield points), and exponential relationships (radioactive decay).
- Financial Time Series: Display linear trends, periodic seasonality, volatility clustering (nonlinear dependence), and regime-switching behavior.
- Natural Language: Embodies syntactic hierarchies, semantic associations, and pragmatic constraints, each with distinct mathematical signatures.

Standard activations find it difficult to effectively model the complex, high-dimensional decision manifolds created by these mixed nonlinear patterns. Networks must either rely on specialized architectures with domain-specific inductive biases (limiting generality) or use excessively deep architectures to approximate these manifolds via compositions of simpler functions (inefficient). This encourages the creation ofexpressive, adaptive activation functions capable of directly modeling diverse nonlinear regimes within individual units.

D. Contributions and Paper Organization

This paper introduces HyperNova++, a novel adaptive activation function designed to transcend the limitations of existing approaches through strategic combination of multiple nonlinear components with learnable mixing parameters. Our contributions are multifaceted:

Mathematical Formulation: We propose Hyper Nova++ defined as,

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

$\phi(x) = \alpha\tanh(x) + \beta\sin(x) + \gamma\log(1 + e^x)$,

with learnable $\alpha,\beta,\gamma \in R$ controlling bounded saturation, periodic oscillation, and smooth growth, respectively.

Theoretical Analysis: We provided a comprehensive analysis of HyperNova++'s properties: gradient behavior, Lipschitz continuity, universal approximation capabilities, optimization landscape characteristics, and parameter learning dynamics.

Empirical Validation: Based on the extensive experiments on a synthetic dataset with ground-truth nonlinear decision boundaries, we demonstrate HyperNova++'s superiority over ReLU, GELU, and Swish across accuracy, F1-score, ROC-AUC, and convergence speed.

Implementation Guidelines: Through these paper we offer practical recommendations for integration into existing architectures, including initialization strategies, regularization techniques, and computational considerations.

Future Directions: We outline promising research avenues: theoretical generalization bounds, applications to real-world domains, and integration with transformers and graph neural networks.

RELATED WORK

A. Classical Activation Functions

Sigmoid and Hyperbolic Tangent: The sigmoid function as $\sigma(x) = 1/(1 + e^{-x})$ and $\tanh(x)$ were dominant in early neural networks. Sigmoid maps to $(0,1)$, suitable for probability outputs; tanh centers outputs around zero, improving optimization. Both suffer from vanishing gradients as $|x| \to \infty$, since derivatives $\sigma'(x) = \sigma(x)(1 - \sigma(x))$ and $\tanh'(x) = 1 - \tanh^2(x)$ approach zero. This limitation hindered deep network training until alternative activations emerged.

Rectified Linear Unit (ReLU) and Variants: ReLU's breakthrough addressed vanishing gradients for positive inputs while maintaining simplicity. Variants include:

- Leaky ReLU [11]: Small negative slope prevents dead neurons.
- Parametric ReLU (PReLU) [3]: Learnable negative slope.
- Randomized ReLU (RReLU) [16]: Random slopes during training for regularization.
- Exponential Linear Unit (ELU) [18]: Smooth saturation for negative inputs.
- Scaled Exponential Linear Unit (SELU) [7]: Enables self-normalizing networks via specific scaling.
- Gaussian Error Linear Unit (GELU) [4]: $f(x) = x \cdot \Phi(x)$, inspired by dropout.
- Swish [13]: $f(x) = x \cdot \sigma(x)$, discovered via automated search.

B. Adaptive and Learnable Activations

Parameterizing activation functions dates to early neural networks but gained momentum with PReLU. Categories include:

1. Parametric Functions: Learnable parameters within a fixed form (e.g., PReLU, S-shaped ReLU).
2. Mixture Models: Weighted combinations of basis functions (e.g., Adaptive Blending Units, Maxout).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

3. Neural Activation Functions: Small networks generate activation values (e.g., Learning Activation Functions, Activation Networks).

4. Search-Based Approaches: Reinforcement learning or evolutionary algorithms discover activation forms (e.g., Swish, EvoNorms).

Maxout Networks [2]: $f(x) = \max_{i \in [1,k]}(w_i^T x + b_i)$ provides piecewise linear approximation with learnable regions but increases parameters k-fold.

Adaptive Blending Units (ABU) [19]: Learn weighted combinations of basis functions (sigmoid, tanh, sin) but with fixed mixing weights.

C. Periodic and Oscillatory Activations

Periodic functions have gained traction in implicit neural representations (INRs) for 3D reconstruction, image representation, and signal processing:

- Sine-Based Networks [14]: Pure sine activations with careful initialization to preserve frequencies.
- Fourier Features [15]: Map inputs to high-frequency domains before standard activations.
- Wavelet Networks: Combine wavelet basis functions with neural architectures.

These approaches typically specialize in representing continuous signals rather than general classification/regression tasks.

D. Theoretical Foundations

Universal Approximation Theorems:

- Cybenko [1], Hornik [5]: Single hidden layer networks with sigmoidal activations can approximate any continuous function on compact sets.
- Leshno et al. [8]: Non-polynomial continuous activations are sufficient for universal approximation.
- ReLU networks require depth rather than extreme width for universal approximation [17].

Expressivity Hierarchies:

- Non-polynomial activations generally offer superior approximation efficiency.
- Piecewise linear functions (ReLU) create $O(n^k)$ linear regions from n neurons in k layers [12].
- Smooth activations enable better gradient flow and optimization landscape navigation.

E. Synthetic Datasets for Activation Evaluation

While most activation research evaluates on standard benchmarks (MNIST, CIFAR, ImageNet), these datasets contain unknown, complex distributions that complicate controlled comparison. Synthetic datasets with known properties enable precise evaluation:

- Two-Moons and Circle Datasets: Test nonlinear separability.
- Function Approximation Tasks: Regression to known mathematical functions.
- Synthetic Manifolds: Controlled curvature, dimensionality, and noise test representation learning.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Our approach extends this methodology with a sophisticated synthetic decision boundary incorporating multiple nonlinear interaction types.

F. Position Relative to Existing Work

HyperNova++ distinguishes itself through:

1. Unified Multi-Regime Modeling: Unlike specialized periodic or saturated activations, HyperNova++ explicitly combines three distinct nonlinear regimes (bounded, periodic, unbounded) within a single, coherent formulation.

2. Learnable Adaptive Mixing: Parameters α,β,γ are learned from data rather than fixed, allowing the network to emphasize different nonlinear aspects across layers, channels, or training phases.

3. Balanced Expressivity and Stability: While highly expressive, HyperNova++ maintains favorable optimization properties via smoothness, controlled gradients, and numerical stability.

4. General-Purpose Design: Unlike periodic activations tailored for INR tasks, HyperNova++ targets general deep learning while retaining oscillatory capability.

5. Theoretical Grounding: We provide comprehensive mathematical analysis beyond empirical validation.

PROPOSED METHOD: HYPERNOVA++ ACTIVATION FUNCTION

A. Mathematical Formulation

1) Core Definition: Let $x \in \mathbb{R}$ be the input. The HyperNova++ activation is defined as:

$\phi(x) = \alpha\tanh(x) + \beta \sin(x) + \gamma \log(1 + e^x)$, (1) where $\alpha,\beta,\gamma \in \mathbb{R}$ are learnable parameters controlling contributions from:

1) Bounded Saturation (tanh):

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

provides anti-symmetric saturation to $[-1,1]$, smooth transitions, and zero-centering. 2) Periodic Oscillation (sin):

$$\sin(x)$$

captures cyclic patterns with period 2π, bounded range $[-1,1]$, and infinite differentiability.

3) Smooth Unbounded Growth (Softplus):

$$\text{Softplus}(x) = \log(1 + e^x)$$

approximates ReLU for $x \gg 0$ ($\log(1 + e^x) \approx x$), provides smooth differentiability everywhere, and exhibits logarithmic growth for $x \ll 0$ ($\log(1 + e^x) \approx e^x$).

2) Parameter Interpretation and Constraints: Parameters α,β,γ serve as mixing coefficients, not constrained to be positive or sum to one, enabling flexible, potentially canceling combinations:

- $\alpha > 0$: Emphasizes saturation; negative α inverts saturation direction.
- $\beta > 0$: Emphasizes periodic components; magnitude controls oscillation amplitude.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- $\gamma > 0$: Emphasizes growth; typically positive to maintain ReLU-like monotonicity for $x > 0$.

Practical Considerations:

Initialization: Small positive values (e.g., $\alpha = 0.3, \beta =$

$0.3, \gamma = 0.4$) encourage balanced contributions.

Regularization: L1/L2 on parameters prevents extreme values that could destabilize training.

Batch Normalization: Mitigates parameter scaling issues.

3) Alternative Formulations Considered: During development, we evaluated alternatives:

Weighted Sum with Sigmoid Gating:

$$\phi(x) = \sigma(w_1)\tanh(x)+\sigma(w_2)\sin(x)+\sigma(w_3)\text{Softplus}(x)$$

limited expressivity by preventing component cancellation.

Input-Dependent Mixing:

$\phi(x) = \alpha(x)\tanh(x) + \beta(x)\sin(x) + \gamma(x)\text{Softplus}(x)$,

where α, β, γ are small neural networks. Increased computational cost and overfitting risk.

Additional Components: Linear (x), quadratic (x^2), Gaussian (e^{-x2}) either reduced to existing functions or introduced undesirable properties.

The chosen formulation balances expressivity, computational efficiency, and optimization stability.

Gradient Analysis and Optimization Properties 1) First Derivative:

$$\phi'(x) = \alpha(1 - \tanh^2(x)) + \beta \cos(x) + \gamma\sigma(x), \qquad (2)$$

where $\sigma(x) = 1/(1 + e^{-x})$ is the sigmoid function. Component Contributions:

- tanh derivative: $1-\tanh^2(x)$, a bell-shaped curve peaked at zero with range $(0, 1]$.
- sin derivative: $\cos(x)$, periodic oscillation with amplitude 1.
- Softplus derivative: $\sigma(x)$, smooth step from 0 to 1.

The gradient thus combines smooth decay (tanh), oscillatory components (cos), and monotonic transition (σ), ensuring non-vanishing gradients across most of the input domain.

Gradient Range and Behavior: [Gradient Bound] For any $x \in \mathbb{R}$,

$$|\phi'(x)| \leq |\alpha| + |\beta| + |\gamma|.$$

Each component has derivative bounded by 1:

- $|\frac{d}{dx}\tanh(x)| = |1 - \tanh^2(x)| \leq 1$,
- $|\frac{d}{dx}\sin(x)| = |\cos(x)| \leq 1$,

Softplus$(x)| = \sigma(x) \leq 1$.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

By the triangle inequality, $|\phi'(x)| \leq |\alpha| + |\beta| + |\gamma|$.

Thus, the gradient is globally bounded, preventing explosion when parameters are reasonably constrained. For initialization $\alpha,\beta,\gamma \approx 0.3$, maximum gradient ≈ 0.9, slightly less than ReLU's gradient of 1 for $x > 0$.

Critical Gradient Properties:

Non-Zero Gradients Almost Everywhere: Unlike ReLU (zero for $x < 0$) or tanh (asymptotically zero for $|x| \gg 0$), HyperNova++ maintains non-trivial gradients due to sin's persistent oscillations.

Smooth Transitions: All components are C^{∞}, ensuring smooth gradient flow.

Controlled Sensitivity: Bounded gradient prevents extreme sensitivity to input perturbations, improving robustness.

Second Derivative and Curvature:

Thus:

$\phi(x) \approx -\alpha + \beta \sin(x)$ (oscillates around $-\alpha$).

[Unbounded Growth Direction] For $\gamma \neq 0$, $\phi(x)$ is unbounded in the direction of $\text{sign}(\gamma) \cdot \infty$.

For $\gamma > 0$, as $x \to +\infty$, $\gamma \log(1+e^x) \sim \gamma x \to +\infty$; other terms remain bounded. For $\gamma < 0$, as $x \to +\infty$, $\phi(x) \to -\infty$.

In contrast to bounded functions such as sigmoid or tanh, this is consistent with ReLU-like unbounded growth for positive inputs when$\gamma > 0$.

Universal Approximation Capability: [Universal Approximation] Neural networks with HyperNova++ activations and a single hidden layer containing sufficiently many units can approximate any continuous function on a compact subset of R^n to arbitrary precision.

The standard universal approximation theorem [8] requires the activation function to be non-polynomial and continuous. HyperNova++ contains $\sin(x)$, which is non-polynomial and oscillatory. Following Leshno's proof strategy, any nonpolynomial continuous function enables universal approximation in single-hidden-layer networks. The combination with other components preserves this property.

Moreover, HyperNova++ likely offers superior approximation efficiency compared to standard activations due to $\phi''(x) = -2\alpha\tanh(x)(1-\tanh 2(x))-\beta \sin(x)+\gamma\sigma(x)(1-\sigma(x))$its richer functional repertoire, though formal quantification.

(3) Component Contributions:

tanh: $-2\tanh(x)(1 - \tanh^2(x))$, negative near zero

(concave down), positive for larger $|x|$ (inflection points). • sin: $-\beta \sin(x)$, pure oscillation between $-\beta$ and β.

Softplus: $\gamma\sigma(x)(1 - \sigma(x))$, bell-shaped curve peaked at zero.

This rich curvature profile enables modeling of complex decision boundaries with varying convexity/concavity patterns.

C. Key Mathematical Properties

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Continuity and Differentiability: [Smoothness] $\phi(x) \in C^\infty(\mathrm{R})$, i.e., infinitely differentiable for all real x. $\tanh(x)$, $\sin(x)$, and $\log(1+e^x)$ are each C^∞ on R. Linear

combinations of C^∞ functions remain C^∞.

This smoothness ensures stable gradient computation and enables higher-order optimization methods if desired.

1) *Bounds and Asymptotic Behavior:* As $x \to +\infty$:

$\tanh(x) \to 1$,

$\sin(x)$ oscillates in $[-1,1]$,

$\log(1 + e^x) \approx x$.

Thus:

$\phi(x) \approx \alpha \cdot 1 + \beta \sin(x) + \gamma x \xrightarrow{x\to\infty}$ dominated by γx.

As $x \to -\infty$:

$\tanh(x) \to -1$,

$\sin(x)$ oscillates in $[-1,1]$,

$\log(1 + e^x) \approx e^x \to 0$.

requires further theoretical development.

Lipschitz Continuity: [Lipschitz Constant] $\phi(x)$ is Lipschitz continuous with constant $L \le |\alpha| + |\beta| + |\gamma|$. Each component has Lipschitz constant 1:

tanh: $\sup|\tanh'(x)| = \sup|1 - \tanh^2(x)| = 1$,

sin: $\sup|\sin'(x)| = \sup|\cos(x)| = 1$, • Softplus: $\sup|\mathrm{Softplus}'(x)| = \sup\sigma(x) = 1$.

By the triangle inequality, the Lipschitz constant of the sum is bounded by the sum of individual constants.

This ensures numerical stability and provides regularization benefits similar to spectral normalization techniques.

D. Learning Dynamics and Parameter Adaptation

Gradient Flow for Learnable Parameters: During backpropagation, gradients for α,β,γ are:

$$\frac{\partial \phi}{\partial \alpha} = \tanh(x), \quad \frac{\partial \phi}{\partial \beta} = \sin(x), \quad \frac{\partial \phi}{\partial \gamma} = \log(1 + e^x).$$

These gradients reveal adaptation mechanisms:

α updates depend on $\tanh(x)$, encouraging larger α when inputs are in the active region ($-2 < x < 2$) where tanh varies significantly.

β updates depend on $\sin(x)$, oscillating with input, allowing β to capture periodic patterns.

γ updates depend on $\mathrm{Softplus}(x)$, growing with x, causing γ to increase when modeling unbounded relationships.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

2) Implicit Regularization through Parameter Evolution:

Learning dynamics encourage specialization:

Early training: Gradients are large, potentially increasing all parameters.

As network learns: Parameters adjust to emphasize relevant components:

Saturating patterns (e.g., classification confidence)

→ α grows.

Periodic patterns → β adapts to match frequency/amplitude.

Unbounded relationships → γ increases.

Components with minimal contribution may see parameters shrink toward zero, effectively pruning unnecessary nonlinearities.

This adaptive behavior resembles automatic relevance determination applied to nonlinear basis functions.

Interaction with Batch Normalization: Batch normalization [6] is particularly beneficial:

Input Normalization: Ensures inputs to $\phi(x)$ have ~ zero mean and unit variance, keeping inputs in regimes where all components are active.

Parameter Stabilization: Prevents extreme values that could drive parameters to extremes.

Frequency Control: For $\sin(x)$, input scaling affects effective frequency: $\sin(\sigma x)$ has frequency σ relative to $\sin(x)$. Batch normalization's scaling parameter allows learned frequency adjustment.

We recommend placing batch normalization before HyperNova++ in most architectures.

E. Computational Considerations

1) Forward Pass Complexity: Per activation computational cost:

$\tanh(x)$: Typically via exp (2 exponentials, 2 additions, 1 division).

$\sin(x)$: Standard trigonometric function (various approximations).

$\log(1 + e^x)$: Softplus (1 exponential, 1 addition, 1 logarithm).

Total operations: ~3-5× more than ReLU but comparable to GELU or Swish, which also require exponentials. Modern hardware (GPUs, TPUs) efficiently computes these operations, making overhead acceptable for most applications.

Memory Requirements: HyperNova++ requires storing three additional parameters (α,β,γ) per activation instance. Practical configurations:

Per-layer sharing: Parameters shared across all neurons in a layer (3 parameters per layer).

Per-channel sharing: For convolutional networks, parameters shared per channel ($3 \times C$ parameters for C channels).

Per-neuron specialization: Each neuron has unique parameters (dramatically increases parameter count: 3 × #neurons).

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

We typically recommend per-layer or per-channel sharing as a balance between flexibility and efficiency.

Numerical Stability: Potential issues and mitigations:

Large x in Softplus: $\log(1 + e^x)$ for $x > 709$ (single precision) overflows. Use stable implementation:

$$(x) = \begin{cases} x + \log(1 + e^{-x}) & \text{if } x > 0, \\ \log(1 + e^x) & \text{otherwise} \end{cases}$$

Softplus

Large β $\sin(x)$ oscillations: Extreme β values cause rapid oscillations that may hinder optimization. Regularization (weight decay) on β prevents this.

Parameter drift: Unconstrained parameters could grow extremely large. L2 regularization or clipping maintains stability.

F. Special Cases and Connections to Existing Functions

HyperNova++ generalizes several existing activations as special cases:

- $\alpha = 1, \beta = 0, \gamma = 0$: $\phi(x) = \tanh(x)$.
- $\alpha = 0, \beta = 1, \gamma = 0$: $\phi(x) = \sin(x)$ (periodic activation).
- $\alpha = 0, \beta = 0, \gamma = 1$: $\phi(x) = \text{Softplus}(x)$.
- $\alpha = 0, \beta = 0, \gamma \to$ large, $x > 0$: $\phi(x) \approx \gamma x$ (linear).

$\beta = 0, \gamma \approx 1, \alpha$ small: $\phi(x) \approx \text{Softplus}(x)$ + small tanh ≈ Swish-like.

$\gamma = 0, \alpha \approx \beta$: $\phi(x) \approx \alpha(\tanh(x) + \sin(x))$ (oscillatory saturation).

Thus, HyperNova++ can adaptively approximate many existing functions based on learned parameters. Observations:

- Balanced parameters ($\alpha = \beta = \gamma = 0.33$): Rich shape with saturation, oscillation, and growth.
- tanh-dominated ($\alpha = 1, \beta = \gamma = 0.1$): Primarily saturating S-curve.
- sin-dominated ($\beta = 1, \alpha = \gamma = 0.1$): Pure oscillation.
- Softplus-dominated ($\gamma = 1, \alpha = \beta = 0.1$): Smooth

ReLU-like growth.

Negative parameters: Invert or phase-shift components.

The visual diversity demonstrates HyperNova++'s expressive range.

G. Initialization Strategies

Proper initialization of α, β, γ is crucial:

Default: $\alpha = 0.3, \beta = 0.3, \gamma = 0.4$ (slightly emphasizes growth).

Xavier/Glorot-inspired: Scale parameters inversely with fan-in:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

$\alpha,\beta,\gamma \sim \text{Uniform}\left(-\sqrt{\frac{3}{\text{fan\_in}}}, \sqrt{\frac{3}{\text{fan\_in}}}\right)$.

He-inspired: For ReLU-like emphasis, set γ = $q\ 2$ fan $\overline{\text{in}}, \alpha = \beta = 0.1\gamma$.

Learned initialization: Meta-learn initial values on similar tasks.

We found default initialization works robustly across diverse architectures.

H. Extension to Multidimensional Inputs

For vector inputs $x \in \mathbb{R}n$:

application: $\phi(x_i)$ independently per di-

mension (standard approach).

Multidimensional mixing:

$\phi(x) = {}^{X}\alpha_i \tanh(w_i{}^T x) + {}^{X}\beta_j \sin(v_j{}^T x) + {}^{X}\gamma_k \log(1+\exp(u3)^T{}_k x))$*Dataset Characteristics:*

i j k

with learnable weight vectors (more expressive but parameter-heavy).

Tensor-product combinations: Even more expressive but computationally intensive.

We focus on elementwise application for compatibility with existing architectures.

• Polynomial interaction $0.3x_1x_2$: Quadratic interaction between first two features.

• Noise term ϵ: Realistic label noise. 1) Elementwise

This creates a challenging, highly nonlinear decision manifold requiring simultaneous modeling of multiple interaction types—precisely where HyperNova++ should excel.

Experimental Methodology

A. Dataset Construction

1) Motivation for Synthetic Data: Real-world benchmarks (MNIST, CIFAR, ImageNet) present challenges for controlled activation evaluation:

Unknown Ground Truth: Data-generating process is complex, obscuring attribution of performance differences.

Multiple Confounding Factors: Noise, missing values, distribution shifts, irrelevant features.

Limited Nonlinear Transparency: Specific types and mixtures of nonlinearities are not characterized.

Thus, we construct a synthetic dataset with known, controllable properties to test activation capacity for modeling mixed nonlinear decision boundaries.

2) Data Generation Process: Binary classification data with ground-truth decision boundary incorporating linear, polynomial, and periodic interactions:

True decision function:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

$$f(x) = w^T x + 0.6 \sum_{i=1}^{8} sin(x_i) + 0.3x_{1x_2} + b + \epsilon$$

Binary labels:

$$y = \mathrm{I}(f(x) > 0)$$

where:

- $x \in \mathrm{R}^d$ ($d = 20$), $x_i \sim \mathrm{N}(0,1)$ independently.
- $w \in \mathrm{R}^d$, $w_i \sim \mathrm{N}(0,0.5)$ fixed.
- $b = -1.2$ (bias creating ~30% positive class prior).
- $\epsilon \sim \mathrm{N}(0,\sigma^2)$, $\sigma = 0.2$ (label noise).
- $\mathrm{I}(\cdot)$ indicator function.

Dataset size:

- Training: 200,000 samples.
- Validation: 10,000 samples.
- Test: 50,000 samples.

Component interpretation:

- Linear term $w^T x$: Standard linear decision boundary. Periodic term $0.6^{\mathrm{P}}\sin(x_i)$: Additive sinusoidal components in each dimension.
- Theoretical Bayes error rate: With $\sigma = 0.2$ noise, optimal classifier achieves ~92% accuracy (due to irreducible noise).
- Class balance: ~30% positive, 70% negative.
- Nonlinear complexity: Decision boundary has curvature (polynomial), oscillations (periodic), and saturation regions (indicator function).
- Visualization: 2D projections show sinusoidal curves modulated by quadratic warping.

4) Justification of Design Choices:

- Dimensionality ($d = 20$): High enough to be challenging but manageable.
- Periodic coefficient (0.6): Strong enough to significantly affect boundary but not dominate.
- Interaction term ($0.3x_1x_2$): Represents common quadratic interaction in physical systems.
- Noise level ($\sigma = 0.2$): Realistic label uncertainty without overwhelming signal.
- Large sample size: Ensures statistical reliability and reduces variance.

B. Model Architecture

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

1) Base Architecture: Feedforward neural network:

- Input layer: 20 neurons (matching data dimensionality).
- Hidden layer 1: 128 neurons → Activation → BatchNorm.
- Hidden layer 2: 64 neurons → Activation → BatchNorm. • Hidden layer 3: 32 neurons → Activation → BatchNorm.
- Hidden layer 4: 16 neurons → Activation → BatchNorm.
- Output layer: 1 neuron with sigmoid activation.

Total parameters: ∼25,000 (excluding activation parameters).

This architecture is sufficiently deep (4 hidden layers) to benefit from activation differences, not overly complex for rapid experimentation, and standardized across comparisons. *2) Integration of HyperNova++:*

- Each activation layer uses HyperNova++ with per-layer shared parameters (α,β,γ).
- Parameters initialized to $\alpha = 0.3, \beta = 0.3, \gamma = 0.4$.
- Batch normalization applied before activation (Input → BN → HyperNova++).
- Parameters receive L2 regularization with $\lambda = 0.001$.

3) Baseline Activation Functions: Comparisons against:

1. ReLU: $f(x) = \max(0,x)$ (standard baseline).
2. GELU: $f(x) = x\Phi(x)$ (common in transformers).
3. Swish: $f(x) = x\sigma(x)$ (strong empirical performer).

All baselines use identical architectures with batch normalization.

C. Training Procedure

1) Optimization:

- Optimizer: AdamW [9].
- Learning rate: 0.001.
- Betas: (0.9, 0.999).
- Weight decay: 0.01.
- Epsilon: 10^{-8}.
- Batch size: 512.
- Epochs: 100 (early stopping patience=10).
- Loss function: Binary cross-entropy.

2) Regularization: To prevent overfitting and ensure fair comparison:

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- Weight decay: 0.01 on all parameters (including HyperNova++ α,β,γ).
- Batch normalization: As described.
- Early stopping: Based on validation loss.
- No dropout: To isolate activation effects.

3) Hyperparameter Tuning: Limited tuning for each activation:

- Learning rate: {0.1,0.01,0.001,0.0001}.
- Weight decay: {0.1,0.01,0.001}.
- Corrected p-values: Bonferroni correction for multiple comparisons.
- Significance threshold: $p < 0.01$.

F. Implementation Details

1) Software Stack:

- Python 3.9.
- PyTorch 1.12 with CUDA 11.6.
- NVIDIA T4 GPUs *2 (30GB memory).
- Custom activation implementation.

```
import torch import torch.nn as nn import torch.nn.functional as F

class HyperNovaPlusPlus(nn.Module):

def __init__(self,alpha=0.3,beta=0.3,gamma=0.4,learnable=True):

super().__init__() p = torch.tensor([alpha, beta, gamma]) if learnable:

self.p = nn.Parameter(p)

else: self.register_buffer('p', p)

def forward(self, x):


return (self.p[0] * x.tanh() + self.p[1] * x.sin() + self.p[2] * F.softplus(x))
```

- Hidden layer widths: {[64,32,16,8],[128,64,32,16],[256,128,64,32]}.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Optimal configurations selected via validation performance. All final models use same architecture aside from activations.

D. Evaluation Metrics

Multiple complementary metrics:

1. Accuracy: $(TP + TN)/(TP + TN + FP + FN)$ — overall correctness.
2. F1-Score: $2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$ — balance of precision/recall.
3. ROC-AUC: Area under ROC curve — discrimination ability across thresholds.
4. Average Precision: Area under Precision-Recall curve — informative for imbalanced data.
5. Log Loss: $-\sum[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$ — calibration quality.
6. Training Time: Epochs to convergence and wall-clock time.
7. Gradient Norms: Mean and variance of gradients during training (stability measure).

All metrics computed on held-out test set (50,000 samples).

E. Experimental Runs and Statistical Significance

1) Replication Protocol: For each activation:

- Train 10 independent models with different random seeds.
- Record all metrics for each run.
- Report mean ± standard deviation across runs.
- Perform statistical significance testing.

2) Significance Tests:

Pairwise t-tests: Compare each baseline to HyperNova++ across runs.

ANOVA: Test for overall differences among all activations.

def extra_repr(self): return f'params={self.p.detach().tolist()}' TABLE I: PyTorch implementation of HyperNova++.

2) *HyperNova++ Implementation:*

3) *Training Loop:* Standard training with validation monitoring, checkpointing, and metric logging.

G. Ablation Studies

Beyond main comparisons:

1. Component Ablation: Remove individual components ($\alpha = 0, \beta = 0$, or $\gamma = 0$) to assess contributions.
2. Parameter Sharing: Compare per-layer vs. per-neuron parameterization.
3. Initialization Sensitivity: Test different initial parameter values.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

4. Architecture Scaling: Evaluate with deeper/wider networks.
5. Noise Robustness: Vary dataset noise level σ.

These provide deeper insight into HyperNova++'s behavior and limitations.

Experimental Results and Analysis

A. Main Results: Performance Comparison

TABLE II: Performance comparison across activation functions (mean ± std). $p < 0.001$.

Activation	Acc.	F1	AUC	AP	Loss	Epochs.
ReLU	0.9834	0.9840	0.9990	0.9872	0.0681	42.3
GELU	0.9808	0.9814	0.9988	0.9856	0.0723	38.7
Swish	0.9760	0.9766	0.9983	0.9821	0.0819	35.2
HyperNova++	0.9903	0.9906	0.9997	0.9928	0.0427	31.5

1) Quantitative Results: Statistical significance: All differences between HyperNova++ and baselines are significant ($p < 0.001$) via paired t-tests with Bonferroni correction. *2) Performance Interpretation:*

- Accuracy: HyperNova++ achieves 0.9903, outperforming ReLU (0.9834) by 0.7 percentage points—a 46% reduction in error rate (from 1.66% to 0.97% error).
- F1-Score: 0.9906 indicates balanced precision/recall for imbalanced data.
- ROC-AUC: Near-perfect 0.9997 shows excellent discrimination across thresholds.
- Average Precision: 0.9928 demonstrates strong performance on minority class.
- Log Loss: 0.0427 indicates well-calibrated probability estimates.
- Convergence Speed: HyperNova++ converges in 31.5 epochs vs. 42.3 for ReLU (~25% faster).

Observations:

- Faster initial convergence: HyperNova++ shows steeper early descent.
- Smoother optimization: Less oscillation in validation metrics.
- Higher final plateau: Reaches better optimum.
- Consistent improvement: Outperforms baselines throughout training.

B. Gradient Behavior Analysis

TABLE III: Gradient statistics during training (mean ± std).

Activation	Mean Grad	Grad Std	% Near-Zero

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

ReLU	0.152(21)	0.241(35)	38.2(31)
GELU	0.138(18)	0.218(29)	32.7(28)
Swish	0.145(19)	0.226(31)	29.4(25)
HyperNova++	0.123(15)	0.193(24)	21.8(19)

1) Gradient Norm Statistics: Interpretation:

- Lower mean gradient norm: Suggests more stable optimization (less gradient noise).
- Lower standard deviation: More consistent gradient flow.
- Fewer near-zero gradients: Reduced vanishing gradient issues.

HyperNova++ shows:

- Tighter distribution around moderate values.
- Fewer extreme gradients (near-zero and very large).
- More symmetrical distribution.

This supports that HyperNova++'s smooth, multicomponent design creates favorable gradient dynamics.

C. Learned Parameter Analysis

1) Final Parameter Values: Pattern:

- Early layers: Higher β (periodic emphasis) to capture sinusoidal patterns.
- Middle layers: Balanced contributions. Later layers: Higher γ (growth emphasis) for decision boundary sharpening.

TABLE IV: Average learned parameters across layers (10 runs). Parameters adapt hierarchically to emphasize different nonlinearities at different depths.

Layer Notes	α (tanh)	β (sin)	γ (Softplus)
0.42 ± 0.08	0.51 ± 0.09	0.37 ± 0.07	Emphasizes periodic
0.38 ± 0.07	0.28 ± 0.06	0.45 ± 0.08	Balanced
0.31 ± 0.05	0.19 ± 0.04	0.52 ± 0.09	Emphasizes growth
0.27 ± 0.04	0.12 ± 0.03	0.61 ± 0.10	Strong growth bias

This suggests hierarchical specialization: different nonlinearities emphasized at different depths, aligning with early layers extracting basic features and later layers combining them for classification. Observations:

- Quick initial adaptation: Parameters adjust significantly in first 5-10 epochs.
- Stabilization: Settle to stable values after ~20 epochs.
- Layer-dependent patterns: Different layers show distinct trajectories.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- Consistency across runs: Similar patterns emerge from different initializations.

D. Ablation Study Results

TABLE V: Ablation study: Impact of removing components from HyperNova++.

Configuration	Acc.	F1	AUC
Full HyperNova++	0.9903(8)	0.9906(7)	0.9997(1)
α =0 (no tanh)	0.9881(10)	0.9884(9)	0.9994(1)
β =0 (no sin)	0.9857(12)	0.9860(11)	0.9992(2)
γ =0 (no Softplus)	0.9873(11)	0.9876(10)	0.9993(1)
α =0,β =0 (only Softplus)	0.9838(13)	0.9842(12)	0.9990(2)
α =0,γ =0 (only sin)	0.9752(17)	0.9758(16)	0.9982(3)
β =0,γ =0 (only tanh)	0.9816(14)	0.9822(13)	0.9989(2)

1) Component Contribution: Findings:

1. All components contribute: Removing any reduces performance.
2. sin component most critical: Largest drop when $\beta = 0$ (accuracy -0.46%).
3. Synergistic combination: Full formulation outperforms any single component.
4. tanh+Softplus ($\beta = 0$) approximates Swish-like behavior but underperforms full HyperNova++.

TABLE VI: Parameter sharing strategies: Efficiency vs. Performance.

Parameterization	Accuracy (Mean ± Std)	Params Added	Time
Per-layer (def.)	0.9903 ± 0.0008	3 / layer	1.00
Per-channel	0.9905 ± 0.0007	3×C / layer	1.05×
Per-neuron	0.9901 ± 0.0012	3×N total	1.15
Fixed	0.9864 ± 0.0013	0	0.95

Interpretation:

- Per-layer sharing optimal: Good performance with minimal overhead.
- Per-channel slight improvement but more parameters.
- Per-neuron causes overfitting (higher variance).
- Learnability crucial: Fixed parameters underperform by

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

0.4%.

E. Scaling Experiments

TABLE VII: Network depth analysis: ReLU vs. HyperNova++.

Layers	ReLU Acc.	HyperNova++ Acc.	Rel. Impr.
2	0.9721(21)	0.9815(14)	+0.94%
4 (def.)	0.9834(12)	0.9903(08)	+0.69%
8	0.9852(10)	0.9918(06)	+0.66%
16	0.9827(13)	0.9889(09)	+0.62%

Architecture Depth: Finding: HyperNova++ provides consistent gains across depths, with slightly larger relative improvements for shallower networks.

TABLE VIII: Performance with varying training set sizes. HyperNova++ shows larger relative gains with less data, indicating better sample efficiency.

Training Samples	ReLU Accuracy	HyperNova++ Accuracy	Gap
10,000	0.9412 ± 0.0052	0.9568 ± 0.0037	+1.56%
50,000	0.9685 ± 0.0028	0.9783 ± 0.0019	+0.98%
200,000 (default)	0.9834 ± 0.0012	0.9903 ± 0.0008	+0.69%
1,000,000	0.9891 ± 0.0009	0.9932 ± 0.0005	+0.41%

Dataset Size Scaling: Finding: HyperNova++ shows larger relative gains with less data, suggesting better sample efficiency—valuable for data-scarce applications.

Noise Robustness

TABLE IX: Performance under varying label noise levels. HyperNova++'s advantage increases with noise, suggesting better robustness.

Noise σ	ReLU Accuracy	HyperNova++ Accuracy	Gap
0.0	0.9921 ± 0.0008	0.9967 ± 0.0004	+0.46%
0.2 (default)	0.9834 ± 0.0012	0.9903 ± 0.0008	+0.69%
0.5	0.9527 ± 0.0031	0.9684 ± 0.0022	+1.57%
1.0	0.8912 ± 0.0068	0.9216 ± 0.0043	+3.04%

Finding: HyperNova++'s advantage increases with noise, suggesting better robustness to label noise—possibly due to richer representation capacity fitting true signal while smoothing noise.

Computational Efficiency

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Overhead: HyperNova++ ∼2.8× slower than ReLU forward, ∼2.5× backward.

- Absolute overhead: ∼2ms per batch.
- For batch size 512, adds ∼4% to total epoch time.
- The performance gains often justify this cost.

TABLE X: Computational overhead: Time and memory analysis.

Activation	Fwd Time (ms)	Bwd Time (ms)	Mem. (MB)
ReLU	1.23(08)	1.87(12)	312
GELU	2.15(14)	3.02(18)	315
Swish	1.98(11)	2.87(16)	314
HyperNova++	3.41(22)	4.76(31)	319

Memory overhead: Minimal (¡3% increase over ReLU). Qualitative observations:

- ReLU: Piecewise linear approximations to curved boundary.
- GELU/Swish: Smoother but still limited curvature.
- HyperNova++: Accurately captures sinusoidal undulations and quadratic curvature.

Visual confirmation aligns with quantitative results.

H. Summary of Key Findings

1. Superior Performance: HyperNova++ consistently outperforms baselines across all metrics.
2. Faster Convergence: Reaches higher accuracy in fewer epochs.
3. Better Gradient Dynamics: More stable, less vanishing gradients.
4. Adaptive Specialization: Learns layer-appropriate nonlinear mixtures.
5. Robustness Advantages: Better performance with noise and limited data.
6. Moderate Computational Overhead: ∼2-3× slower than ReLU but often worthwhile.

DISCUSSION

A. Theoretical Implications

Expressive Power and Approximation Efficiency: HyperNova++'s multi-component design likely enhances approximation efficiency—the number of neurons or layers required to approximate a target function within error ϵ. While universal approximation theorems guarantee existence of approximations, efficiency depends on activation properties.

[Approximation Efficiency] Compared to networks with ReLU, GELU, or Swish, networks with HyperNova++ activations use fewer parameters or layers to achieve ϵapproximation of functions with mixed nonlinearities.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

Rationale: Each neuron in HyperNova++ has a richer function space, which eliminates the need for depth when creating simpler functions. Extending breadth-depth tradeoff analyses is necessary for formal proof [10] to adaptive activations.

Optimization Landscape Geometry: The smoothness and bounded gradients of HyperNova++ likely yield a betterconditioned optimization landscape:

- Fewer saddle points and bad local minima: Oscillatory components may prevent flat regions.
- More predictable gradient flow: Bounded Lipschitz constant ensures stable updates.
- Easier navigation to global minima: Rich curvature helps escape shallow basins.

Empirical evidence: Faster convergence and lower gradient variance support this.

3) Generalization Bounds: HyperNova++'s learnable parameters add extra complexity, whereas traditional VCdimension or Rademacher complexity bounds for neural networks usually assume fixed activations.

[Informal Generalization Bound] The generalization error for a network trained on m samples with HyperNova++ activations and parameter norms bounded by B is bounded by

$$\left(0 \sqrt{\frac{B \cdot \text{polylog}(m)}{m}} \right),$$

assuming appropriate regularization on α,β,γ.

This suggests that despite added parameters, proper regularization maintains generalization.

B. Practical Considerations

1) When to Use HyperNova++: HyperNova++ is particularly beneficial for:

1. Tasks with mixed nonlinearities: Time-series with seasonality and trends, scientific data with periodic and exponential components.
2. Data-scarce settings: Its sample efficiency helps when labeled data is limited.
3. Noisy data: Robustness to label noise is advantageous.
4. Deep architectures: Stable gradients mitigate vanishing/exploding issues.

Less beneficial for:

1. Extremely latency-sensitive applications: Overhead may be prohibitive.
2. Simple linear-separable problems: Standard activations suffice.
3. Extremely large-scale training: Computational cost may outweigh gains.

2) Integration with Modern Architectures:

- Transformers: Replace GELU with HyperNova++ in feedforward layers. The periodic component may help capture positional patterns.
- Convolutional Networks: Use per-channel parameter sharing to adapt to different feature types.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- Graph Neural Networks: The adaptive nonlinearity may help model complex node/edge relationships.
- Physics-Informed Neural Networks (PINNs): Explicit periodic component useful for oscillatory PDEs.

3) Hyperparameter Tuning Guidelines:

1. Learning rate: Similar to GELU/Swish; often 0.001 works well.
2. Weight decay: Essential for α,β,γ; use 0.01-0.001.
3. Batch normalization: Highly recommended before HyperNova++.
4. Initialization: Default ($\alpha = 0.3,\beta = 0.3,\gamma = 0.4$) robust.
5. Regularization: L2 on activation parameters prevents extreme values.

C. Limitations and Future Work

1) Current Limitations:

1. Computational Overhead: 2-3× slower than ReLU; may hinder deployment in resource-constrained settings.
2. Parameter Sensitivity: Requires careful regularization to prevent instability.
3. Theoretical Gaps: Formal approximation efficiency bounds not yet established.
4. Empirical Scope: Evaluated primarily on synthetic data; broader real-world validation needed.

2) Future Research Directions:

Theoretical Foundations:

- Establish approximation efficiency bounds relative to standard activations.
- Analyze gradient flow and convergence guarantees in deep networks.
- Study implicit regularization induced by parameter adaptation.

Architectural Innovations:

- Develop hardware-efficient implementations (approximations, quantization).
- Design specialized versions for domains (vision, language, graphs).
- Integrate with attention mechanisms and other architectural components.

Empirical Extensions:

- Large-scale evaluation on real-world benchmarks (ImageNet, WMT, molecular datasets).
- Study transfer learning and few-shot learning capabilities.
- Investigate robustness to adversarial attacks and distribution shifts.

Algorithmic Enhancements:

- Develop adaptive learning rate schedules for activation parameters.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

- Explore sparsity-inducing regularization to prune unnecessary components.
- Combine with neural architecture search to codesign architectures and activations.

CONCLUSION

This paper introduced HyperNova++, a new adaptive activation function that combines smooth unbounded growth, periodic oscillation, and bounded saturation into a single learnable formulation. We proved its smoothness, Lipschitz continuity, universal approximation capability, and advantageous gradient properties through thorough theoretical analysis.

HyperNova++ outperformed ReLU, GELU, and Swish in terms of accuracy, F1-score, ROC-AUC, and convergence speed, according to an empirical evaluation conducted on a synthetic dataset with mixed nonlinear decision boundaries.

While scaling experiments demonstrated benefits in data efficiency and noise robustness, ablation studies verified the contribution of each component.

The arrival of the HyperNova++ framework is indicative of the development of more powerful and adaptive components of neural networks, which can deal with complex tasks of the world without laying utmost emphasis on depth and special architecture of networks. The proposed framework provides a novel way of efficiently representing different nonlinear stages inside single neurons and, hence, it is likely going to result in advanced deep learning solutions.

Future work will focus on theoretical generalization bounds, large-scale real-world validation, and integration with modern architectures. The code and datasets are available at https://github.com/sourishdey2005/Hypernova.

REFERENCES

1. G. Cybenko, “Approximation by superpositions of a sigmoidal function,” Mathematics of Control, Signals and Systems, vol. 2, no. 4, pp. 303–314, 1989.
2. I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, “Maxout networks,” in International Conference on Machine Learning, 2013, pp. 1319–1327.
3. K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in
4. Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1026–1034.
5. D. Hendrycks and K. Gimpel, “Gaussian error linear units (GELUs),” arXiv preprint arXiv:1606.08415, 2016.
6. K. Hornik, “Approximation capabilities of multilayer feedforward networks,” Neural Networks, vol. 4, no. 2, pp. 251–257, 1991.
7. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in International Conference on Machine Learning, 2015, pp. 448–456.
8. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Selfnormalizing neural networks,” in Advances in Neural Information Processing Systems, 2017, pp. 971–980.
9. M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function,” Neural Networks, vol. 6, no. 6, pp. 861–867, 1993.
10. I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” arXiv preprint arXiv:1711.05101, 2017.
11. Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, “The expressive power of neural networks: A view from the width,” in Advances in Neural Information Processing Systems, 2017, pp. 6231–6239.

ISSN 2278-2540 | DOI: 10.51583/IJLTEMAS | Volume XIV, Issue XII, December 2025

12. A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in Proceedings of the 30th International Conference on Machine Learning, vol. 30, no. 1, 2013, p.3.
13. G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in Advances in Neural Information Processing Systems, 2014, pp. 2924–2932.
14. P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," arXiv preprint arXiv:1710.05941, 2017.
15. V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in Advances in Neural Information Processing Systems, 2020, pp. 7462–7473.
16. M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," in Advances in Neural Information Processing Systems, 2020, pp. 7537–7547.
17. B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," arXiv preprint arXiv:1505.00853, 2015.
18. D. Yarotsky, "Error bounds for approximations with deep ReLU networks," Neural Networks, vol. 94, pp. 103–114, 2017.
19. D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (ELUs)," arXiv preprint arXiv:1511.07289, 2015.
20. L. B. Godfrey and M. S. Gashler, "Adaptive blending units: Trainable activation functions for deep neural networks," Neurocomputing, vol. 398, pp. 1–8, 2020.