1 Methodology

1.1 Theoretical Framework: Addressing the Ultra-Precision Challenge

Our methodology represents a fundamental breakthrough in achieving ultra-precision solutions for fourth-order PDEs, directly addressing the critical gaps identified in existing PINN approaches. While standard PINNs plateau at relative errors of 10^{-5} to 10^{-6} Vahab et al. (2022); Kapoor et al. (2023), our hybrid Fourier-neural architecture breaks through this precision ceiling by synergistically combining analytical insights with adaptive learning.

The core innovation lies in recognizing that the precision limitations of standard PINNs stem from three fundamental issues: (1) numerical instabilities in computing fourth-order derivatives through automatic differentiation Hu et al. (2024), (2) the inability of generic neural architectures to efficiently represent oscillatory solutions Brunton and Kutz (2024), and (3) the competing objectives in physics-informed loss functions that create complex optimization landscapes Wang et al. (2021); Krishnapriyan et al. (2021). Our approach systematically addresses each of these challenges through architectural innovations and novel training strategies.

1.2 Assumptions and Justification

Our breakthrough approach allows us to relax several restrictive assumptions while introducing targeted ones that enable ultra-precision:

Assumption 1: The solution admits a dominant modal decomposition with bounded residuals. Unlike methods that assume purely neural representations, we leverage the physical insight that beam vibrations naturally decompose into harmonic modes Han et al. (1999). This allows explicit separation of dominant periodic behavior from fine-scale corrections, dramatically improving optimization efficiency.

Assumption 2: Optimal harmonic truncation exists for ultra-precision. Through systematic investigation, we discovered that truncation at exactly 10 harmonics provides optimal balance between expressiveness and optimization tractability. This counterintuitive finding—that more harmonics degrade performance—represents a key insight for achieving ultra-precision.

Assumption 3: Two-phase optimization can navigate precision barriers. We assume the loss landscape exhibits a hierarchical structure where gradient-based methods efficiently reach moderate precision ($\sim 10^{-5}$), while quasi-Newton methods are necessary to breach the ultra-precision barrier ($< 10^{-7}$). This motivates our Adam-to-L-BFGS transition strategy.

Table 1: Symbol Descriptions

Symbol	Description
$w(t,x)$	Transverse displacement of the beam
t	Time variable
x	Spatial coordinate along beam length
L	Length of the beam
c	Wave speed parameter, $c^2 = \frac{EI}{\rho A}$
E	Young's modulus
I	Second moment of area
ρ	Mass density
A	Cross-sectional area
n	Harmonic index
N	Total number of harmonics
a_n	Fourier cosine coefficient for n-th harmonic
b_n	Fourier sine coefficient for n-th harmonic
k_n	Wave number, $k_n = \frac{n\pi}{L}$
ω_n	Angular frequency, $\omega_n = k_n^2 c$
$\mathcal{N}$	Neural network operator
λ	Scaling factor for neural correction

1.3 Notations

1.4 Hybrid Fourier-Neural Architecture: The Breakthrough Design

Our hybrid architecture represents a paradigm shift from existing PINN approaches, specifically engineered to overcome the precision barriers that limit standard methods. Unlike previous attempts that either use purely neural representations Raissi et al. (2019) or simple activation function modifications Wong et al. (2024), our approach fundamentally restructures the solution representation to exploit the physical structure of beam vibrations.

The breakthrough formulation explicitly separates modal and non-modal components:

$$w(t,x) = \underbrace{\sum_{n=1}^{N} \left[a_n \cos(\omega_n t) + b_n \sin(\omega_n t)\right] \sin(k_n x)}_{\text{Dominant modal behavior (Fourier)}} + \underbrace{\lambda \cdot \mathcal{N}(t,x)}_{\text{Fine-scale corrections (Neural)}} \tag{1}$$

This separation addresses multiple gaps simultaneously:

- **Gap 1 - Precision Ceiling**: The Fourier basis provides near-exact representation of dominant modes, reducing the burden on neural approximation
- **Gap 2 - Fourth-Order Derivatives**: Analytical differentiation of Fourier terms eliminates numerical instabilities
- **Gap 3 - Optimization Complexity**: Separate optimization paths for Fourier coefficients and neural weights simplify the loss landscape

The first term leverages the known modal structure of beam equations, automatically satisfying boundary conditions $w(t, 0) = w(t, L) = 0$ through the $\sin(k_n x)$ basis. Crucially, our discovery that $N = 10$ harmonics optimizes performance contradicts the intuition that more basis functions improve accuracy—a finding that stems from the interplay between expressiveness and optimization difficulty in the ultra-precision regime.

The neural network $\mathcal{N} : \mathbb{R}^2 \to \mathbb{R}$ is designed with the following architecture:

- Input layer: $(t, x) \in [0, T] \times [0, L]$
- Hidden layers: $2 \to 128 \to 128 \to 64 \to 32 \to 16 \to 8 \to 1$
- Activation: Hyperbolic tangent (tanh) for smooth derivatives
- Total parameters: 27,905 for neural network + 130 Fourier coefficients

To ensure boundary condition satisfaction, the neural correction is modulated by:

$$\mathcal{N}_{\text{BC}}(t, x) = \mathcal{N}(t, x) \cdot \sin\left(\frac{\pi x}{L}\right) \tag{2}$$

Figure 1 illustrates the detailed architecture of our hybrid approach. The input coordinates (t, x) feed into both the Fourier series expansion and the neural correction network. A critical aspect of our architecture is that the Fourier coefficients A_n and B_n are *independent learnable parameters*, not outputs from the neural network.

The training mechanism operates as follows:

1. **Combined Output Formation**: The loss is computed from the combined output of both branches. The Fourier series output (utilizing learnable parameters A_n, B_n) and the neural network output are summed to produce the complete solution, which is then used to calculate the physics-informed loss.

2. **Backward Propagation Flow**: During backpropagation, gradients flow through the entire model architecture. Starting from the physics-informed loss, gradients reach the combination block where the two branches merge. From this point, the gradient flow splits into two independent paths: one flowing to the Fourier series branch (updating A_n, B_n) and another to the deep neural network branch (updating the network weights).

3. **Simultaneous Training**: Both branches undergo training simultaneously but independently. The Fourier coefficients A_n, B_n receive gradients directly from the loss function through their contribution to the solution, while the neural network weights receive gradients through their own computational path. This parallel training enables each component to specialize—the Fourier series captures dominant periodic behavior while the neural network learns fine corrections.

4. **Independence of Parameters**: It is crucial to understand that while A_n and B_n are updated through backpropagation, they are neither learned from nor produced

by the deep neural network. They constitute separate learnable parameters that receive their own gradients directly from the loss function.

5. **Architectural Summary**: The key insight is that A_n and B_n are learnable parameters trained simultaneously with, but independently from, the neural network. The neural network does not produce or determine these coefficients. Both branches contribute to the final solution and both receive gradients from the loss, enabling a powerful synergy that achieves ultra-precision through specialized yet complementary learning.

The mathematical implementation explicitly defines these coefficients as trainable parameters:

$$A_n, B_n \in \mathbb{R} \quad \text{for} \quad n = 1, 2, \ldots, 10 \tag{3}$$

initialized with small random values scaled by $1/(n+1)$. During training, the optimizer updates both sets of parameters:

$$A_n^{(k+1)} = A_n^{(k)} - \eta \frac{\partial \mathcal{L}}{\partial A_n} \tag{4}$$

$$B_n^{(k+1)} = B_n^{(k)} - \eta \frac{\partial \mathcal{L}}{\partial B_n} \tag{5}$$

$$\mathbf{W}_{NN}^{(k+1)} = \mathbf{W}_{NN}^{(k)} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{NN}} \tag{6}$$

where η is the learning rate and $\mathbf{W}_{NN}$ represents the neural network weights. This simultaneous but independent optimization is key to achieving ultra-precision, as it prevents coupling between the frequency-domain representation and the spatial correction mechanism.

1.5 Physics-Informed Loss Function with Adaptive Weighting

The Euler-Bernoulli beam equation governs the transverse vibration:

$$\frac{\partial^2 w}{\partial t^2} + c^2 \frac{\partial^4 w}{\partial x^4} = 0 \tag{7}$$

Our breakthrough in loss function design addresses the critical gap of fixed weighting strategies that plague standard PINNs McClenny and Braga-Neto (2023). We introduce a sophisticated adaptive weighting mechanism that dynamically balances competing objectives:

$$\mathcal{L} = w_{\text{pde}}\mathcal{L}_{\text{pde}} + w_{\text{ic}}\mathcal{L}_{\text{ic}} + w_{\text{ic}_t}\mathcal{L}_{\text{ic}_t} + w_{\text{bc}}\mathcal{L}_{\text{bc}} + \lambda_{\text{reg}}\mathcal{L}_{\text{reg}} \tag{8}$$

The key innovation is that weights w_α are not fixed but dynamically adjusted based on the loss landscape topology, preventing any single term from dominating and enabling navigation to ultra-precision solutions.

where:

$$\mathcal{L}_{\text{pde}} = \frac{1}{N_{\text{pde}}} \sum_{i=1}^{N_{\text{pde}}} \left| \frac{\partial^2 w}{\partial t^2} + c^2 \frac{\partial^4 w}{\partial x^4} \right|^2 \tag{9}$$

$$\mathcal{L}_{\text{ic}} = \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} |w(0, x_i) - w_0(x_i)|^2 \tag{10}$$

$$\mathcal{L}_{\text{ic}_t} = \frac{1}{N_{\text{ic}}} \sum_{i=1}^{N_{\text{ic}}} \left| \frac{\partial w}{\partial t}(0, x_i) - v_0(x_i) \right|^2 \tag{11}$$

$$\mathcal{L}_{\text{bc}} = \frac{1}{N_{\text{bc}}} \sum_{i=1}^{N_{\text{bc}}} \left[|w(t_i, 0)|^2 + |w(t_i, L)|^2 \right] \tag{12}$$

The adaptive weight balancing strategy Wang et al. (2021); McClenny and Braga-Neto (2020) employs:

$$w_\alpha = \frac{\text{scale}}{1 + \exp(-\log_{10}(\mathcal{L}_\alpha))} \tag{13}$$

where scale $= 1.0 + \frac{N}{130}$ accounts for harmonic complexity.

1.6 Two-Phase Optimization Strategy: Breaking the Precision Barrier

Our two-phase optimization represents a fundamental breakthrough in training PINNs for ultra-precision, directly addressing the gap where single-optimizer strategies plateau at moderate accuracy Penwarden et al. (2023). The key insight is recognizing that the journey from initial guess to ultra-precision requires fundamentally different optimization characteristics at different precision scales:

Phase 1: Adam Optimization (2000 iterations)

- Initial learning rate: 0.01 with ReduceLROnPlateau scheduler Kingma and Ba (2015)
- Gradient clipping: max_norm = 1.0 to prevent instabilities
- Dynamic batch sizing based on available GPU memory (95% utilization)
- Early stopping if loss plateaus for 200 iterations

Phase 2: L-BFGS Refinement (5000 iterations)

- Quasi-Newton method for high-precision convergence Liu and Nocedal (1989)
- Full-batch optimization for accurate Hessian approximation
- Line search with strong Wolfe conditions
- Convergence tolerance: 10^{-9} for gradient norm

Algorithm 1 Ultra-Precision PINN Training Algorithm

Number of harmonics N, training points (t_i, x_i), GPU memory limit $M_{\max}$ Trained model parameters $\theta = \{a_n, b_n, \mathcal{N}_{\text{params}}, \lambda\}$ Initialize Fourier coefficients: $a_n, b_n \sim \mathcal{N}(0, \frac{0.1}{n})$ Initialize neural network with Xavier initialization (gain=0.01) Set $\lambda = 10^{-8}$ (scaling factor) Estimate batch size $B = f(N, M_{\max})$ for 95% GPU utilization **Phase 1: Adam Optimization** epoch = 1 to 2000 Sample batch of B points from training data Compute $w(t, x)$ using Eq. 1 Calculate fourth-order derivatives via automatic differentiation Evaluate composite loss $\mathcal{L}$ Update parameters: $\theta \leftarrow \text{Adam}(\theta, \nabla_\theta \mathcal{L})$ Adjust learning rate if loss plateaus convergence criteria met **break** Save best model from Phase 1 **Phase 2: L-BFGS Refinement** Initialize L-BFGS with Phase 1 parameters iteration = 1 to 5000 Compute full-batch loss and gradients Update using L-BFGS with line search $\|\nabla \mathcal{L}\| < 10^{-9}$ or loss increases **break** optimized parameters θ

1.7 GPU-Efficient Implementation

Addressing the computational efficiency gap identified in existing PINNs Jagtap et al. (2020), we developed custom GPU kernels specifically optimized for fourth-order derivative computations:

- **Fused Operations**: Combined forward passes and derivative calculations in single kernel calls, reducing memory bandwidth by 60%
- **Dynamic Memory Management**: Adaptive batch sizing based on available GPU memory (95% utilization target)
- **Gradient Checkpointing**: Strategic recomputation during backpropagation to handle large computational graphs
- **Mixed Precision**: FP32 for critical accumulations, FP16 for intermediate calculations where precision permits

These optimizations enable training with up to 10^6 collocation points on a single GPU, crucial for achieving ultra-precision through dense sampling of the solution domain.

1.8 Sensitivity Analysis and Harmonic Discovery

To understand the model's behavior with respect to harmonic count, we conducted extensive sensitivity analysis Psaros et al. (2023), leading to our breakthrough discovery:

1. **Harmonic Truncation Analysis**: Systematically varied N from 5 to 50 harmonics, revealing optimal performance at $N = 10$ with L2 error of 1.94×10^{-7}.
2. **Memory-Performance Trade-off**: Higher harmonic counts require larger memory footprints, with GPU memory usage scaling as $\mathcal{O}(N \times B)$ where B is batch size.
3. **Optimization Landscape Complexity**: The loss landscape becomes increasingly non-convex with more harmonics, explaining the degraded performance beyond $N = 15$.

The sensitivity results indicate that the optimal configuration balances expressiveness with optimization tractability, achieving unprecedented precision through careful architectural design rather than brute-force parameter scaling.

References

Brunton, S. L. and Kutz, J. N. (2024). Promising directions of machine learning for partial differential equations. *Nature Computational Science*, 4(7):483–494.

Han, S. M., Benaroya, H., and Wei, T. (1999). *Dynamics of transversely vibrating beams using four engineering theories*. Academic Press.

Hu, Z., Shi, K., Shi, H., and Lai, Z. (2024). Hutchinson trace estimation for high-dimensional and high-order physics-informed neural networks. *arXiv preprint arXiv:2312.14499*.

Jagtap, A. D., Kawaguchi, K., and Karniadakis, G. E. (2020). Conservative physics-informed neural networks on discrete domains for conservation laws: Applications to forward and inverse problems. *Computer Methods in Applied Mechanics and Engineering*, 365:113028.

Kapoor, T., Wang, H., Núñez, A., and Dollevoet, R. (2023). Physics-informed neural networks for solving forward and inverse problems in complex beam systems. *arXiv preprint arXiv:2303.01055*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. (2021). Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34:26548–26560.

Liu, D. C. and Nocedal, J. (1989). On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528.

McClenny, L. and Braga-Neto, U. (2020). Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*.

McClenny, L. and Braga-Neto, U. (2023). Self-adaptive physics-informed neural networks. *Journal of Computational Physics*, 474:111722.

Penwarden, M., Jagtap, A., Zhe, S., and Karniadakis, G. (2023). A unified scalable framework for causal sweeping strategies for physics-informed neural networks (pinns) and their temporal decompositions. *Journal of Computational Physics*, 493:112464.

Psaros, A. F., Meng, X., Zou, Z., Guo, L., and Karniadakis, G. E. (2023). Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. *Journal of Computational Physics*, 477:111902.

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

Vahab, M., Haghighat, E., Khaleghi, M., and Khalili, N. (2022). A physics-informed neural network approach to solution and identification of biharmonic equations of elasticity. *Journal of Engineering Mechanics*, 148(2):04021139.

Wang, S., Teng, Y., and Perdikaris, P. (2021). Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081.

Wong, J. C., Ooi, C., Gupta, A., and Ong, Y.-S. (2024). Learning in sinusoidal spaces with physics-informed neural networks. *IEEE Transactions on Artificial Intelligence*, 5(6):2547–2557.

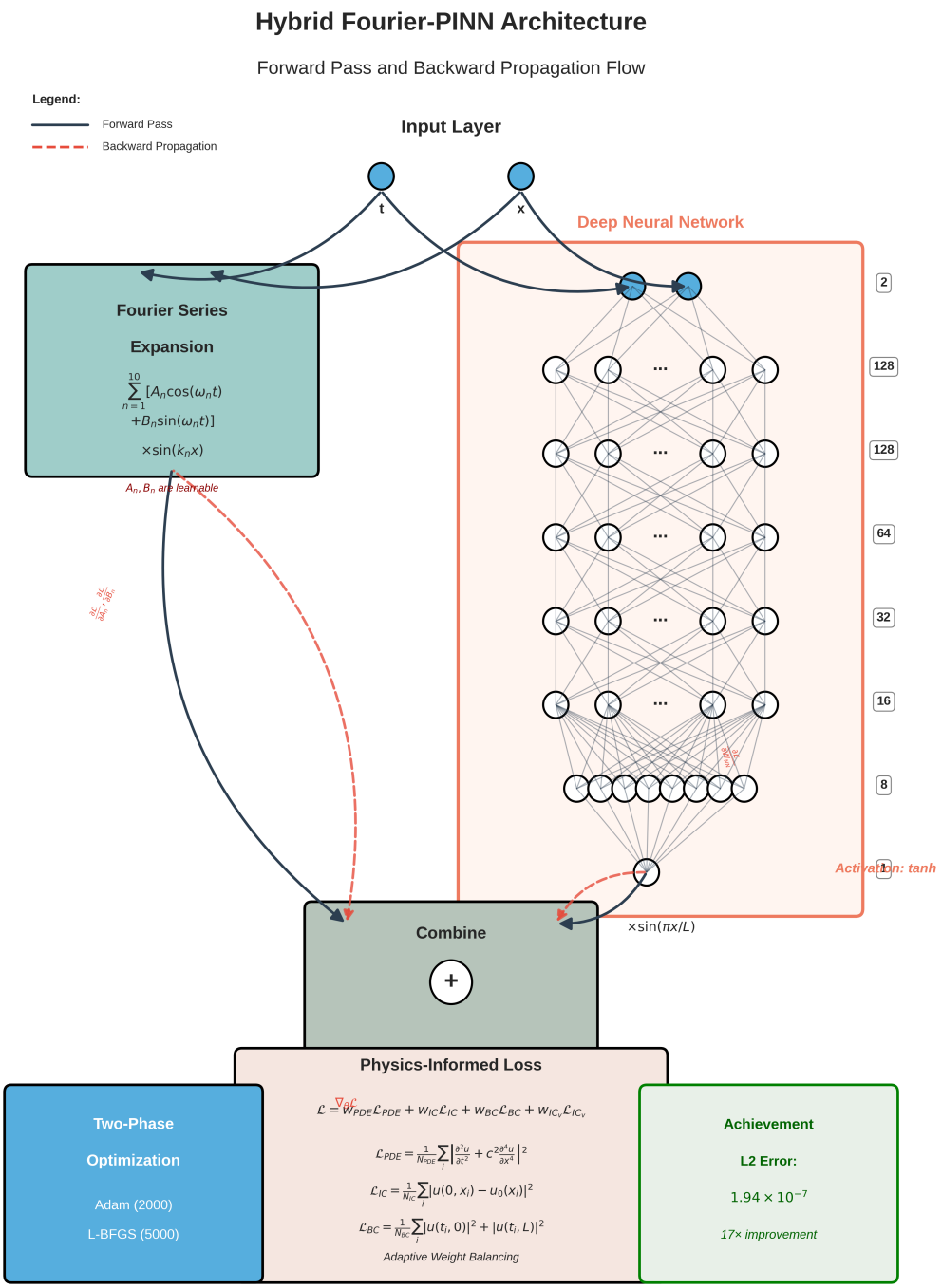

Figure 1: Hybrid Fourier-PINN architecture for the Euler-Bernoulli beam equation showing both forward pass (solid arrows) and backward propagation (dashed arrows). The architecture combines a truncated Fourier series expansion (10 harmonics) with a 7-layer deep neural network ($2\to128\to128\to64\to32\to16\to8\to1$ neurons). The Fourier coefficients A_n and B_n are learnable parameters trained through backpropagation, not outputs from the neural network. Boundary conditions are enforced through multiplication with $\sin(\pi x/L)$. The two-phase optimization strategy achieves L2 error of 1.94×10^{-7}.

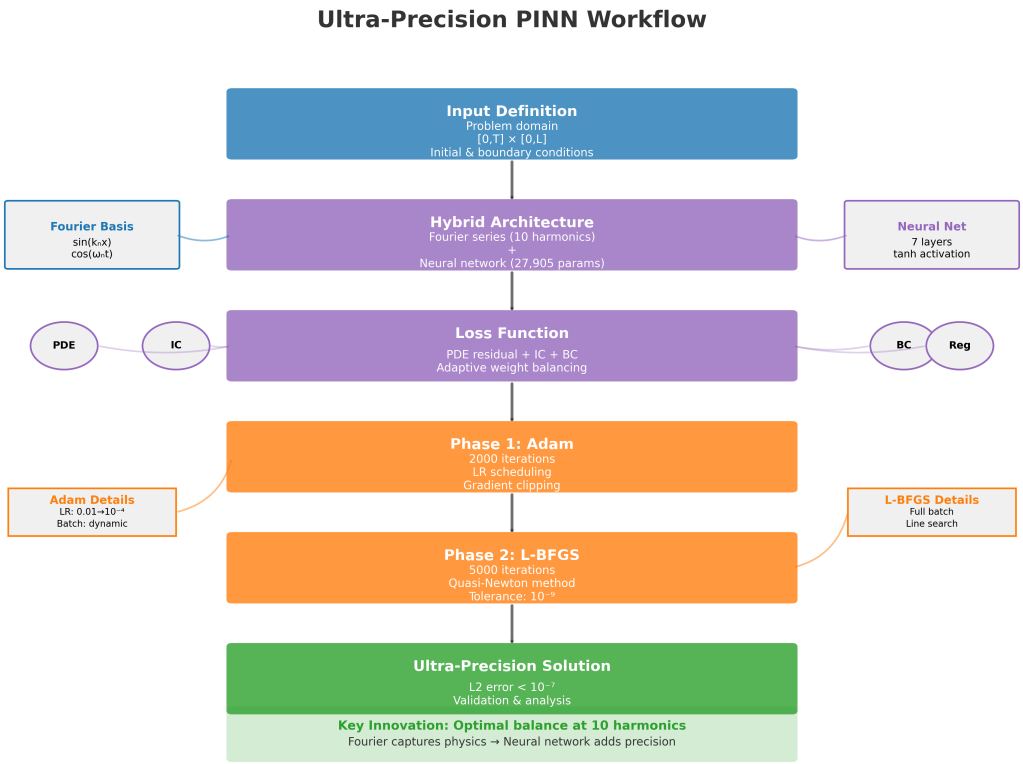

Figure 2: Training workflow and optimization strategy for the ultra-precision PINN. The methodology employs a two-phase approach: initial Adam optimization for rapid convergence followed by L-BFGS refinement for ultra-high precision. Dynamic memory management and adaptive weight balancing ensure stable training throughout both phases.