From Kolmogorov to Cauchy: Shallow XNet Surpasses KANs

Xin Li[*]
College of Computer Science and Technology
Dongguan University of Technology, China
Institute for Advanced Research
Great Bay University, Dongguan, Guangdong, China
`xinli2023@u.northwestern.edu`

Xiaotao Zheng[*]
Center for Financial Engineering
Soochow University, Suzhou, China
`20234013002@stu.suda.edu.cn`

Zhihong Xia[†]
Institute for Advanced Research
Great Bay University, Dongguan, Guangdong, China
Department of Mathematics
Northwestern University, Evanston, IL, USA
`xia@math.northwestern.edu`

Abstract

We study a shallow variant of XNet, a neural architecture whose activation functions are derived from the Cauchy integral formula. While prior work focused on deep variants, we show that even a single-layer XNet exhibits near-exponential approximation rates—exceeding the polynomial bounds of MLPs and spline-based networks such as Kolmogorov–Arnold Networks (KANs).

Empirically, XNet reduces approximation error by over 600× on discontinuous functions, achieves up to 20,000× lower residuals in physics-informed PDEs, and improves policy accuracy and sample efficiency in PPO-based reinforcement learning—while maintaining comparable or better computational efficiency than KAN baselines.

These results demonstrate that expressive approximation can stem from principled activation design rather than depth alone, offering a compact, theoretically grounded alternative for function approximation, scientific computing, and control.

1 Introduction

As deep learning models grow increasingly large and resource-intensive, there is renewed interest in shallow architectures that combine theoretical rigor with practical efficiency. In particular, scientific applications such as PDE solving, symbolic regression, and control require models that are not only expressive but also interpretable, sample-efficient, and robust to irregularities.

Traditional universal approximators like shallow MLPs [5, 10] suffer from slow convergence [1, 7], especially in high dimensions. Recent efforts such as Kolmogorov–Arnold Networks (KANs) [19] improve expressivity via spline-based activations, but still rely on multi-layer composition and struggle with discontinuities due to their smooth basis structure.

[*]Equal contribution
[†]Corresponding author

In this work, we study a principled *single-layer* variant of **XNet**, a neural architecture built on the *Cauchy activation function* introduced in [16]:

$$\phi(x) = \frac{\lambda_1 x + \lambda_2}{x^2 + d^2}, \quad \lambda_1, \lambda_2, d \in \mathbb{R}.$$

While [16] proposed Cauchy activations as a plug-and-play component for general deep architectures, we focus on the *shallow, single-layer* setting and provide the first theoretical analysis of its approximation rates. We prove that even a single-layer XNet achieves arbitrarily high-order convergence $O(N^{-r})$ for any $r > 0$ on real-analytic functions, and demonstrate its effectiveness across function approximation, PDE solving, and reinforcement learning tasks.

These rational functions—derived from the Cauchy integral formula—yield smooth, localized, and trainable basis elements, enabling XNet to approximate irregular, high-dimensional, and even discontinuous functions with far fewer neurons and no depth.

Main theoretical result. We prove that XNet achieves *arbitrarily fast* convergence $O(N^{-p})$ for any $p > 0$ on smooth target functions. In contrast, classical ReLU networks are limited by the curse of dimensionality with rate $O(N^{-2r/d})$ (for target smoothness r and input dimension d), and spline-based KANs achieve $O(N^{-k})$ constrained by the fixed spline degree k. XNet's analytic Cauchy basis functions overcome both limitations by enabling tunable, high-order approximation in a single layer. See Section 3.2 for the formal result and Appendix A.5 for the proof.

Figure 1 illustrates this convergence behavior: as the number of neurons N increases, XNet maintains a significantly steeper error decay rate compared to ReLU and KAN.

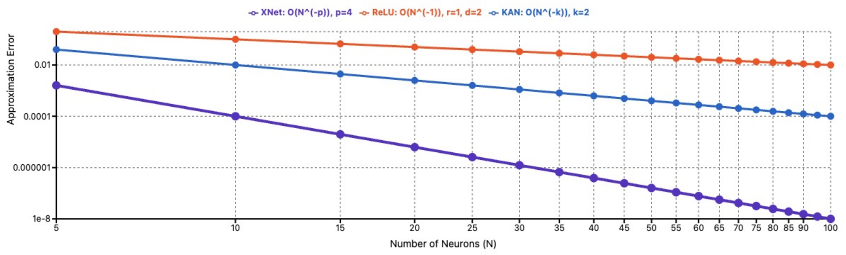


Figure 1: Comparison of approximation rates. XNet achieves $O(N^{-p})$ with $p = 4$, outperforming ReLU ($O(N^{-2r/d})$ with $r = 1$, $d = 2$) and KAN ($O(N^{-k})$ with $k = 2$). Log-log plot shows approximation error versus neuron count N.

Empirical results: Across three domains, XNet outperforms strong baselines:

- **Function approximation:** over **600×** lower MSE on discontinuous targets.
- **PDE solving:** over **10,000×** lower error than deep MLPs in PINN settings. (see Appendix A.1 for computational analysis)
- **Reinforcement learning:** faster convergence and higher rewards under PPO.

These findings challenge the depth-centric paradigm, showing that *activation design alone* can enable expressive, compact, and theoretically grounded models—often with computational efficiency comparable to or better than multi-layer alternatives.

Why this matters now. As foundation models encounter scaling bottlenecks, there is renewed interest in shallow networks that combine interpretability, efficiency, and provable expressivity. XNet exemplifies this direction by achieving strong performance without depth or overparameterization.

[16] introduced the Cauchy activation and established its universal approximation properties. Building on this foundation, we establish new convergence rates in dual function spaces (Theorem 3.2; proof in Appendix A.5), proving $O(N^{-r})$ for any $r > 0$ on real-analytic functions—the first such result for shallow XNet. While our analysis focuses on single-layer networks, [16] demonstrated the modularity of Cauchy activations in deeper architectures (multi-layer MLPs, CNNs, ResNets) on MNIST and CIFAR-10.

Together, these results challenge the depth-centric paradigm, showing that principled activation design alone can yield compact, expressive models for scientific and symbolic AI.

2 Related Work

Universal Approximation and Expressive Efficiency. Shallow MLPs with non-polynomial activations are universal approximators [5, 10], but their approximation efficiency degrades in high dimensions [1, 26]. Recent work improves expressivity through spline [19], rational [3], and interpolant-based [23] activations. Parametric activations [2, 15] offer adaptability, but often increase complexity or lack convergence guarantees. XNet addresses this gap by connecting activation design with provable, high-order approximation in a shallow architecture.

Structured Activation Networks. KANs [19] approximate functions via learnable B-splines along edges, enabling efficient approximation with compositional depth. However, their reliance on spline smoothness and fixed grid resolution limits adaptability to discontinuities or noise. Recent applications span time-series [25], RL [11], and generative modeling [9]. In contrast, XNet provides analytic, localized responses with no depth or grid assumptions.

Rational and Complex-Valued Models. Rational activations [4, 3] have been used to improve approximation power via learnable Padé-like functions. However, these approaches are primarily heuristic and often lack theoretical guarantees beyond empirical smooth function fitting. Our work draws from complex analysis—specifically the Cauchy integral formula—to derive rational activations with localized and tunable responses. This analytic foundation enables high-order approximation in both classical and dual function spaces, distinguishing XNet from prior rational models.

Scientific Machine Learning and Operator Learning. PINNs [22] and neural operators [17, 14] enable learning in scientific domains, but often rely on deep networks, spectral bias, or task-specific assumptions. Spectral solvers [8] and low-rank architectures [27] improve convergence, but require expert tuning. XNet offers a shallow alternative with strong theoretical grounding and minimal tuning—achieving fast convergence across physics-informed and control tasks.

3 Theoretical Foundations of XNet

3.1 Limitations of Spline-Based KANs

Kolmogorov–Arnold Networks (KANs) [19] implement the classical Kolmogorov superposition theorem [13] via spline-based edge activations:

$$f(x) = \sum_q \Phi_q \left(\sum_p \phi_{q,p}(x_p) \right), \quad \phi(x) = w_b b(x) + w_s \sum_i c_i B_i(x),$$

where $B_i(x)$ are B-spline basis functions. While KANs offer adaptive local representations and $O(N^{-k})$ convergence for spline degree k, they suffer from poor generalization near discontinuities, require multi-layer structures, and incur high computational costs due to spline interpolation and grid dependencies.

3.2 Cauchy Basis Architecture (XNet)

This section analyzes a shallow variant of **XNet**, a neural architecture built on *Cauchy basis activations* derived from the Cauchy integral formula:

$$\phi(x) = \frac{\lambda_1 x + \lambda_2}{x^2 + d^2}, \quad \text{with trainable } \lambda_1, \lambda_2, d.$$

This yields localized, tunable, and analytically smooth responses. Below, we present the theoretical foundations supporting its expressivity.

(1) Approximation Efficiency. XNet achieves arbitrarily fast convergence $O(N^{-p})$ on smooth target functions, where the convergence exponent $p > 0$ depends on the analyticity (i.e., smoothness) of the target. This surpasses the $O(N^{-2r/d})$ rate of ReLU MLPs [1, 26] and the $O(N^{-k})$ rate of spline-based KANs [19], where r is the Sobolev smoothness and d is the input dimension.

See Appendix A.2 for a detailed derivation.

(2) Localization and Noise Adaptivity. Cauchy activations exhibit rational decay $O(1/x)$, yielding naturally localized basis functions:

$$\phi_i(x) = \frac{\lambda_{1,i}(x - \mu_i) + \lambda_{2,i}}{(x - \mu_i)^2 + d_i^2}.$$

By adjusting the parameter d_i, neurons can adapt their receptive field: small d_i captures sharp discontinuities, while large d_i smooths over noisy regions. Unlike B-splines, which require predefined grid partitions, XNet offers neuron-wise adaptivity for heterogeneous signals.

See Appendix A.3 for analytical expressions and derivative properties.

(3) Comparison with B-Spline KANs. We contrast the analytical and computational properties of XNet (Cauchy basis) and Kolmogorov–Arnold Networks (B-spline basis) in Table 1.

Table 1: Comparison of XNet vs. B-spline KANs

Aspect	**Cauchy Basis (XNet)**	**B-spline (KAN)**
Approximation Rate	$O(N^{-p})$, arbitrary $p > 0$	$O(N^{-k})$, limited by degree k
Basis Adaptivity	Continuous, localized, trainable	Grid-constrained, piecewise
Matrix Structure	Dense (condition number $O(N)$)	Sparse (condition number $O(1)$)
Derivative Computation	Closed-form, differentiable w.r.t. d^2	Piecewise, non-smooth
Architectural Cost	Single-layer, low parameter count	Multi-layer, higher cost

This comparison highlights XNet's key advantages over KANs: global analytical structure, tunable locality, and full differentiability—without reliance on grid resolution. For further details on matrix conditioning and gradient properties, see Appendix A.4.

(4) Cauchy Approximation Theorem.

Theorem 3.1 (Th.1 in [16]). *Let $f(z^1, \ldots, z^d)$ be analytic on open $U \subset \mathbb{C}^d$. Then for any $\varepsilon > 0$, there exists a finite sum:*

$$\sup_{z \in U} \left| f(z) - \sum_{k=1}^{N} \frac{\lambda_k}{\prod_{j=1}^{d} (\xi_k^j - z^j)} \right| < \varepsilon, \quad \textit{with} \quad \varepsilon = O(N^{-p}).$$

This theorem shows that Cauchy activation functions can approximate analytic functions with arbitrarily high order. The specific form

$$\phi(x) = \frac{\lambda_1 x + \lambda_2}{x^2 + d^2}$$

was introduced in [16] as a simplified instantiation.

While the above result applies to partial fractions, it does not directly yield rates for XNet. Our Theorem 3.2 provides the first constructive $O(N^{-r})$ rate for single-layer XNet on real-analytic functions $f \in C^\omega(K)$ over compact domains $K \subset \mathbb{R}^d$.

(5) New Result: High-Order Approximation of XNet. We now prove that a one-hidden-layer neural network using Cauchy activations inherits the same approximation rate.

Theorem 3.2 (High-Order Approximation of XNet)**.** *Let $K \subset \mathbb{R}^d$ be compact, and let $f \in C^\omega(K)$ be real analytic. Let $\Phi = \left\{ \frac{\lambda_1 x + \lambda_2}{x^2+d^2} \right\}$ be the Cauchy activation family. Then for any $r > 0$, there exists a one-hidden-layer XNet function $f_N \in span(\Phi)$ such that*

$$\|f - f_N\|_{C^0(K)} = O(N^{-r}).$$

This is a new result: while Theorem 2 in [16] shows that XNet is universal, it does not quantify convergence rate. Our theorem gives an explicit $O(N^{-r})$ bound for real-analytic f.

Sketch. (1) Use the Cauchy approximation theorem to approximate any 1D analytic function with rate $O(N^{-r})$; (2) Represent multivariate polynomials using $(w^\top x)^k$, reducing to the 1D case; (3) Approximate f by polynomials, then apply Step (2). See Appendix A.5 for details. □

Remark 3.3. While our theorem assumes $f \in C^\omega(K)$ (real-analytic), empirical results (Section 4.1.1) show XNet performs well even on discontinuous functions like the Heaviside step. This aligns with classical results on rational approximation to non-smooth functions (e.g., [20, 24] for $|x|^\alpha$), which demonstrate near-exponential rates. Extending our theoretical framework to broader function classes is an important direction for future work.

(6) Neuron Efficiency at Arbitrary Approximation Orders. We show that one-hidden-layer XNet achieves convergence rates of $O(N^{-r})$ for any desired order $r > 0$. To achieve an approximation error of ϵ, the number of required neurons satisfies:

$$N = O\left(\epsilon^{-1/r}\right), \quad \forall r > 0.$$

This tunable convergence order distinguishes XNet from classical architectures: ReLU MLPs are fixed at $r = 1/d$ [1], while spline-based KANs are limited by spline degree k, i.e., $r \leq k$.

In contrast, XNet leverages the analytic structure of the Cauchy basis to achieve flexible, localized, and high-precision approximation with fewer neurons. This result is particularly beneficial for smooth, high-dimensional, or irregular targets. A complete derivation of these scaling laws is provided in Appendix A.6.

Table 2: Theoretical comparison of approximation rate and neuron complexity.

Architecture	**Approx. Rate**	**Neurons for Error ϵ**	**Adjustable Order r?**	**Depth Requirement**
ReLU MLP	$O(N^{-1/d})$	$O(\epsilon^{-d})$	✗ (fixed)	Deep
KAN (Spline)	$O(N^{-k})$	$O(\epsilon^{-1/k})$	✗ (limited by k)	Moderate
XNet (Ours)	$O(N^{-r})$	$O(\epsilon^{-1/r})$	✓ (Any $r > 0$)	**Single-layer**

Remark 3.4. The ability to *tune the approximation order arbitrarily* via r is a direct consequence of the Cauchy approximation theorem [16], which guarantees exponential convergence for analytic functions.

This theoretical framework not only guarantees approximation efficiency, but also explains XNet's empirical strength in modeling irregular and high-dimensional scientific data.

4 Experimental Setup and Results

We evaluate XNet across three domains—function approximation, PDE solving, and reinforcement learning—selected to assess different capabilities of neural architectures. These tasks respectively test expressivity under discontinuities, accuracy in modeling physical operators, and generalization in sequential decision-making.

This diversity highlights XNet's key strengths: high-order approximation, compact design, and efficient training. In all settings, we compare against MLP and KAN baselines under matched parameter budgets and consistent training protocols.

4.1 Function Approximation Experiments

We evaluate the networks on four categories of functions with increasing complexity: a low-dimensional discontinuous function, low-dimensional special functions, high-dimensional functions, and a function with noise. The experimental comparison between XNet, B-spline, and KAN indicates that XNet provides enhanced approximation capabilities. Except for the first function example and the final task, all other examples are from the referenced article [19], with KAN settings matching those from the original experiments. This ensures a fair comparison, demonstrating that XNet exhibits strong approximation capabilities across various benchmarks.

4.1.1 Heaviside step function approximation

Dataset and Implementation Details. We evaluate discontinuous function approximation capabilities using the Heaviside step function:

$$f(x) = \begin{cases} 1, & x > 0, \\ 0, & x \leq 0. \end{cases} \tag{1}$$

An XNet architecture [1, 64, 1] and a KAN configuration [1, 1] with 200 grids were implemented. Both models were trained using the Adam optimizer [12] under identical conditions.

Table 3: Heaviside function approximation comparison

Model	Architecture	MSE	RMSE	MAE
XNet	[1,64,1]	**8.99e-08**	**3.00e-04**	**1.91e-04**
ReLU (shallow)	[1,64,1]	2.05e-03	4.53e-02	8.82e-03
ReLU (deep)	[1,64,64,1]	6.81e-05	8.25e-03	3.76e-04
KAN	[1,1] (200 grids)	5.98e-04	2.45e-02	3.03e-03

Overall Performance. As shown in Figure 4, both B-Spline and KAN exhibit "overshoot," resulting in local oscillations at discontinuities. While adjusting the grid can mitigate this phenomenon, it introduces complexity in parameter tuning (see Table 9, Appendix B.2). In contrast, XNet demonstrates superior performance, providing smoother transitions at discontinuities. In terms of fitting accuracy in these regions, XNet outperforms KAN, with an MSE approximately 1000-fold smaller (8.99e-08 versus 5.98e-04) as shown in Table 3. The mathematical foundation for XNet's advantages in discontinuous functions is provided in Section 3.2.

4.1.2 Approximation with special functions

Dataset and Implementation Details. We collect several special functions common in math and physics, summarized in Table 4. For neural network architectures, we implemented consistent parameter configurations following [19]. MLPs are configured with fixed widths of 5 or 100, with depths varying across $\{2, 3, 4, 5, 6\}$. KANs maintain a fixed width of 5 with depths similarly swept through $\{2, 3, 4, 5, 6\}$. XNet was constructed as a single-layer architecture with either 500 or 5000 basis functions.

Table 4: Special functions benchmark comparison with fixed XNet shape and extended test RMSEs.

Name	scipy.special API	Minimal KAN shape	XNet shape	Best KAN test RMSE	MLP test RMSE	XNet test RMSE
Jacobian elliptic functions	`ellipj(x, y)`	[2,2,1]	[2,500,1]	$1.33e{-}04$	$6.48e{-}04$	$\mathbf{4.03e{-}05}$
Incomplete elliptic integral of the first kind	`ellipkinc(x, y)`	[2,2,1,1]	[2,500,1]	$1.24e{-}04$	$5.52e{-}04$	$\mathbf{6.76e{-}05}$
Incomplete elliptic integral of the second kind	`ellipeinc(x, y)`	[2,2,1,1]	[2,500,1]	$8.26e{-}05$	$3.04e{-}04$	$\mathbf{5.60e{-}05}$
Bessel function of the first kind	`jv(x, y)`	[2,3,1,1,1]	[2,500,1]	$1.64e{-}03$	$5.52e{-}03$	$\mathbf{2.39e{-}04}$
Bessel function of the second kind	`yv(x, y)`	[2,2,2,1]	[2,5000,1]	$1.49e{-}05$	$3.45e{-}04$	$\mathbf{1.12e{-}05}$
Modified Bessel function of the first kind	`iv(x, y)`	[2,4,3,2,1,1]	[2,500,1]	$9.28e{-}03$	$1.07e{-}02$	$\mathbf{6.39e{-}05}$
Associated Legendre function ($m = 0$)	`lpmv(0, x, y)`	[2,2,1]	[2,500,1]	$5.25e{-}05$	$1.74e{-}02$	$\mathbf{4.01e{-}05}$
Associated Legendre function ($m = 1$)	`lpmv(1, x, y)`	[2,4,1]	[2,500,1]	$6.90e{-}04$	$1.50e{-}03$	$\mathbf{2.71e{-}04}$
Associated Legendre function ($m = 2$)	`lpmv(2, x, y)`	[2,3,2,1]	[2,500,1]	$2.26e{-}04$	$9.43e{-}04$	$\mathbf{7.44e{-}05}$
spherical harmonics ($m = 1, n = 1$)	`sph_harm(1,1,x,y)`	[2,3,2,1]	[2,500,1]	$1.22e{-}04$	$6.70e{-}04$	$\mathbf{2.28e{-}05}$
spherical harmonics ($m = 1, n = 2$)	`sph_harm(1,2,x,y)`	[2,2,1,1]	[2,500,1]	$1.50e{-}05$	$1.84e{-}03$	$\mathbf{7.78e{-}06}$
spherical harmonics ($m = 2, n = 2$)	`sph_harm(2,2,x,y)`	[2,2,3,2,1]	[2,500,1]	$9.45e{-}05$	$6.21e{-}04$	$\mathbf{2.43e{-}05}$

Overall Performance. As presented in Table 4, experiments demonstrate that XNet achieves superior approximation accuracy on the special functions. Particularly noteworthy are XNet's results on complex functions such as spherical harmonics, where it achieves an RMSE of 7.78×10^{-6} for $(m = 1, n = 2)$ compared to KAN's 1.50×10^{-5} and MLP's 1.84×10^{-3}. For Bessel functions, XNet similarly shows considerable improvement, with an RMSE of 2.39×10^{-4} for the first kind versus 1.64×10^{-3} and 5.52×10^{-3} for KAN and MLP respectively. These findings suggest that XNet's single-layer architecture effectively approximates special functions with high accuracy.

4.1.3 Approximation with high-dimensional functions

Dataset and Implementation Details. Following the benchmarks established in [19], we evaluated two challenging functions: a 4-dimensional function $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2 + x_2^2)\right) + x_3x_4\right)\right)$ and a 100-dimensional function $\exp(\frac{1}{100}\sum_{i=1}^{100}\sin^2(\frac{\pi x_i}{2}))$. Each experiment utilized 8000 training points and 1000 test points. For the 4D function, KAN was configured as $[4, 4, 2, 1]$ and XNet was implemented with 5000 basis functions. For the 100D function, KAN was structured as $[100, 1, 1]$ with XNet maintaining the same configuration. XNet was optimized using the Adam algorithm, while KAN is initialized with $G = 3$ grid points and trained with the LBFGS algorithm [18], progressively increasing grid density through $G = 3, 5, 10, 20, 50$ every 200 steps.

Table 5: Performance comparison of XNet and KAN on 4D and 100D test functions

Function	Model	MSE	RMSE	MAE	Time (s)
$e^{\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)}$	XNet [2, 5000, 1]	2.31e-06	1.52e-03	8.39e-04	78.18
	KAN [4,2,2,1]	2.62e-03	5.11e-02	3.63e-02	143.10
$e^{\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)}$	XNet [2, 5000, 1]	6.85e-04	2.62e-02	2.09e-02	158.69
	KAN [100, 1, 1]	6.59e-03	8.12e-02	6.46e-02	556.50

Overall Performance. The results presented in Table 5 indicate that XNet outperforms KAN across both dimensionalities. For the 4D function, XNet achieves an MSE three orders of magnitude lower than KAN while requiring approximately half the computation time. In the challenging 100D case, XNet maintains nearly an order of magnitude better MSE while executing 3.5× faster than KAN. Notably, XNet's computational efficiency scales significantly better as dimensionality increases, while achieving consistently higher accuracy across all evaluation metrics.

4.1.4 Functions with noise

Dataset and Implementation Details. To assess function approximation capabilities in noisy environments, we examined a dynamic system governed by:

$$x_5^i = 0.1x_0^ix_1^i + 0.5\sin(x_2^ix_3^i) + \sin(x_4^i) + \mu^i,$$

where $i = 1, 2, ..., n$, with state transitions:

$$x_0^i = x_1^{i-1}, x_1^i = x_2^{i-1}, x_2^i = x_3^{i-1}, x_4^i = x_5^{i-1}.$$

Initial conditions $x_0^0, x_1^0, x_2^0, x_3^0, x_4^0$ were sampled from $[0, 0.2]$, with Gaussian noise $\mu^i \sim N(0, \sigma^2)$ at three distinct levels: none ($\sigma = 0$), moderate ($\sigma = 0.05$), and high ($\sigma = 0.1$). We generated 300 sequential points, training models on the first 80% (noise-corrupted data) and evaluating prediction performance on the remaining 20% (noise-free data). This experimental design evaluates each model's capability to extract underlying patterns despite noise in the training data. For this experiment, we compared KAN [5, 64, 1] against XNet [5, 20, 1], both optimized using the Adam optimizer.

Overall Performance. Table 6 summarizes the comparative results. XNet outperforms KAN across all metrics, particularly in noisy environments. In the noise-free scenario, XNet achieves an MSE of 4.7099×10^{-7}, approximately 28× lower than that of KAN (1.3209×10^{-5}). As noise increases, XNet maintains its advantage, with 3.7× better MSE at moderate noise and 14.5× better at high noise. Figures 8 and 9 provide visual evidence of XNet's enhanced robustness to noise. Furthermore, XNet achieves these performance improvements with fewer parameters and reduced training times. These results indicate that XNet may be more suitable for real-world noisy datasets, while the KAN model exhibits characteristics consistent with overfitting.

Table 6: Performance comparison of KAN [5, 64, 1] and XNet [5, 20, 1] under different noise levels.

Noise Level	Model	MSE	RMSE	MAE	Time (s)
0	**KAN [5, 64, 1]**	1.3209e-05	3.6344e-03	4.8311e-03	17.8863
	XNet [5, 20, 1]	4.7099e-07	6.8629e-04	5.2943e-04	9.6502
0.05	**KAN [5, 64, 1]**	1.6784e-03	4.0968e-02	3.1549e-02	17.9535
	XNet [5, 20, 1]	4.5109e-04	2.1239e-02	1.5797e-02	9.4391
0.1	**KAN [5, 64, 1]**	9.4653e-03	9.7290e-02	7.8063e-02	17.8863
	XNet [5, 20, 1]	6.5517e-04	2.5596e-02	2.0233e-02	9.0296

Given its performance in function approximation tasks, both in terms of computational efficiency and accuracy, experimental results suggest that XNet provides an efficient neural network architecture with effective approximation capabilities. Building on this, in the following subsection, we apply PINN, KAN, and XNet to approximate the value function of the Heat and Poisson equations.

4.2 PDE Solving Capability

Experimental Setup and Implementation Details. We selected the Heat equation as a benchmark PDE problem:

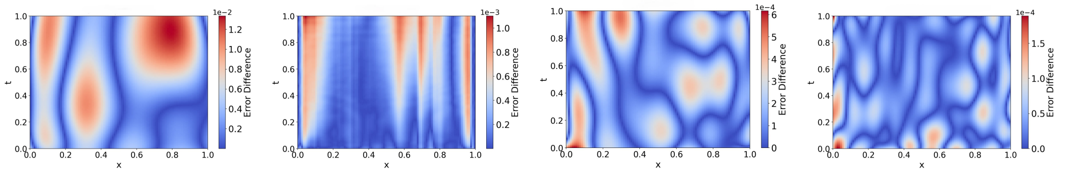
$$\frac{\partial u}{\partial t} = \nu \frac{\partial^2 u}{\partial x^2}, \quad (x,t) \in [0,1] \times [0,1], u(x,0) = \sin(\pi x), \quad x \in [0,1], u(0,t) = u(1,t) = 0, \tag{2}$$

with known analytical solution $u(x,t) = e^{-\nu\pi^2 t}\sin(\pi x)$. Following the physics-informed neural networks (PINNs) framework, we defined the loss function as

$$\text{loss} = \alpha\, \text{loss}_i + \text{loss}_o = \frac{1}{n_o}\sum_{i=1}^{n_o} \left|u^\theta(z_i) - u(z_i)\right|^2 + \alpha \frac{1}{n_i}\sum_{i=1}^{n_i} \left|u_t^\theta(z_i) - \nu u_{xx}^\theta(z_i)\right|^2, \tag{3}$$

where $n_i = 2500$ interior points and $n_o = 150$ boundary points were sampled uniformly, with $\alpha = 0.1$ serving as a weighting coefficient to balance the loss components.

We evaluate four network architectures within the PINN framework: a two-layer MLP $[2, 20, 20, 1]$, a single-layer KAN $[2, 10, 1]$, and two XNet configurations with varying widths $[2, 20, 1]$ and $[2, 200, 1]$.



(a) MLP [2, 20, 20, 1] (b) KAN [2, 10, 1] (c) XNet [2, 20, 1] (d) XNet [2, 200, 1]

Figure 2: Performance comparison of different neural network architectures

Table 7: Comparison of XNet and KAN on the Heat equation.

Metric	MSE	RMSE	MAE	Time (s)
MLP [2,20,20,1]	2.4536e-05	4.9534e-03	3.8323e-03	43.8
XNet [2,20,1]	3.8936e-08	1.9732e-04	1.5602e-04	43.5
KAN [2,10,1]	1.5106e-07	3.8866e-04	2.9661e-04	254.6
XNet [2,200,1]	3.6867e-09	6.0718e-05	5.0027e-05	108.3

Overall Performance. Table 7 quantitatively compares the performance of MLP, XNet, and KAN architectures on the heat equation task. Across all evaluated metrics, XNet consistently achieves superior accuracy and computational efficiency. The [2,200,1] XNet configuration achieves the lowest error, with an MSE of 3.69×10^{-9}—approximately two orders of magnitude lower than

KAN (1.51×10^{-7})—while also requiring less than half the training time (108.3s vs. 254.6s). Even with a more compact architecture ([2,20,1]), XNet outperforms KAN, showing a threefold lower MSE and completing training more than five times faster. In comparison, the standard MLP ([2,20,20,1]) produces higher errors across all metrics, highlighting the limitations of conventional architectures for PDE approximation. These results demonstrate XNet's effectiveness in capturing the underlying structure of PDE solutions, suggesting its promise as a compact and efficient model reduction framework.

Additionally, we evaluated physics-informed machine learning models based on MLP, KAN, and XNet architectures for solving the 2D Poisson equation (detailed results in Appendix B.6). The XNet [2,200,1] configuration outperforms KAN [2,10,1] in both accuracy and computational efficiency, while achieving approximately $20,000\times$ better precision compared to the standard PINN [2,20,20,1] architecture.

4.3 Reinforcement Learning Applications

Experimental Setup and Implementation Details. We evaluated XNet, KAN, and MLP architectures as function approximators within the Proximal Policy Optimization (PPO) framework on two continuous control tasks from the DeepMind Control Suite: *HalfCheetah-v4* and *Swimmer-v4*.

Proximal Policy Optimization (PPO) optimizes the stochastic policy π_θ by employing a clipped surrogate objective function to ensure stable policy updates:

$$L_{\text{PPO}}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right], \tag{4}$$

where $r_t(\theta)$ represents the policy ratio and ϵ is a clipping parameter (configured at 0.2). This objective function is combined with a value function loss and entropy regularization to balance exploration and exploitation.

Table 8 summarizes the network configurations used in our experiments. The MLP baseline uses two hidden layers with 64 units each for both actor and critic networks. The KAN implementation employs spline functions with order $k = 2$ and grid size $g = 3$, chosen based on preliminary work [11]. The XNet configuration utilizes 64 basis functions to maintain parameter count parity with the MLP architecture for fair comparison. All models were trained using the Adam optimizer, implementing identical hyperparameters across architectures to isolate the effects of the function approximator choice.

Table 8: Network configurations for reinforcement learning tasks.

Model	Configuration
MLP	Two hidden layers with 64 units each
KAN	Order $k = 2$, grid size $g = 3$
XNet	64 basis functions

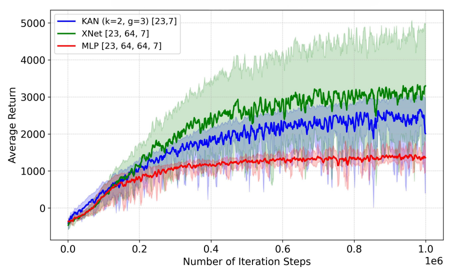
Overall Performance. Figure 3 summarize the results of the three models on the two tasks. XNet consistently outperforms MLP and KAN in both environments, achieving the highest rewards and faster convergence. In *HalfCheetah-v4* environment, XNet improves the reward by 64% compared to KAN and 142% compared to MLP. In *Swimmer-v4* environment, XNet achieves reward improvements of 47% over MLP and 28% over KAN.

Theoretical Analysis of XNet's Advantage. XNet's performance in reinforcement learning stems from its Cauchy kernel activation, which provides three key benefits:

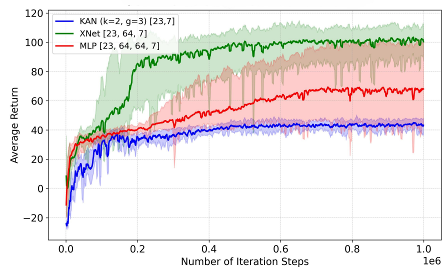
1. **Localized response with fast decay:** Cauchy activations enable sharper adaptation to localized patterns in policy and value functions—particularly effective in continuous control tasks with abrupt transitions.

2. **High-order approximation capability:** As shown in Theorem, XNet supports arbitrarily high-order convergence for analytic targets, yielding more precise representations than MLPs or KANs.

3. **Reduced parameter complexity:** XNet achieves strong performance with fewer parameters due to its expressive basis functions, leading to faster convergence in reinforcement learning.



(a) HalfCheetah-v4



(b) Swimmer-v4

Figure 3: Reward comparison for PPO training across environments. XNet significantly outperforms both KAN and MLP in final performance after 1M training steps: In HalfCheetah-v4, XNet achieves 3298.52, compared to KAN (2010.52) and MLP (1358.49). In Swimmer-v4, XNet reaches 100.38, while KAN and MLP attain 43.25 and 68.08 respectively.

These properties align well with the demands of policy gradient methods like PPO, where accurate approximation of both policy $\pi_\theta(a|s)$ and value function $V(s)$ is crucial. Figure 3 empirically confirms these advantages.

5 Conclusion and Outlook

Our results suggest that neural expressivity need not come from depth alone. XNet, built on principled analytic activation functions, achieves high-order approximation with only a single layer—offering both theoretical rigor and empirical power.

This design unlocks a new perspective for compact and interpretable architectures, particularly in scientific and symbolic domains. Its ability to approximate discontinuous functions, solve PDEs efficiently, and improve sample efficiency in RL agents makes it a promising candidate for hybrid symbolic-numeric solvers and real-time control systems.

In future work, we plan to integrate XNet into multi-scale or Transformer-style models, extend it to generative and multi-modal learning, and explore its role in operator learning and symbolic program induction.

References

[1] Andrew R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[2] G. Bingham and R. Miikkulainen. Discovering parametric activation functions. *Neural Networks*, 148:48–65, 2022.

[3] Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[4] Daniele Caldarola, Tommaso Furlanello, Silvio Savarese, and Matteo Bruni. Rational relu: Towards exact function approximation with rational activation functions. In *International Conference on Learning Representations (ICLR)*, 2021.

[5] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

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

[7] Ronen Eldan and Ohad Shamir. The power of depth for feedforward neural networks. In *Proceedings of the 29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 907–940. PMLR, 2016.

[8] Zhewei Fang, Yujun Shen, Xiao Yang, Xiangyu Xu, Xintao Wang, Bo Dai, and Chen Change Loy. Fourier neural operator with learnable activation functions. *arXiv preprint arXiv:2302.11661*, 2023.

[9] Remi Genet and Hugo Inzirillo. Tkan: Temporal kolmogorov-arnold networks. *arXiv preprint arXiv:2405.07344*, 2024.

[10] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[11] Victor A Kich, Jair A Bottega, Raul Steinmetz, Ricardo B Grando, Ayano Yorozu, and Akihisa Ohya. Kolmogorov-arnold networks for online reinforcement learning. In *2024 24th International Conference on Control, Automation and Systems (ICCAS)*, pages 958–963. IEEE, 2024.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[13] A.N. Kolmogorov. On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables. *Dokl. Akad. Nauk*, 108(2), 1956.

[14] Nikola B. Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Anima Anandkumar, and Andrew M. Stuart. Neural operator: Learning maps between function spaces. *Journal of Machine Learning Research*, 23:1–97, 2022.

[15] Vladimír Kunc and Jiří Kléma. Three decades of activations: A comprehensive survey of 400 activation functions for neural networks. *arXiv*, 2024.

[16] Xin Li, Zhihong Xia, and Hongkun Zhang. Cauchy activation function and xnet. *Neural Networks*, 188:107375, 2025.

[17] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

[18] Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

[19] Siyuan Liu, Ruochen Zhang, Huizhuo Qiu, Xuan Zhao, Jiaxu Zhang, Zhen Yang, Zhiqiang Zhang, Qingmin Wang, Jiang Liu, Ting Liu, et al. Kan: Kolmogorov-arnold networks. In *International Conference on Learning Representations (ICLR)*, 2025.

[20] D. J. Newman. Rational approximation to $|x|$. *Michigan Mathematical Journal*, 11(1):11–14, 1964.

[21] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[22] Maziar Raissi, Paris Perdikaris, and George E. Karniadakis. 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, 2019.

[23] David Rolnick, Alexander Amini, Aditi Raghunathan, Pulkit Agrawal, and Thomas Poggio. Reverse-engineering the universal approximation theorem. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[24] Herbert R. Stahl. Best uniform rational approximation of x^α on $[0, 1]$. *Acta Mathematica*, 190(2):241–306, 2003.

[25] Kunpeng Xu, Lifei Chen, and Shengrui Wang. Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability. *arXiv preprint arXiv:2406.02496*, 2024.

[26] Dmitry Yarotsky. Error bounds for approximations with deep ReLU networks. *Neural Networks*, 94:103–114, 2017.

[27] Kaichen Zhang, Lingxiao Li, and George Em Karniadakis. Low-rank physics-informed neural networks (lr-pinns). *Journal of Computational Physics*, 463:111218, 2022.

A Theoretical Foundations

A.1 Computational Efficiency Analysis

FLOP Comparison: The Cauchy activation $\phi(x) = \frac{\lambda_1 x+\lambda_2}{x^2+d^2}$ introduces 2-3× more FLOPs per activation compared to ReLU, but remains significantly lighter than spline-based KANs.

Time-to-Solution: Despite higher per-activation cost, XNet achieves faster convergence. For example, on the Heat equation (Table 7), XNet [2,200,1] reaches target accuracy in 108s vs KAN's 254s, demonstrating that expressivity gains outweigh computational overhead.

Scalability: For Transformer architectures, Cauchy maintains $O(n)$ complexity where n is sequence length, adding only a constant factor increase in FLOPs.

A.2 Approximation Efficiency

We recall the classical approximation rate result from Cauchy kernel expansions. If $f \in C^p(K)$, then for any $p > 0$, there exists a rational function f_N composed of N Cauchy basis functions such that:

$$\|f - f_N\|_\infty \leq C(p)N^{-p},$$

where $C(p)$ depends on the smoothness of f and the domain geometry.

To achieve error at most ϵ, we require:

$$N = O\left(\epsilon^{-1/p}\right).$$

In contrast, ReLU networks yield $O(N^{-1/d})$ for d-dimensional f, and KANs offer $O(N^{-k})$ bounded by spline degree k. Thus, XNet provides tunable convergence rates with fewer neurons for smooth targets.

A.3 Localization and Adaptivity

The activation function:

$$\phi(x) = \frac{\lambda_1 x + \lambda_2}{x^2 + d^2}$$

satisfies:

$$\lim_{|x|\to\infty} \phi(x) = 0, \quad \phi'(x) = \frac{\lambda_1(x^2 + d^2) - 2x(\lambda_1 x + \lambda_2)}{(x^2 + d^2)^2}.$$

For localized representation centered at μ_i, we define:

$$\phi_i(x) = \frac{\lambda_{1,i}(x - \mu_i) + \lambda_{2,i}}{(x - \mu_i)^2 + d_i^2}.$$

This enables each neuron to adapt to regions of discontinuity or local variation. Unlike B-splines, XNet does not require explicit partitioning or fixed grid support.

A.4 Matrix Conditioning and Derivative Properties

Matrix Conditioning: The kernel matrix A with entries $A_{ij} = \phi_i(x_j)$ is dense. Using Gershgorin's theorem:

$$|\lambda - A_{ii}| \leq \sum_{j\neq i} |A_{ij}|,$$

the condition number scales as $\kappa(A) = O(N)$ in general, potentially requiring preconditioning for large N. B-spline matrices are sparse and banded with $\kappa(A) = O(1)$.

Derivative Computation: For efficient optimization, we compute:

$$\frac{d^n}{d(d^2)^n}\left(\frac{1}{x^2+d^2}\right) = (-1)^n \frac{(n-1)!}{(x^2+d^2)^{n+1}}.$$

Cauchy activations are fully differentiable with respect to d^2, enabling second-order or adaptive methods.

A.5 Proof of High-Order Approximation

Theorem 3.2 (High-Order Approximation of XNet)**.** *Let $K \subset \mathbb{R}^d$ be compact, and let $f \in C^\omega(K)$ be real analytic. Let $\Phi = \left\{\frac{\lambda_1 x+\lambda_2}{x^2+d^2}\right\}$ be the Cauchy activation family. Then for any $r > 0$, there exists a one-hidden-layer XNet function $f_N \in span(\Phi)$ such that*

$$\|f - f_N\|_{C^0(K)} = O(N^{-r}).$$

Proof. We now establish the high-order approximation capability of XNet in the uniform norm $C^0(K)$ for any compact set $K \subset \mathbb{R}^d$. The proof is divided into three steps: we begin in the one-dimensional real setting, then extend to high-dimensional polynomials using a classical lifting theorem, and finally conclude by composing the approximations.

Step 1: One-dimensional high-order approximation via the Cauchy approximation theorem.

Let $I = [-1, 1] \subset \mathbb{R}$, and let $f \in C^\omega(I)$ be any real-analytic function on the interval. By the one-dimensional Cauchy approximation theorem (a special case of Theorem 1 in [16]), for any $\varepsilon > 0$, $\left\|f(x) - \sum_{k=1}^{M} \frac{\lambda_k}{\xi_k - x}\right\|_{C^0(I)} < \varepsilon$, i.e., we have $g_M(x) = \sum_{k=1}^{M} \phi_k(x)$:

$$\|f - g_M\|_{C^0(I)} = O(M^{-r}) \quad \text{for any } r > 0.$$

Step 2: Multivariate polynomial approximation by Cauchy activations (XNet)

Let $K \subset \mathbb{R}^d$ be a compact set, and let $\mathbb{P}_n(\mathbb{R}^d)$ denote the set of all real multivariate polynomials of total degree at most n.

We refer to the General Approximation Theorem in [16]. If the one-dimensional function $\phi(x)$ can approximate all monomials x^k with arbitrarily high-order accuracy in the C^0 norm, then for any fixed $w \in \mathbb{R}^d$, the functions $\phi(w^\top x + b)$ can approximate $(w^\top x)^k$ with the same rate. Since any multivariate monomial $x^\alpha = \prod_{j=1}^{d} (x^j)^{\alpha_j}$ can be written as a linear combination of such functions (as shown in [16]), it follows that Cauchy activations of the form $\phi(w^\top x + b)$ can approximate x^α with arbitrarily high-order convergence in $C^0(K)$.

Step 3: High-order approximation of analytic functions via XNet

Let $f \in C^\omega(K)$ be a real analytic function on a compact domain $K \subset \mathbb{R}^d$. By classical approximation theory (e.g., Bernstein, Mhaskar), there exists a sequence of multivariate polynomials $\{p_n(x)\} \subset \mathbb{P}_n(\mathbb{R}^d)$ such that

$$\|f - p_n\|_{C^0(K)} = O(n^{-r}) \quad \text{for any } r > 0.$$

From Step 2, each polynomial p_n can be approximated by a function g_M in the XNet class:

$$g_M(x) = \sum_{i=1}^{M} a_i \cdot \phi(w_i^\top x + b_i),$$

with

$$\|p_n - g_M\|_{C^0(K)} = O(M^{-s}) \quad \text{for any } s > 0.$$

Applying the triangle inequality, we obtain:

$$\|f - g_M\|_{C^0(K)} \le \|f - p_n\|_{C^0(K)} + \|p_n - g_M\|_{C^0(K)} = O(n^{-r}) + O(M^{-s}).$$

By choosing $M = n$, we deduce:

$$\|f - g_M\|_{C^0(K)} = O(M^{-k}) \quad \text{for any } k > 0.$$

Conclusion:

Let $\phi(x) = \frac{\lambda_1 x + \lambda_2}{x^2 + d^2}$, and define the single-layer neural network class (XNet) as

$$\mathcal{X}_M := \left\{ x \mapsto \sum_{i=1}^{M} a_i \cdot \phi(w_i^\top x + b_i) \right\}.$$

Then for any compact set $K \subset \mathbb{R}^d$, and any function $f \in C^\omega(K)$, there exists $g_M \in \mathcal{X}_M$ such that

$$\|f - g_M\|_{C^0(K)} = O(M^{-k}) \quad \text{for any } k > 0.$$

That is, XNet achieves arbitrarily high-order approximation accuracy for real analytic functions in the uniform norm.

□

A.6 Neuron Efficiency under Arbitrary Approximation Orders

The table in Section 3.2 summarizes how different architectures compare in terms of neurons required to reach a desired approximation error ϵ. Here, we provide the derivation behind the scaling laws.

XNet. From Theorem 3.2, we have

$$\|f - f_N\| \leq CN^{-r} \Rightarrow N = O(\epsilon^{-1/r}).$$

This result holds for arbitrary $r > 0$, assuming $f \in C^k(K)$ and sufficient smoothness.

ReLU MLP. For classical ReLU networks [1], the best known approximation rate is:

$$\|f - f_N\| \leq CN^{-1/d} \Rightarrow N = O(\epsilon^{-d}),$$

which reflects the curse of dimensionality.

KAN. For B-spline-based KANs [19], convergence is limited by the spline degree k:

$$\|f - f_N\| \leq CN^{-k} \Rightarrow N = O(\epsilon^{-1/k}),$$

and the degree k is typically small (e.g., $k = 3$), fixed by construction.

Summary. These derivations support the theoretical table in Section 3.2, confirming that XNet allows flexible approximation with significantly fewer neurons under high-order settings.

B Experimental Details

B.1 Implementation Environment

The numerical experiments presented below were performed in Python using the Tensorflow-CPU processor on a Dell computer equipped with a 3.00 Gigahertz (GHz) Intel Core i9-13900KF. When detailing grids ans k for KAN models, we always use values provided by respective authors (Kan).

B.2 One-Dimensional Discontinuous FUNCTION APPROXIMATION

Figure 4 provides visualizations of approximation results for the Heaviside function, highlighting the characteristic ”overshoot” phenomenon in B-spline and KAN methods near the discontinuity, which XNet successfully avoids.

Table 9 presents an extensive comparison of various KAN configurations, showing that even with deeper architectures ([1,3,1]), KAN cannot match XNet’s approximation accuracy for discontinuous functions. The B-spline results further demonstrate that increasing the spline degree k from 3 to 10 yields only marginal improvements, while XNet provides orders of magnitude better approximation with its single-layer architecture.

Table 9: KAN reference

	[1,1]KAN			[1,3,1]KAN		
k,G	MSE	RMSE	MAE	MSE	RMSE	MAE
k=3, G=3	2.20E-02	1.48E-01	9.89E-02	3.50E-04	1.87E-02	5.56E-03
k=3, G=10	1.22E-02	1.10E-01	5.91E-02	1.84E-04	1.36E-02	2.54E-03
k=3, G=50	2.44E-03	4.94E-02	1.22E-02	4.28E-05	6.55E-03	2.71E-03
k=3, G=200	5.98E-04	2.45E-02	3.03E-03	3.79E-04	1.95E-02	1.24E-02

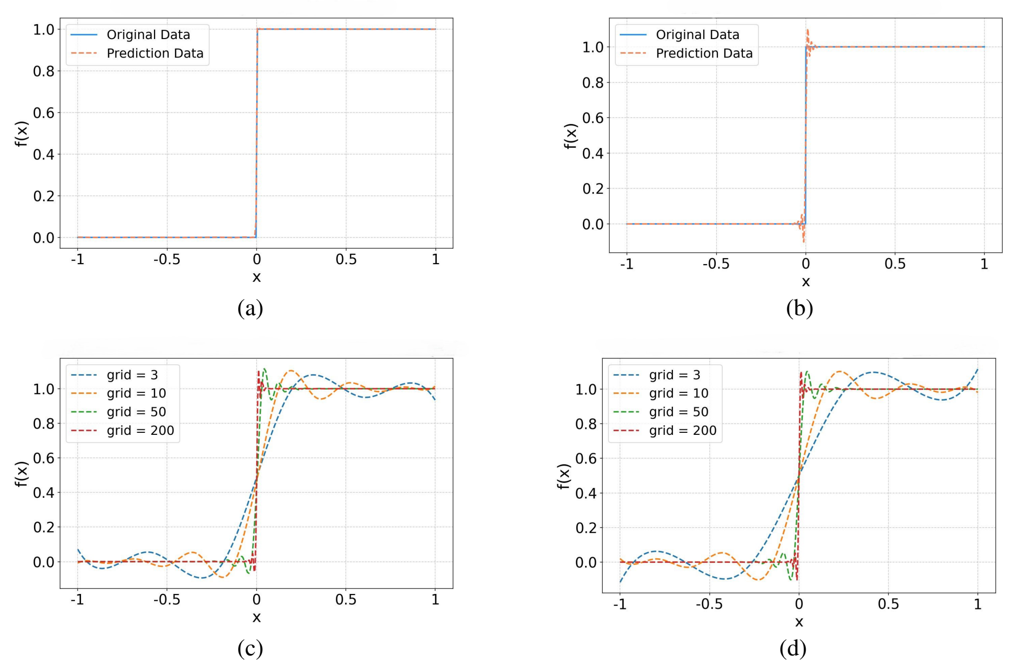


Figure 4: Heaviside step function approximation comparison: (a) XNet, with 64 basis functions. (b) KAN [1,1], with $k = 3$, grid = 200. (c) B-Spline, with $k = 3$. (d) KAN [1,1], with $k = 3$.

Table 10: B-Spline Performance metrics comparison for different G and K values. reference

B-Spline			
k, G	**MSE**	**RMSE**	**MAE**
k=50, G=200	5.8477e-01	7.6470e-01	6.1076e-01
k=3, G=10	9.2871e-03	9.6369e-02	4.7923e-02
k=3, G=50	2.3252e-03	4.8221e-02	1.2255e-02
k=10, G=50	1.9881e-03	4.4588e-02	1.0879e-02
k=3, G=200	1.1252e-03	3.3544e-02	4.4737e-03
k=10, G=200	1.1029e-03	3.3210e-02	5.1904e-03

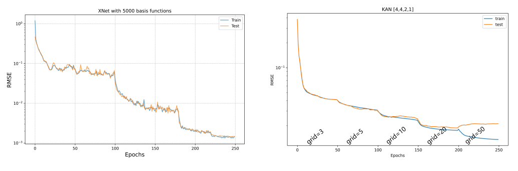


Figure 5: $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)\right)$

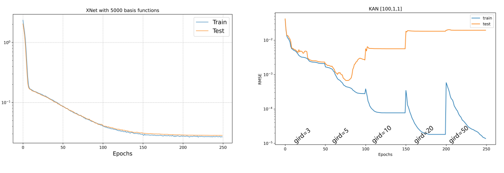
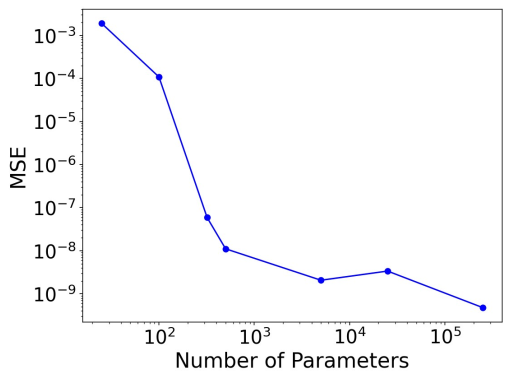


Figure 6: $\exp(\frac{1}{100}\sum_{i=1}^{100}\sin^2(\frac{\pi x_i}{2}))$

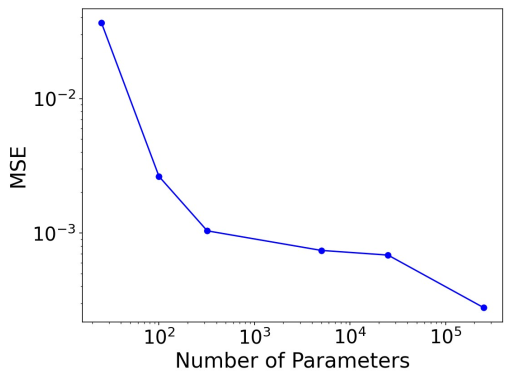
B.3 High-Dimensional Continuous FUNCTION APPROXIMATION

Figure 5 and 6 show the training loss curves for both XNet and KAN on the 4D and 100D test functions.

Figure 7 shows XNet's performance with varying numbers of Cauchy basis functions. Performance improves consistently with increasing width, confirming the theoretical scaling laws.



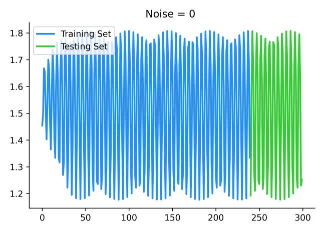
(a)



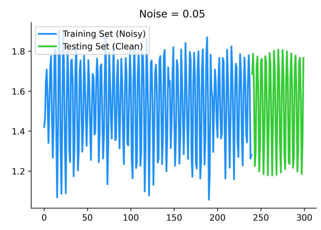
(b)

Figure 7: Performance of XNet on approximating different functions with varying numbers of parameters: (a) $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)\right)$; and (b) $\exp\left(\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)\right)$.

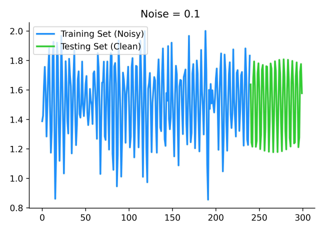
B.4 Function Approximation with noise



(a) Noise = 0

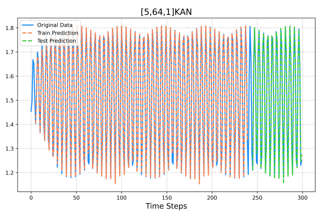


(b) Noise = 0.05

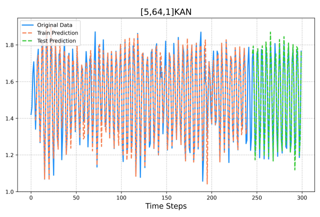


(c) Noise = 0.1

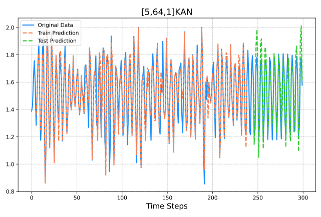
Figure 8: Function fitting tested on datasets with different noise levels: (a) Noise = 0, (b) Noise = 0.05, and (c) Noise = 0.1. Blue lines represent the noise-free ground truth function, while red dots represent the noisy training samples.



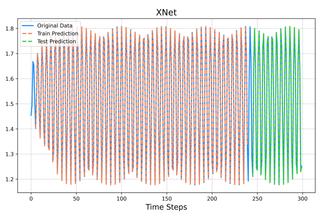
KAN (Noise = 0)



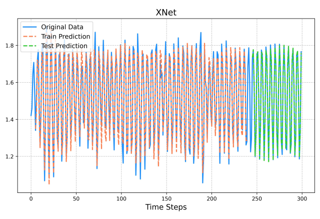
KAN (Noise = 0.05)



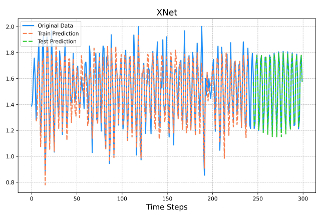
KAN (Noise = 0.1)



XNet (Noise = 0)



XNet (Noise = 0.05)



XNet (Noise = 0.1)

Figure 9: Comparison of the performance of KAN and XNet under different noise levels. Blue lines represent ground truth, while red lines show predictions. Note that XNet better captures the underlying pattern even at high noise levels, while KAN shows signs of overfitting.

Figures 8 and 9 provide a detailed comparison of how XNet and KAN handle increasing noise levels. XNet significantly outperforms KAN across all noise levels, with the advantage becoming more pronounced as noise increases. This confirms XNet's theoretical noise robustness through adaptive localization. At the highest noise level, XNet achieves 14.5× lower MSE than KAN while requiring only half the training time.

B.5 Heat equation

For solving the Heat equation under the framework of PINN, the loss function plots of various models (see Figure 10).

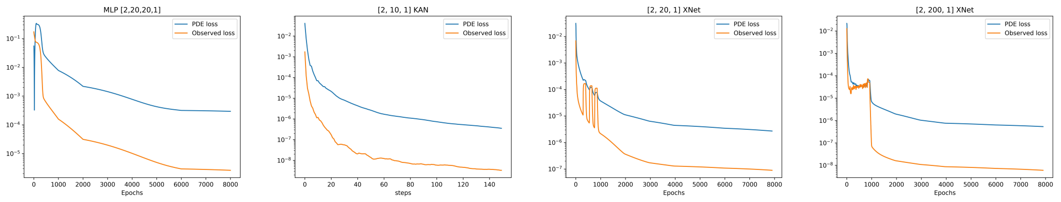


Figure 10: Comparison of KAN, PINN and XNet approximations on PDE loss.

B.6 Poisson equation

Task Description. We aim to solve a 2D Poisson equation $\nabla^2 v(x,y) = f(x,y)$, $f(x,y) = -2\pi^2 \sin(\pi x)\sin(\pi y)$, with boundary condition $v(-1,y) = v(1,y) = v(x,-1) = v(x,1) = 0$. The ground truth solution is $v(x,y) = \sin(\pi x)\sin(\pi y)$. We use the framework of physics-informed neural networks (PINNs) to solve this PDE, with the loss function given by

$$\begin{aligned} \text{loss}_{\text{pde}} &= \alpha \text{loss}_i + \text{loss}_b \\ &:= \alpha \frac{1}{n_i} \sum_{i=1}^{n_i} |v_{xx}(z_i) + v_{yy}(z_i) - f(z_i)|^2 + \frac{1}{n_b} \sum_{i=1}^{n_b} v^2 \,, \end{aligned} \tag{5}$$

where we use loss$_i$to denote the interior loss, discretized and evaluated by a uniform sampling of n_i points $z_i = (x_i, y_i)$ inside the domain, and similarly we use loss$_b$ to denote the boundary loss, discretized and evaluated by a uniform sampling of n_b points on the boundary. $\alpha = 0.1$ is the hyperparameter balancing the effect of the two terms.

Model Configurations. We evaluate four network architectures within the PINN framework: a two-layer MLP $[2, 20, 20, 1]$, a single-layer KAN $[2, 10, 1]$, and two XNet configurations with different widths $[2, 20, 1]$ and $[2, 200, 1]$.

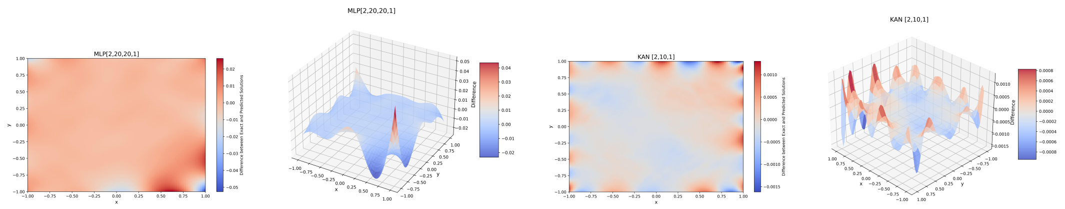


Figure 11: PINN and KAN Performance

Table 11: Comparison of XNet and KAN on the Poisson equation.

Metric	MSE	RMSE	MAE	Time (s)
PINN [2,20,20,1]	1.7998e-05	4.2424e-03	2.3300e-03	48.9
XNet (20)	1.8651e-08	1.3657e-04	1.0511e-04	57.2
KAN [2,10,1]	5.7430e-08	2.3965e-04	1.8450e-04	286.3
XNet (200)	1.0937e-09	3.3071e-05	2.1711e-05	154.8

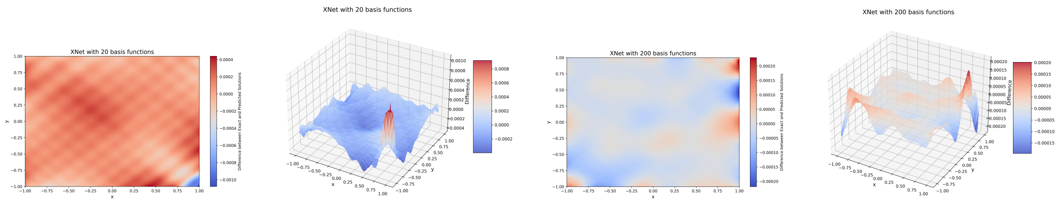


Figure 12: XNet Performance

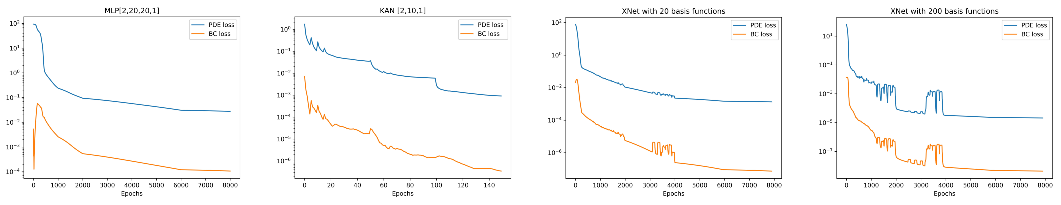


Figure 13: Comparison of KAN, PINN and XNet on PDE loss. XNet shows smoother and faster convergence, with significantly lower final loss values than other methods.

Comparative Results. XNet-200 achieves approximately 53× lower MSE than KAN while requiring only 54% of the computational time. The more compact XNet-20 still outperforms KAN with 3× better accuracy and 5× faster training. Therefore we speculate that the XNet might have the potential of serving as a good neural network representation for model reduction of PDEs. In general, KANs and PINNs are good at representing different function classes of PDE solutions, which needs detailed future study to understand their respective boundaries.

B.7 Benchmarking Study of different activation functions

This survey is compared with the existing survey/performance analysis and the experimental performance analysis of selected activation functions is performed over text data. The German to English translation is used to test the performance of the activation functions over text data. Benchmark Seq2Seq model consisting of a Long Short Term Memory (LSTM) based autoencoder network is used for the experiment. The model and dataset are downloaded from Kaggle2 [3].

Table 12: Experimental results for German to English language translation tasks.

Activations	Bleu Score	Activations	Bleu Score
Cauchy	21.47 ± 0.67	Swish	19.51 ± 0.97
Sigmoid	14.59 ± 0.47	ABReLU	17.55 ± 0.63
Tanh	20.93 ± 0.91	LiSHT	20.39 ± 0.93
Elliott	14.49 ± 0.96	SRS	20.66 ± 0.78
ReLU	18.88 ± 0.86	Mish	19.56 ± 1.15
LReLU	18.89 ± 0.82	PAU	20.11 ± 1.24
PReLU	20.04 ± 0.98	Softplus	16.78 ± 0.84
ELU	19.40 ± 1.33	CELU	18.71 ± 0.55
SELU	20.85 ± 0.64	GELU	18.75 ± 1.83
SIRENs	20.27 ± 0.85	SiLU	18.61 ± 0.72

Building upon the experiments conducted by Dubey et al. [6], we performed further updates and evaluations to assess the performance of different activation functions in language translation tasks. In these experiments, activation functions were applied to the feature embeddings immediately before the dropout layer. The training configuration included 50 epochs, a learning rate of 0.001, and a batch size of 256. Both the encoder and decoder had an embedding size of 300, with a dropout factor of 0.5. The Adam optimizer was employed, and training was conducted using cross-entropy loss. The Bleu score [21] with 4-gram evaluation was used as the metric for translation quality, with the mean and standard deviation of Bleu scores over five trials reported for each activation function. The results,

[3]`https://www.kaggle.com/datasets/parthplc/translatedtext`

summarized in Table 12, indicate that the Cauchy activation function consistently outperformed traditional activation functions such as Sigmoid, Tanh, and ReLU, achieving higher Bleu scores. This suggests that the Cauchy activation function is particularly well-suited for language translation tasks, as it enhances feature representation quality and demonstrates greater stability across training trials.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our paper's contributions: theoretically proving XNet achieves higher-order approximation rates than MLPs and KANs, and empirically demonstrating XNet's superior performance across function approximation, PDE solving, and reinforcement learning with PPO. All claims are supported by mathematical proofs in Section 3 and experimental results in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section Conclusion and Outlook, we acknowledge limitations of our current work and outline areas for future research, including extending XNet to transformer-style architectures, multi-modal tasks, and deepening theoretical analysis for high-dimensional regimes.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Our paper provides complete mathematical foundations in Section 3, with all assumptions clearly stated. Theorem 3.2 presents our main theoretical result with a proof sketch, and the complete proof is provided in Appendix A.5.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 provides detailed experimental setups for all three domains (function approximation, PDE solving, and reinforcement learning). We specify model architectures, hyperparameters, and training protocols. For each benchmark, we include specific function definitions, data generation procedures, and evaluation metrics to enable reproducibility.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the full code and reproduction instructions in the Supplementary Material (see "supplementary.zip").

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All experimental settings are thoroughly documented in their respective sections, with specific hyperparameters, network configurations, and optimization procedures detailed in each experimental subsection and supplementary tables.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Our paper reports appropriate statistical measures for all experiments. Tables 1-4 include MSE, RMSE, and MAE metrics across all comparative experiments. For reinforcement learning experiments in Section 4.3, we plot learning curves with standard deviation bands across 5 runs to show variability.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Our experimental setup is detailed in Appendix B, where we specify that all experiments were conducted using Tensorflow-CPU on a Dell computer with a 3.00 GHz Intel Core i9-13900KF processor. Section 4 also reports execution times for each experiment, enabling readers to gauge computational requirements for reproduction.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research fully complies with the NeurIPS Code of Ethics. We have carefully reviewed the guidelines and ensured our work respects intellectual property by properly citing all prior work, particularly KAN and XNet approaches. These methods pose no risks to privacy, security, or discrimination concerns as defined in the ethics guidelines.

   Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: In Conclusion and Outlook, we discuss both positive impacts (improved efficiency in scientific computing, reduced computational resources for high-precision tasks, interpretability improvements) and potential concerns (limitations in very high-dimensional settings). We emphasize XNet's potential for widespread applications through improved efficiency and reduced computational demands compared to deep learning methods.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our work presents a mathematical framework and neural architecture that poses no inherent risks for misuse. XNet does not involve scraped datasets, language models, or image generators that would require special safeguards against potential harmful applications.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All external assets used in our experiments are properly cited with appropriate attribution. We reference the original KAN paper (Liu et al., 2024) when comparing to their method, and clearly cite all benchmark datasets and environments used in our experiments. The DeepMind Control Suite environments used in Section 4.3 are credited with proper citation to their original authors.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release an anonymous implementation of XNet and associated experiments. Documentation includes training setup, model hyperparameters, and evaluation protocols.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our research does not involve human subjects or crowdsourcing. All experiments were conducted using computational methods and standard benchmark datasets.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: Our research does not involve human subjects, and therefore no IRB approval was required.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our research does not involve LLMs as components of our core method.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.