MODEL COMPARISONS: XNET OUTPERFORMS KAN

Anonymous authors
Paper under double-blind review

ABSTRACT

In the fields of computational mathematics and artificial intelligence, the need for precise data modeling is crucial, especially for predictive machine learning tasks. This paper explores further XNet, a novel algorithm that employs the complex-valued Cauchy integral formula, offering a superior network architecture that surpasses traditional Multi-Layer Perceptrons (MLPs) and Kolmogorov-Arnold Networks (KANs). XNet significant improves speed and accuracy across various tasks in both low and high-dimensional spaces, redefining the scope of data-driven model development and providing substantial improvements over established time series models like LSTMs.

1 INTRODUCTION

We initially proposed a novel method for constructing real networks from the complex domain using the Cauchy integral formula in Li et al. (2024); Zhang et al. (2024), utilizing Cauchy kernels as basis functions. This work comprehensively compares these networks with KANs, which use B-spline as basis functions in Liu et al. (2024), and MLPs to highlight our significant improvements.

Multi-layer perceptrons (MLPs) (Haykin (1994); Cybenko (1989); Hornik et al. (1989)), recognized as fundamental building blocks in deep learning, have their limitations despite their wide use, particularly in its accuracy, and large number of parameters needed in structures such as in transformers (Vaswani et al. (2017)), and lack interpretability without post-analysis tools (Cunningham et al. (2023)). The Kolmogorov-Arnold Networks (KANs) were introduced as a potential alternative, drawing on the Kolmogorov-Arnold representation theorem (Kolmogorov (1956); Braun & Griebel (2009)), and demonstrate their efficiency and accuracy in computational tasks, especially in solving PDEs and function approximation (Sprecher & Draghici (2002); Köppen (2002); Lin & Unbehauen (1993); Lai & Shen (2021); Leni et al. (2013); Fakhoury et al. (2022)).

In the swiftly advancing domain of deep learning, the continuous search for novel neural network designs that deliver superior accuracy and efficiency is pivotal. While traditional activation functions such as the Rectified Linear Unit (ReLU) (Nair & Hinton (2010)) have been widely adopted due to their straightforwardness and efficacy in diverse applications, their shortcomings become evident as the complexity of challenges escalates. This is particularly true in areas that demand meticulous data fitting and the solutions of intricate partial differential equations (PDEs). These limitations have paved the way for architectures that merge neural network techniques with PDEs, significantly enhancing function approximation capabilities in high-dimensional settings (Sirignano & Spiliopoulos (2018); Raissi et al. (2019); Jin et al. (2021); Wu et al. (2024); Zhao et al. (2023)).

Time series forecasting is critical in various sectors including finance, healthcare, and environmental science. While LSTM models are well-regarded for their ability to capture temporal dependencies (Yu et al. (2019); Zhao et al. (2017)), KAN models have also shown promise in managing time series predictions (Hochreiter & Schmidhuber (1997); Staudemeyer & Morris (2019); Xu et al. (2024)). Our study compares these models, providing insights into their applications and theoretical foundations. We also examine the performance of transformers and our novel XNet model in time series forecasting in the appendix, highlighting their capabilities in managing sequential data (Vaswani et al. (2017); Wen et al. (2023)).

Inspired by the mathematical precision of the Cauchy integral theorem, Li et al. (2024) introduced the XNet architecture, a novel neural network model that incorporates a uniquely designed Cauchy activation function. This function is mathematically expressed as:

$$\phi_a(x) = \frac{\lambda_1 * x}{x^2 + d^2} + \frac{\lambda_2}{x^2 + d^2},$$

where λ_1, λ_2, and d are parameters optimized during training. This design is not only a theoretical advancement but also empirical advantageous, offering a promising alternative to traditional models for many applications. By integrating Cauchy activation functions, XNet demonstrates superior performance in function approximation tasks and in solving low-dimensional PDEs compared to its contemporaries, namely Multilayer Perceptrons (MLPs) and Kolmogorov-Arnold Networks (KANs). This paper will systematically compare these architectures, highlighting XNet's advantages in terms of accuracy, convergence speed, and computational demands.

Furthermore, empirical evaluations reveal that the Cauchy activation function possesses a localized response with decay at both ends, significantly benefiting the approximation of localized data segments. This capability allows XNet to fine-tune responses to specific data characteristics, a critical advantage over the globally responding functions like ReLU.

The implications of this research are significant. It has been demonstrated that the XNet can serve as an effective foundation for general AI applications, our findings in this paper indicate that it can even outperform meticulously designed networks tailored for specific purposes.

Principal Contributions

Our study elucidates several critical advancements in the domain of neural network architectures and their applications:

(i) Enhanced Function Approximation Capabilities: We conduct a comparative analysis between XNet and KAN within the context of function approximation, demonstratting the superior performance of XNet, particularly in handling the Heaviside step function and complex high-dimensional scenarios. Detailed examinations are presented in Sections 3.1 through 3.3, showcasing empirical validations that underscore XNet's robust adaptability across varying dimensions.

(ii) Superiority in Physics-Informed Neural Networks: Utilizing the Poisson equation as a benchmark, we demonstrate XNet's enhanced efficacy within the Physics-Informed Neural Network (PINN) framework. Our results indicate that XNet significantly outstrips the performance metrics of both Multi-Layer Perceptron (MLP) and KAN, as detailed in Section 3.5. This investigation not only highlights XNet's prowess but also sets a new benchmark for subsequent applications in the field.

(iii) Innovation in Time Series Forecasting–By innovatively substituting the conventional feedforward neural network (FNN) with XNet in the LSTM architecture, we introduce the XLSTM model. In a series of time series forecasting experiments, XLSTM consistently surpasses traditional LSTM models in accuracy and reliability, establishing a new frontier in predictive analytics.

We summarize our results with a representative graph (fig 1), which compares the performance of various models in solving partial differential equations (PDEs). The parameterization of Kolmogorov-Arnold Networks (KANs) is fundamentally different from that of Multi-layer Perceptrons (MLPs); thus, even though KANs sometimes require fewer parameters and fewer training iterations, the training time can be substantially longer. In the context of solving PDEs, XNets with 200 basis functions typically operate at a pace that is 3-4 times slower than Physics-Informed Neural Networks (PINNs), 2 times faster than KANs, yet they achieve significantly higher precision-10000 times more precise than PINNs, to be exact.

2 Experimental Setup

Our research is designed to rigorously evaluate the capabilities of KAN and XNet across three fundamental domains: function approximation, solving partial differential equations (PDEs), and time series prediction. This structured evaluation allows us to systematically assess the performance and applicability of each model in varied computational tasks.

Function Approximation: We divide the function approximation experiments based on the dimensionality and complexity of the functions:

- **Low-Dimensional Functions:** Both irregular and regular functions are tested to evaluate the models' ability to handle variations in functional behavior and data distribution irregularities.

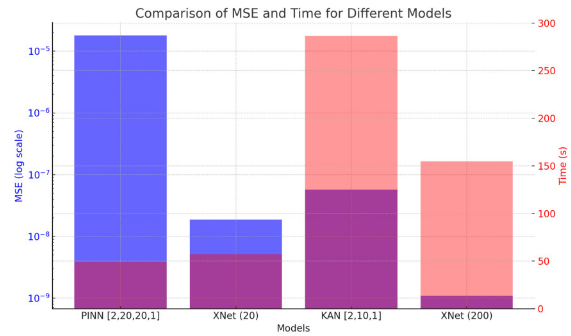
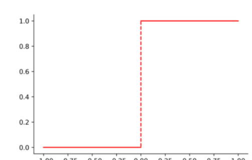
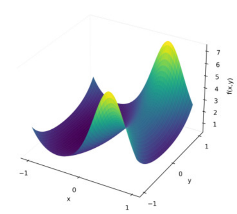
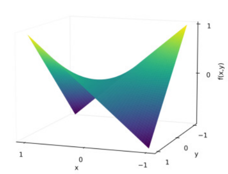


Figure 1: Comparing the MSE and training time for: PINN, XNet(20), KAN, and XNet(200). The MSE values are displayed on a logarithmic scale to better visualize the differences among the models.

- **High-Dimensional Functions:** Smooth functions that simulate complex real-world phenomena are used to examine the models' generalization in higher-dimensional spaces.

Evaluation metrics for accuracy, computational efficiency, and convergence are applied to each functional type.

Table 1: Low-dimensional and High-dimensional Functions Examples

Several Types of Functions and Their Examples		
$f(x) = \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases}$	$f(x, y) = \exp(\sin(\pi x) + y^2)$ jpg	$f(x, y) = xy$
		
High-dimensional Functions		
$f(x_1, x_2, x_3, x_4) = \exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2 + x_2^2)\right) + x_3 x_4\right)\right)$		
$f(x_1, \ldots, x_{100}) = \exp\left(\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)\right)$		

Solving Partial Differential Equations: We utilize a series of well-known differential equations from physics and engineering to test the efficacy of KAN and XNet. These include:

- Both linear and non-linear systems to provide a comprehensive assessment reflective of common scientific computing scenarios.

We consider the Poisson equation:

$$\nabla^2 v(x, y) = f(x, y), \quad f(x, y) = -2\pi^2 \sin(\pi x)\sin(\pi y),$$

with the boundary conditions,$v(-1, y) = v(1, y) = v(x, -1) = v(x, 1) = 0$. The PDE has the explict solution, $v(x, y) = \sin(\pi x)\sin(\pi y)$, as shown in the figure 2. In the subsection, we aim to compare the performance of three neural network architectures: PINN, KAN, and XNet.

Time Series Prediction: The proficiency of the models in capturing temporal dynamics and dependencies is explored through:

- The use of both synthetic and real-world time series datasets, which range from financial market data to weather forecasting, focusing on predictive accuracy, response time, and robustness at various temporal scales.

we also conducted time series forecasting experiments in different scenarios. One scenario is driven by mathematical and physical models. The example we provide is Apple's stock close price (adj) from the U.S. market, with the test period spanning from July 1, 2016 to July 1, 2017, as shown in the figure 3.

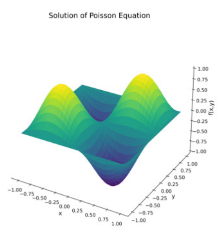


Figure 2: Solution of the Poisson equation

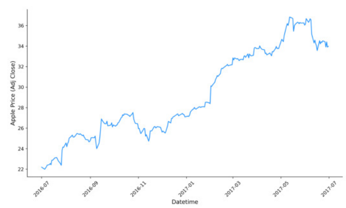


Figure 3: Apple's stock price: 7/1/2016 - 7/1/2017

Data Sets and Implementation Details: Detailed descriptions of the datasets is provided in Section 3.7. Additionally, implementation specifics such as hyperparameter settings, training procedures, and computational resources used are documented to ensure the experiments' reproducibility and transparency.

3 RESULTS

In Section 3.1, we perform the heaviside function approximation tasks using KAN and XNet. In Section 3.2, we conduct 2D smooth function approximation tasks using KAN and XNet. Section 3.3 evaluates the approximation of high-dimensional functions. In Section 3.4, we employ PINN, KAN, and XNet to construct physics-informed machine learning models for solving the 2D Poisson equation. In Section 3.5, we apply XNet to improve the performance of LSTM across various scenarios, then compare with KAN.

3.1 HEAVISIDE STEP FUNCTION APPRXIAMTION

The experimental comparison between XNet, B-spline, and KAN demonstrates XNet's superior approximation ability. Except for the first example, all other examples are from the referenced article, with KAN settings matching those from the original experiments. This ensures a fair comparison, fully proving that XNet has stronger approximation capabilities in various benchmarks.

Metric	MSE	RMSE	MAE
XNet with 64 basis functions	8.99e-08	3.00e-04	1.91e-04
[1,1]KAN with 200 grids	5.98e-04	2.45e-02	3.03e-03

Table 2: Performance comparison between XNet and KAN.

As shown in Figure 6 and 7, both B-Spline and KAN exhibit "overshoot," leading to local oscillations at discontinuities. We speculate that this is due to the fact that a portion of KAN's output is represented by B-Splines. While adjusting the grid can alleviate this phenomenon, it introduces complexity in tuning parameters (see Table 12 in appendix A.1). In contrast, XNet demonstrates superior performance, providing smooth transitions at discontinuities. Notably, in terms of fitting accuracy in these regions, XNet's MSE is 1,000-fold times smaller than that of KAN.

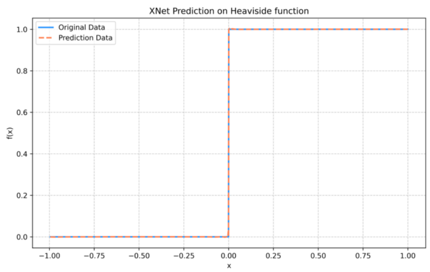


Figure 4: XNet approximation, with 64 basis functions

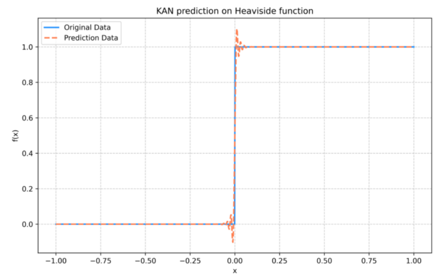


Figure 5: [1,1] KAN approximation, with k=3, grid =200

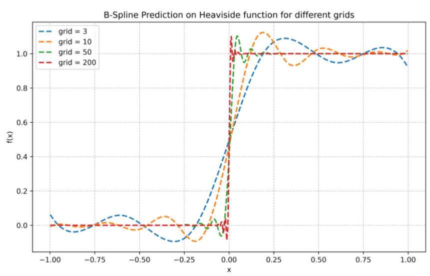


Figure 6: B-Spline comparision, with k=3

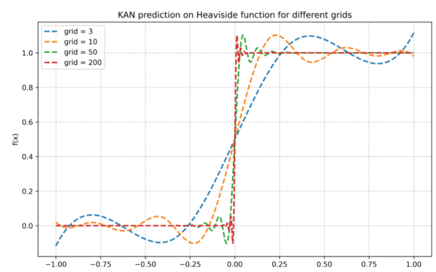


Figure 7: [1,1] KAN comparision, with k=3

3.2 Function Approximation with $\exp(\sin(\pi x) + y^2)$ and xy

The function used is $f(x, y) = \exp(\sin(\pi x) + y^2)$. Following the procedure described in the article, 1,000 points were used for training and another 1,000 points for testing. After sufficient training, the model's predictions were evaluated on a 100×100 grid. The KAN structure consists of a two hidden layer with configuration [2, 1, 1], We compare its computational efficiency with the XNet model using two examples: $\exp(\sin(\pi x) + y^2)$ and xy .

Following the official model configurations, XNet with 5,000 basis functions is trained with adam, while KAN is initialized to have G = 3, trained with LBFGS, with increasing number of grid points every 200 steps to cover G = 3, 5, 10, 20, 50. Overall, both networks performed similarly on these two-dimensional examples (see Table 3 and 4). However, XNet produced a more uniform fit, with no significant local oscillations (see Figure 9). In contrast, KAN exhibited sharp variations in certain regions, consistent with the behavior observed in the heaviside step function (see Section 3.1).

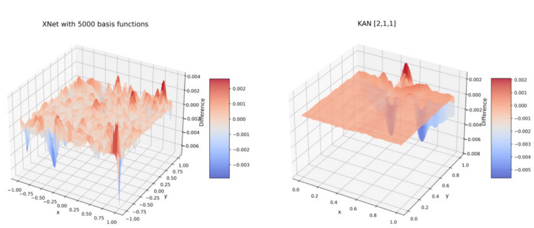


Figure 8: Difference on $\exp(\sin(\pi x) + y^2)$

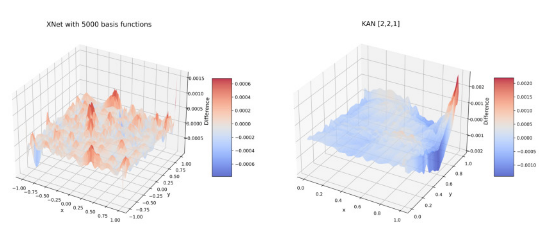


Figure 9: Difference on xy

3.3 Approximation with high-dimensional functions

We continue to compare the approximation capabilities of KAN and XNet in solving high-dimensional functions. Following the procedure described in the article, 8000 points were used for training and another

Table 3: Comparison of XNet and KAN on $\exp(\sin(\pi x) + y^2)$.

Metric	MSE	RMSE	MAE	Time (s)
XNet (5000)	3.9767e-07	6.3061e-04	4.0538e-04	61.0
KAN[2,1,1]	3.0227e-07	5.4979e-04	1.6344e-04	56.1

Table 4: Comparison of XNet and KAN on xy.

Metric	MSE	RMSE	MAE	Time (s)
XNet (5000)	2.1544e-08	1.4678e-04	1.0439e-04	61.8
KAN[2,2,1]	4.9306e-08	2.2205e-04	1.4963e-04	62.4

1000 points for testing. XNet is trained with adam, while KAN is initialized to have G = 3, trained with LBFGS, with increasing number of grid points every 200 steps to cover G = 3, 5, 10, 20, 50.

First, we consider the four-dimensional function $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)\right)$. For this case, the KAN structure is configured as [4,4,2,1], while XNet is equipped with 5,000 basis functions. Under the same number of iterations, XNet achieves higher accuracy in less time (see Table 5), the MSE is 1,000-fold smaller than that of KAN.

Table 5: Comparison of XNet and KAN on $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)\right)$.

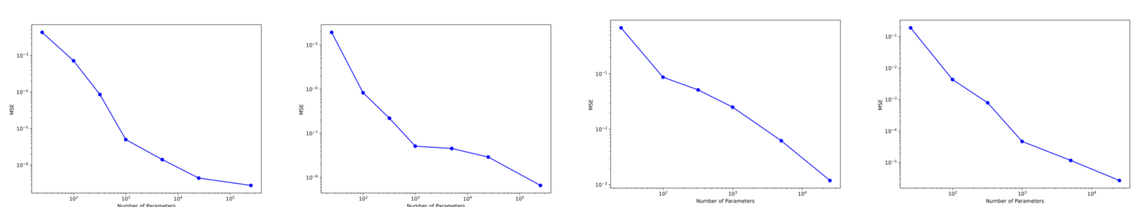
Metric	MSE	RMSE	MAE	Time (s)
XNet (5,000)	2.3079e-06	1.5192e-03	8.3852e-04	78.18
KAN [4,2,2,1]	2.6151e-03	5.1138e-02	3.6300e-02	143.1

Next, we consider the 100-dimensional function $\exp(\frac{1}{100}\sum_{i=1}^{100}\sin^2(\frac{\pi x_i}{2}))$. For this case, the KAN structure is configured as [100,1,1], while XNet has 5,000 basis functions. Under the same number of iterations, XNet achieved higher accuracy in less time compared to KAN (see Table 6).

Table 6: Comparison of XNet and KAN on $\exp\left(\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)\right)$.

Metric	MSE	RMSE	MAE	Time (s)
XNet (5,000)	6.8492e-04	2.6171e-02	2.0889e-02	158.69
KAN [100,1,1]	6.5868e-03	8.1159e-02	6.4611e-02	556.5

As dimensionality increases, the computational efficiency of KAN decreases significantly, while XNet shows an advantage in this regard. The approximation accuracy of both networks declines with increasing dimensions, which we hypothesize is related to the sampling method and the number of samples used.



$\exp(\sin(\pi x) + y^2)$ xy $\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2+x_2^2)\right)+x_3x_4\right)\right)$ $\exp\left(\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)\right)$

Figure 10: XNet Performance with Number of Parameters

As shown in Figure 10, XNet achieves high accuracy with relatively few network parameters. Moreover, as the number of parameters increases, XNet can further enhance its accuracy. Given its performance in function approximation tasks, both in terms of computational efficiency and accuracy, we conclude that

XNet is a highly efficient neural network with strong approximation capabilities. Building on this, in the following subsection, we apply PINN, KAN, and XNet to approximate the value function of the Poisson equation.

3.4 POSSION FUNCTION

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

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

where we use $\text{loss}_i$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. α is the hyperparameter balancing the effect of the two terms.

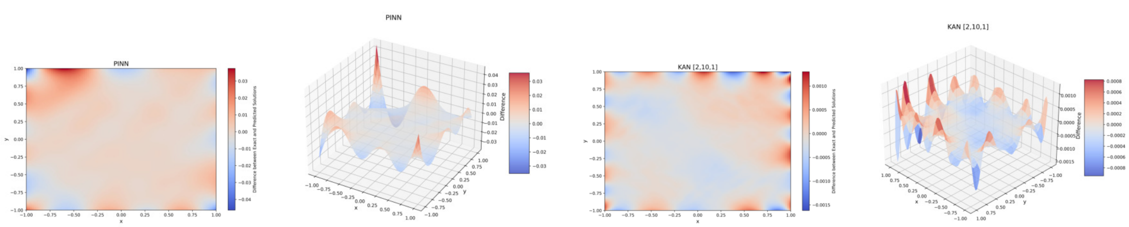


Figure 11: PINN and KAN Performance

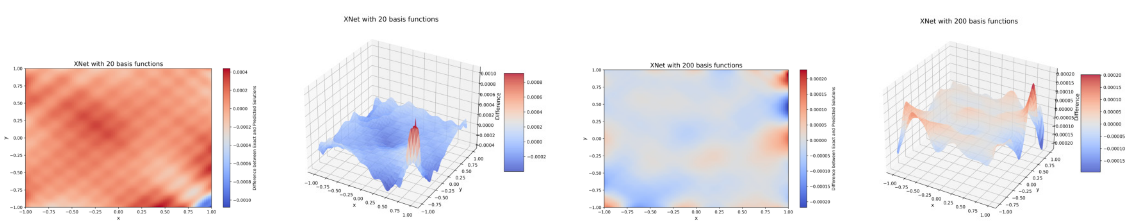


Figure 12: XNet Performance

We compare the KAN, XNet and PINNs using the same hyperparameters $n_i = 2500$, $n_b = 200$, and $\alpha = 0.01$. We measured the error in the L^2 norm (MSE) and observed that XNet achieved a smaller error, requiring less computational time, as shown in Figure 13. A width-200 XNet is 50 times more accurate and 2 times faster than a 2-Layer width-10 KAN; a width-20 XNet is 3 times more accurate and 5 times faster than a 2-Layer width-10 KAN (see Table 7). 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.

Table 7: 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

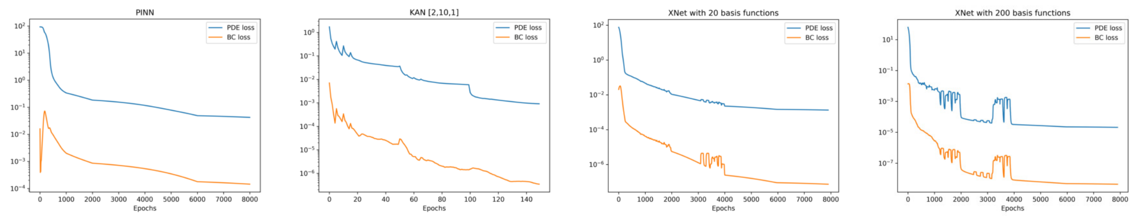


Figure 13: Comparison of KAN, PINN and XNet approximations on PDE loss.

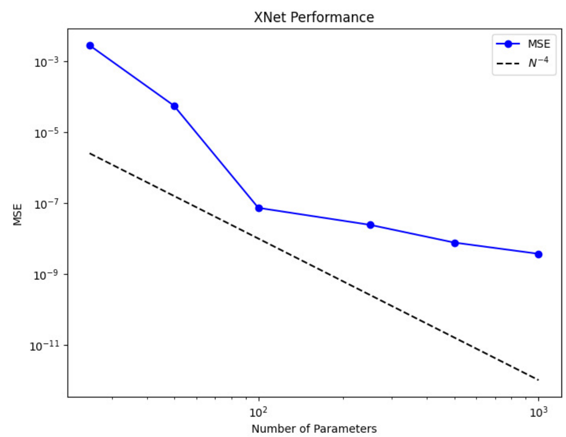


Figure 14: XNet Performance with Number of Parameters

3.5 XNet enhance the LSTM

Time prediction tasks can generally be categorized into two types: those driven by mathematical and physical models, and those that are data-driven. In the former, time prediction can often be formulated as a function approximation problem, while the latter involves noisy data, cannot be easily described by deterministic partial differential equations (PDEs). In this subsection, we introduce the **XLSTM** algorithm, which enhances the standard LSTM framework by replacing its feed-forward neural network (FNN) component with **XNet**. Across various examples, XLSTM consistently demonstrates superior predictive performance compared to the traditional LSTM. In the following experiments, we will demonstrate that XLSTM also significantly outperforms the KAN model in noisy time series examples. The KAN implementation for time series prediction is sourced from this repository: https://github.com/Nixtla/neuralforecast

Example 1: Predicting a Synthetic Time Series

The time series is generated by the following equations:

$$x_5^i = 0.1 * x_0^i x_1^i + 0.1 * sin(x_2^i x_3^i) + sin(x_4^i) + \mu^i, i = 1, 2, ..., n$$

and

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

where the initial conditions $x_0^0, x_1^0, x_2^0, x_3^0, x_4^0 \sim \text{rand}(0, 0.2)$ are randomly sampled in the range $[0, 0.2]$, and the noise term μ^i is sampled from a normal distribution, $\mu^i \sim N(0, \text{noise})$. This generates a time series $\{f^i = x_5^i\}_{i=1,...,n}$, with $n = 200$. In this example, the time series is governed by relatively simple functions. The task of predicting the sixth data point using the first five data points becomes a high-dimensional function approximation problem.

Figures [15] and [16] show a comparison of the predictive performance of LSTM and XLSTM on two scenarios: one with no noise (noise = 0) and one with moderate noise (noise = 0.05). The results indicate that XLSTM significantly outperforms LSTM in both settings, particularly under non-noisy conditions. When there is no noise, XLSTM achieves an MSE of 3.4252×10^{-11}, which is lower than that of LSTM (1.5925×10^{-7}). Similarly, XLSTM's **RMSE** and **MAE** are drastically lower than LSTM's, while the computation time remains comparable. In the presence of moderate noise (noise = 0.05), although XLSTM does not show a significant advantage in metrics such as MSE, it is clear from Figure (15) that XLSTM captures the underlying patterns of the data better than LSTM.

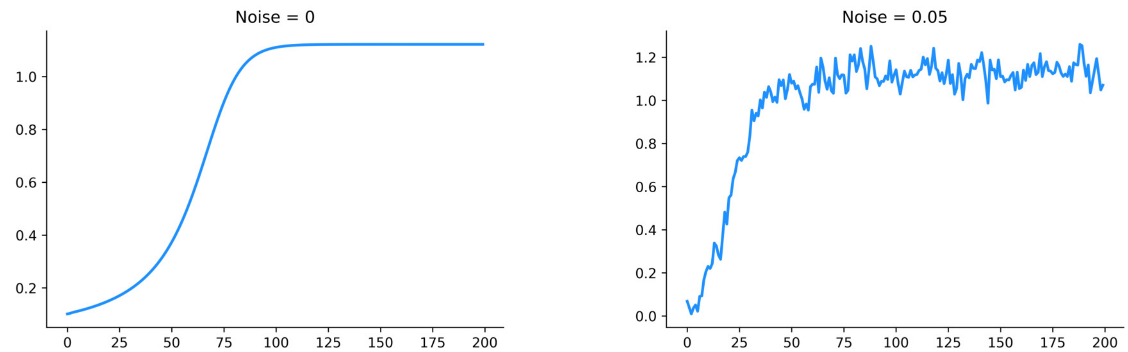


Figure 15: noise=0,0.05

Table 8: Comparison of LSTM and XLSTM on the example1 (noise=0).

Metric	MSE	RMSE	MAE	Time (s)
LSTM	1.5925e-07	3.9906e-04	3.9906e-04	9.01
XLSTM	3.4252e-11	5.8525e-06	5.8457e-06	9.42
[5,64,1]KAN	9.8281e-13	9.9137e-07	8.0000e-07	11.63

Table 9: Comparison of LSTM and XLSTM on the example1 (noise=0.05).

Metric	MSE	RMSE	MAE	Time (s)
LSTM	2.5919e-03	5.0911e-02	3.8814e-02	9.07
XLSTM	2.2080e-03	4.6990e-02	3.7182e-02	9.56
[5,64,1]KAN	4.6537e-03	6.8218e-02	5.3703e-02	11.59

In this example of a mathematical model-driven time series, XLSTM clearly outperforms LSTM, particularly in noisy and noise-free environments. Given these results, we hypothesize that XLSTM will also exhibit superior performance in highly noisy, real-world datasets, such as financial time series, where traditional LSTM models may struggle. The [5,64,1] KAN model, however, shows signs of overfitting, with excellent performance on the training set but noticeable degradation on the test set.

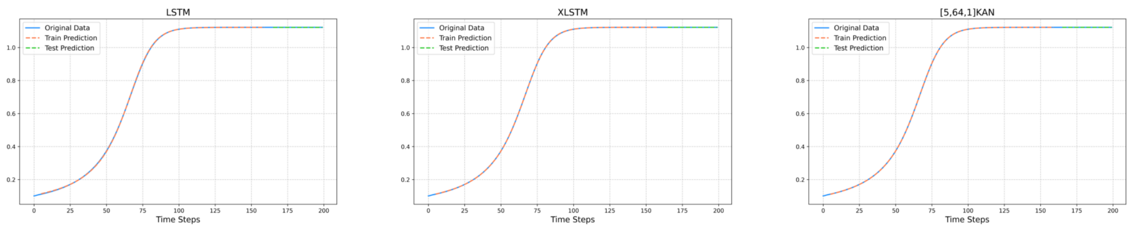
Example 2: Predicting a Financial Time Series

This is a toy model case with extremely noisy data. Stock price patterns are notoriously unpredictable, and we do not claim that our simplistic model outperforms others. We included this case merely to demonstrate the modelÂ's potential. In this experiment, we focus on Apple's stock price from the U.S. market, with the test period spanning from July 1, 2016 to July 1, 2017. The entire set of 252 data points is divided into two parts: 201 for training and 51 for testing. We consider using LSTM and XLSTM for time series prediction, where the model uses the first 10 data points and predicts the 11th. After 500 iterations, training was deemed complete.

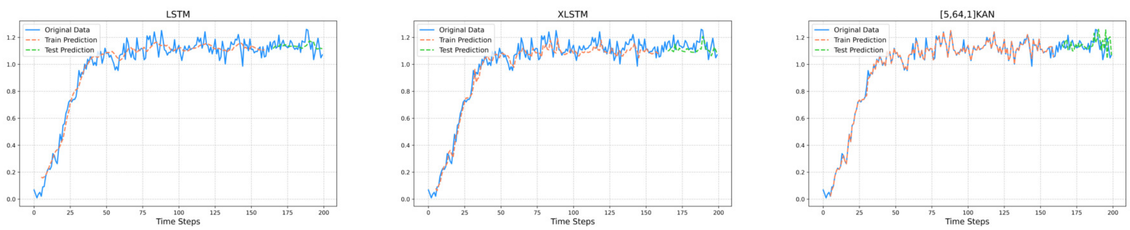
As shown in Figure 17, XLSTM aligns more closely with the original data, outperforming LSTM by a significant margin. In this example, the KAN model continues to exhibit overfitting, making it unsuitable for direct application to time series prediction with significant noise.

4 Summary and Outlook

1. **XNet vs. KAN for Function Approximation** Recently, KAN has gained popularity as a function approximator. However, our experiments demonstrate that XNet outperforms Kan, particularly when approximating discontinuous or high-dimensional functions.



LSTM (Noise = 0) XLSTM (Noise = 0) KAN (Noise = 0)



LSTM (Noise = 0.05) XLSTM (Noise = 0.05) KAN (Noise = 0.05)

Figure 16: Comparison of the performance of LSTM, XLSTM, and KAN under different noise levels. The first row shows the results for noise level 0, while the second row corresponds to noise level 0.05.

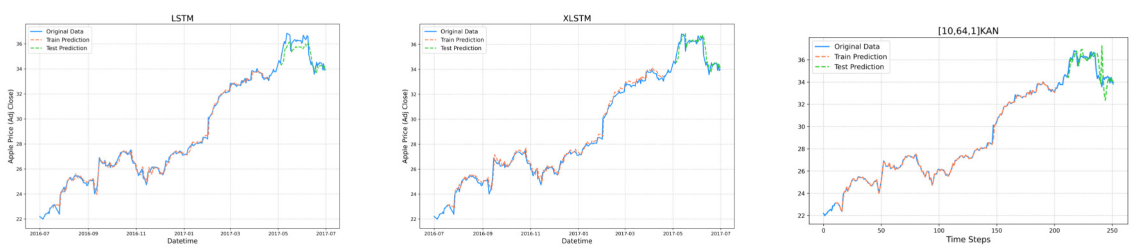


Figure 17: Comparison of the performance of LSTM, XLSTM, and KAN on Apple's stock price

Table 10: Comparison of LSTM, XLSTM and KAN on the Financial Time Series.

Metric	MSE	RMSE	MAE	Time (s)
LSTM	3.3768E-01	5.8110E-01	4.8787E-01	8.9574
XLSTM	2.3878E-01	4.8865E-01	3.3764E-01	10.1159
[10,64,1]KAN	8.5918e-01	9.2692e-01	5.9108e-01	11.7505

2. **XNet in the PINN Framework** Within the Physics-Informed Neural Networks (PINN) framework, we verified that using KAN significantly improves the accuracy of traditional PINNs. Moreover, implementing XNet further enhances both accuracy and computational efficiency. We hypothesize this is due to XNet's superior approximation capabilities.

3. **Enhancing LSTM with XNet** Given XNet's ability to capture complex data features, we found that XNet can enhance LSTM performance by replacing the embedded feed-forward neural network (FNN) within the LSTM structure.

4. **Potential Applications of XNet** We believe that XNet can improve the performance of models in other machine learning domains, including image recognition, image generation, computer vision, and more.

REFERENCES

Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive approximation*, 30:653–675, 2009.

Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models. *arXiv preprint arXiv:2309.08600*, 2023.

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

Daniele Fakhoury, Emanuele Fakhoury, and Hendrik Speleers. Exsplinet: An interpretable and expressive spline-based neural network. *Neural Networks*, 152:332–346, 2022.

Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997. URL `https://api.semanticscholar.org/CorpusID:1915014`.

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

P. Jin, L. Lu, G. Pang, Z. Zhang, and G. E. Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021. doi: 10.1038/s42256-021-00302-5. URL `https://www.nature.com/articles/s42256-021-00302-5`.

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.

Mario Köppen. On the training of a kolmogorov network. In *Artificial Neural Networksâ€"ICANN 2002: International Conference Madrid, Spain, August 28â€"30, 2002 Proceedings*, volume 12, pp. 474–479. Springer, 2002.

Ming-Jun Lai and Zhaiming Shen. The kolmogorov superposition theorem can break the curse of dimensionality when approximating high dimensional functions. *arXiv preprint arXiv:2112.09963*, 2021.

Pierre-Emmanuel Leni, Yohan D Fougerolle, and FrÃ©dÃ©ric Truchetet. The kolmogorov spline network for image processing. In *Image Processing: Concepts, Methodologies, Tools, and Applications*, pp. 54–78. IGI Global, 2013.

Xin Li, Zhihong Xia, and Hongkun Zhang. Cauchy activation function and xnet. *arXiv preprint arXiv:2409.19221*, 2024.

Ji-Nan Lin and Rolf Unbehauen. On the realization of a kolmogorov network. *Neural Computation*, 5(1): 18–20, 1993.

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024. URL `https://arxiv.org/abs/2404.19756`.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

M. Raissi, P. Perdikaris, and G. 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. doi: 10.1016/j.jcp.2018.10.045. URL `https://www.sciencedirect.com/science/article/pii/S0021999118307125`.

J. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018. doi: 10.1016/j.jcp.2018.08.029. URL `https://www.sciencedirect.com/science/article/pii/S0021999118305525`.

David A Sprecher and Sorin Draghici. Space-filling curves and kolmogorov superposition-based neural networks. *Neural Networks*, 15(1):57–67, 2002.

Ralf C Staudemeyer and Eric Rothstein Morris. Understanding lstm–a tutorial into long short-term memory recurrent neural networks. *arXiv preprint arXiv:1909.09586*, 2019.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, volume 30, 2017.

Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in time series: a survey. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, pp. 6778–6786, 2023.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for pdes on general geometries. *arXiv preprint arXiv:2402.02366*, 2024. URL https://arxiv.org/abs/2402.02366.

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

Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural Computation*, 31, 2019.

Hongkun Zhang, Xin Li, and Zhihong Xia. Cauchynet: Utilizing complex activation functions for enhanced time-series forecasting and data imputation. Submitted for publication, 2024.

Zheng Zhao, Weihai Chen, Xingming Wu, Peter C. Y. Chen, and Jingmeng Liu. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11, 2017.

Zhiyuan Zhao, Xueying Ding, and B Aditya Prakash. Pinnsformer: A transformer-based framework for physics-informed neural networks. *arXiv preprint arXiv:2307.11833*, 2023. URL https://arxiv.org/abs/2307.11833.

A Appendix

A.1 Additional Experiment Details

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).

A.2 A.1 Function Approximation

For 1d heaciside function, we set different configurations. The results are shown as follows

Table 11: 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

For 2d functions, loss function

for high-dimensional functions, loss functions

Table 12: 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

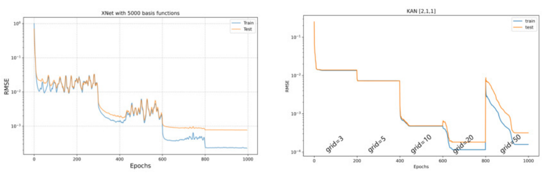


Figure 18: Loss on $\exp(\sin(\pi x) + y^2)$

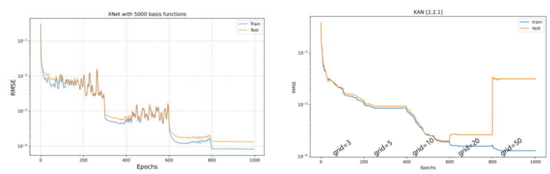
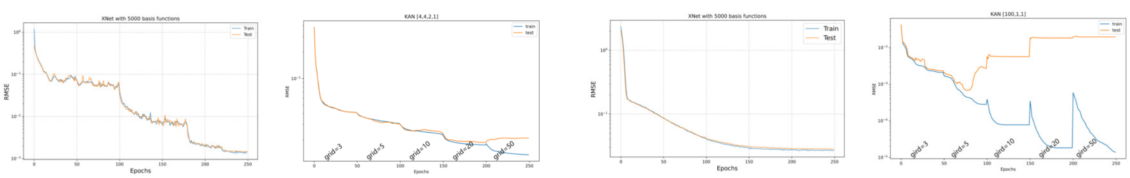


Figure 19: Loss on xy



$\exp\left(\frac{1}{2}\left(\sin\left(\pi(x_1^2 + x_2^2)\right) + x_3 x_4\right)\right)$ $\exp(\frac{1}{100}\sum_{i=1}^{100}\sin^2(\frac{\pi x_i}{2}))$

Figure 20: Loss on high-dimensional functions

A.3 A.2 TIME SEREIS

In Section 3.5, we present two examples to forecast future unknown data using LSTM and XLSTM. In the function-driven example (15), the loss functions of LSTM and XLSTM are shown in Figure 21; for the task of predicting AppleÃ¢Â€Â™s stock price, the loss functions of LSTM and XLSTM are illustrated in Figure 22.

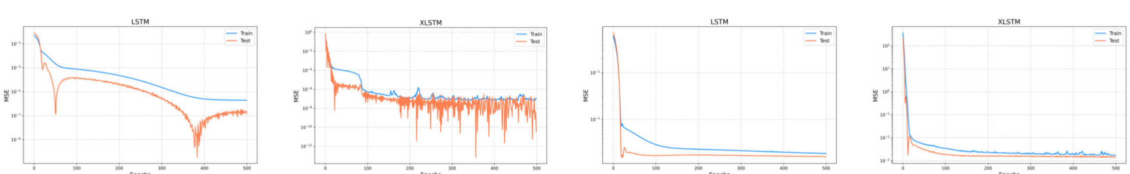


Figure 21: Loss of LSTM and XLSTM on Example 1.

Example 2 loss

A.4 TIME SERIES

There exists two types of time prediction applications. One is driven by mathematical and physical models, where time prediction can essentially be viewed as a function approximation. The other is data-driven, where the data often contains significant noise and cannot be easily described by PDEs. In this section, we introduce the XLSTM algorithm, which replaces the FNN component in the standard LSTM framework with XNet. In the following examples, XLSTM consistently demonstrates superior predictive performance.

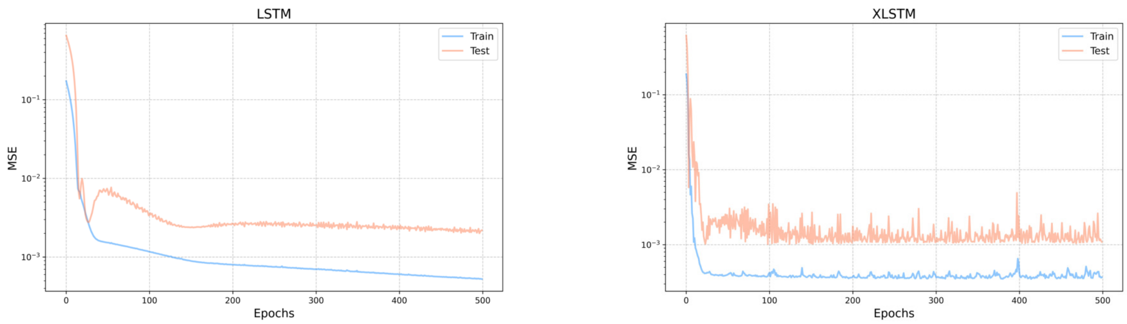


Figure 22: loss of LSTM and XLSTM on Apple's stock price

A.5 HIGH-DIMENSIONAL FUNCTION 1

The time series is generated by the following equations:

$$x_5^i = 0.1 * x_0^i * x_1^i + 0.5 * sin(x_2^i * x_3^i) + 1 * sin(x_4^i), i = 1, 2, ..., n$$

and

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

with

$$x_0^0, x_1^0, x_2^0, x_3^0, x_4^0 \sim rand(0, 0.2).$$

This generates the time series $\{f^i = x_5^i\}_{i=1,...,n}$. We consider the data n=200. In this example, the time series is driven by simple functions. Specifically, when the task is to predict the sixth data point using the first five, it essentially becomes a high-dimensional function approximation problem.

We first split the data into a training set (80%) and a validation set (20%) and performed predictions using different models including 2-Layer width-10 FNN, 1-layer width-10 LSTM, width-10 XNet and width-10 XLSTM.

For each training iteration, the first five data points were used as input, and the model predicted the sixth data point, which was then compared with the target values. After five thousand iterations, the training process was considered complete. On the test set, we used the first five data points as input to predict the sixth, sliding through the sequence until all predictions were made. In essence, this can be viewed as a function-fitting problem.

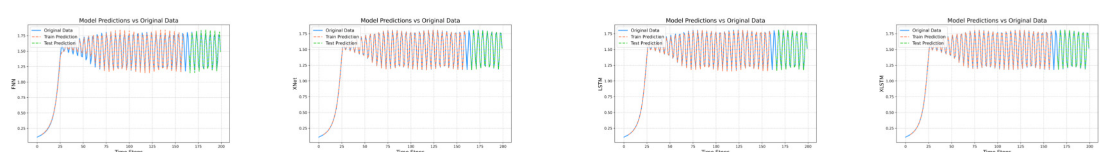


Figure 23: different models

XLSTM demonstrates stronger predictive capabilities compared to standard LSTM. With the same training cost, XLSTM improves accuracy by a factor of fifty.

	FNN	XNet	LSTM	X-LSTM
MSE (Val)	1.6253E-03	1.0758E-05	1.1187E-04	2.5222E-06
RMSE (Val)	4.0315E-02	3.2800E-03	1.0577E-02	1.5881E-03
MAE (Val)	3.3874E-02	2.7836E-03	9.0519E-03	1.1279E-03
MSE (Train)	3.0175E-02	3.3013E-03	8.2499E-03	1.3336E-03
Time(s)	6	6	12	12

Next, we apply XLSTM to stock price prediction and power consumption forecasting, where it again demonstrates stronger predictive capabilities compared to LSTM.

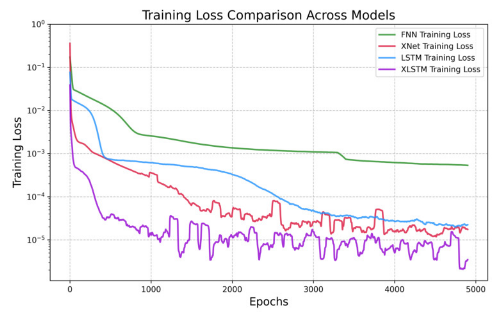


Figure 24: loss

A.6 ELECTRIC POWER

In this experiment, the time series represents electricity consumption in Zone 1 of the United States, with the test period from 01/01/2017 00:00 to 01/14/2017 21:20. The data is sourced from https://www.kaggle.com/datasets/fedesoriano/electric-power-consumption. The 2,000 data points are divided into two parts: 1,602 for training and 398 for testing. During training, the model takes the first 10 data points as input and predicts the 11th, comparing it with the target.

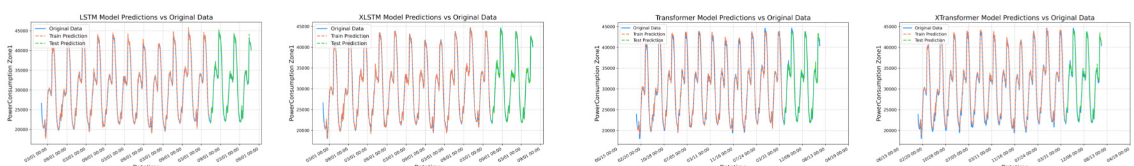


Figure 25: electric power

XNet enhanced transformer and lstm model. transformer has little advantage in this case

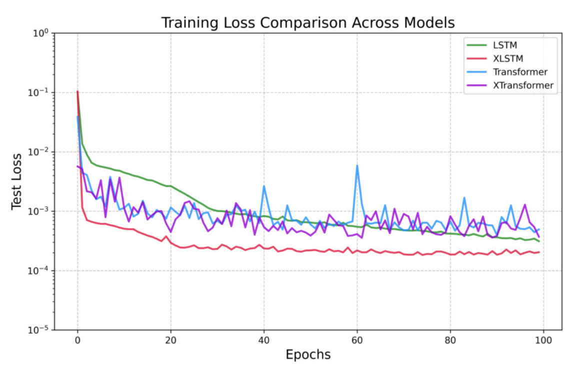


Figure 26: loss

	LSTM	XLSTM	Transformer	XTransformer
MSE (Val)	2.3937E+05	1.1505E+05	3.7482E+05	2.7868E+05
RMSE (Val)	4.8925E+02	3.3920E+02	6.1223E+02	5.2790E+02
MAE (Val)	3.2422E+02	2.6051E+02	4.9423E+02	4.1865E+02
MSE (Train)	3.2729E+02	2.4623E+02	3.8049E+02	3.7939E+02
Time(s)	15	26	127	90