Kolmogorov Superposition Theorem: Theory, Challenges, and Applications in Neural Networks

Team 3 - KAN

Maxwell Lam
Bagley College of Engineering
Mississippi State University
Starkville, MS 39762, USA
mvl57@msstate.edu

Ben Moore
Bagley College of Engineering
Mississippi State University
Starkville, MS 39762, USA
bhm128@msstate.edu

Matthew Robinson
Bagley College of Engineering
Mississippi State University
Starkville, MS 39762, USA
mbr253@msstate.edu

Justin Xie
Bagley College of Engineering
Mississippi State University
Starkville, MS 39762, USA
hx34@msstate.edu

***Abstract*—The Kolmogorov Superposition Theorem (KST) provides a powerful mathematical framework for representing multivariate continuous functions as superpositions of univariate functions. This paper presents a comprehensive study of KST, its applications in neural networks, and the challenges associated with its practical implementation, especially in high-dimensional settings. We explore recent advancements, propose methods to address existing challenges, and present experimental results comparing KST-based neural networks with traditional Multilayer Perceptrons (MLPs). Our findings highlight the trade-offs between computational efficiency and model accuracy, offering insights into optimizing neural network architectures based on KST.**

***Index Terms*—Kolmogorov Superposition Theorem, Neural Networks, High-Dimensional Function Approximation, Spline Functions, Computational Efficiency, Optimization Strategies**

I. Introduction

In 1957, A.N. Kolmogorov introduced the Kolmogorov Superposition Theorem (KST), offering a mathematical framework for representing multivariate continuous functions as superpositions of simpler univariate functions. The theorem states that any multivariate continuous function $f : [0, 1]^n \to \mathbb{R}$ can be represented as a finite sum of continuous univariate functions of linear combinations of its variables:

$$f(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{2n+1} \varphi_i \left(\sum_{j=1}^{n} \psi_{ij}(x_j) \right),$$

where φ_i and ψ_{ij} are continuous univariate functions.

This theorem has significant implications in computational mathematics, neural networks, and function approximation. Over the years, it has been refined for applications in encryption, image analysis, and neural networks. Recent advancements have improved the practical utility of KST by providing Lipschitz continuous functions and reducing storage and evaluation complexity compared to earlier methods.

Hecht-Nielsen's observation in 1987 highlighted that Kolmogorov's theorem could serve as the basis for a three-layer neural network capable of approximating any function, further propelling interest in this area. However, initial algorithms lacked guaranteed approximation accuracy, a limitation later addressed by research providing constructive proofs and computational algorithms with more reliable guarantees. These developments have led to deeper studies of computational methods and neural network architectures.

Innovations such as spline-based networks and architectures like the Kolmogorov Spline Network (KSN) have shown promise in addressing dimension reduction problems. These developments involve incorporating space-filling curves and spline functions to reduce storage and computational complexity. Despite these strides, challenges persist in crafting efficient neural networks based on KST, constructing accurate surrogates for high-dimensional data, and ensuring generalization. Solving these issues is critical for advancing fields such as deep learning, image analysis, and high-dimensional data processing.

II. Literature Survey

A. Taxonomy

The literature on KST can be classified into four categories:

1) **Kolmogorov Superposition Theorem (KST)**: Focuses on the theorem's theory, proofs, and computational aspects.
2) **Neural Networks Based on KST (NN-KST)**: Explores how the theorem is utilized in neural network architectures.
3) **Mathematical and Computational Methods (MCM)**: Involves techniques and algorithms directly related to

KST or applied in similar function approximation scenarios.

4) **Spline-Based Methods and Approximation (SBMA)**: Focuses on spline functions and approximations in the context of KST for high-dimensional function approximation.

Figure 1 illustrates the taxonomy of the literature, categorizing key works based on their primary focus.

B. Chronological Overview

The development of KST and its applications have evolved over the years. Figure 2 presents a visual representation of the literature chronology, and Table I provides a textual summary.

III. Research Gaps

Despite significant advancements, challenges remain in achieving optimal computational efficiency for high-dimensional problems. Recent research on spline-based approximations highlights the need for methods that can effectively handle the curse of dimensionality. Continued exploration of neural networks based on KST, including novel architectures like the Kolmogorov Spline Network (KSN), shows promise for addressing these challenges and expanding the applicability of the theorem in modern computational fields.

IV. Problem Importance and Challenges

A. Problem Statement

The central problem addressed in this research is the efficient approximation of complex multivariate functions using the KST framework, particularly in high-dimensional settings and its application in neural networks. We aim to represent a multivariate continuous function f as:

$$f(x_1, x_2, \ldots, x_n) \approx \sum_{i=1}^{n} \varphi_i \left(\sum_{j=1}^{n} \psi_{ij}(x_j) \right),$$

where φ_i and ψ_{ij} are continuous univariate functions representing the outer and inner functions in the superposition, respectively.

B. Importance of the Problem

Efficient approximation of complex multivariate functions is fundamental to many computational and data-driven fields. While KST provides a theoretical framework, its practical application faces challenges, especially in high-dimensional settings. Overcoming these challenges can lead to significant advancements in modeling complex systems and processing high-dimensional data.

C. Challenges

1) **Function Decomposition and Representation**: Determining the appropriate decomposition, specifically finding the univariate inner functions ψ_{ij} and outer functions φ_i, is challenging for high-dimensional, complex functions. The non-constructive nature of Kolmogorov's original proof does not offer explicit algorithms for obtaining these functions.
2) **Computational Complexity and Efficiency**: High-dimensional problems lead to significant computational overhead when applying KST in practical settings like neural networks. Efficient computation and evaluation of the univariate functions while managing exponential increases in storage and processing demands are critical challenges.
3) **Integration into Neural Networks**: Designing KST-based neural networks that can generalize well and provide reliable function approximations requires further development. The integration of KST into modern neural network architectures remains non-trivial.

V. Difficulty Justification

A. Function Decomposition and Representation

The "curse of dimensionality" makes it challenging to find efficient representations as the number of dimensions increases. For example, in a 100-dimensional space, even a simple hypercube has 2^{100} corners, which is computationally intractable.

B. Computational Complexity and Efficiency

The computational complexity of KST-based approximations grows exponentially with the number of dimensions. If each dimension requires m basis functions, a naive implementation would require $O(m^n)$ operations for an n-dimensional function, becoming infeasible for high-dimensional problems.

C. Integration into Neural Networks

Traditional neural network architectures are not designed to directly implement the KST structure. The non-linearity introduced by the KST decomposition can lead to training difficulties, such as vanishing or exploding gradients, especially in deep networks.

VI. Expected Contributions

This research aims to bridge the gap between the theoretical foundations of KST and its practical implementation in modern machine learning contexts. By developing novel algorithms and architectures that efficiently leverage KST, we expect to:

- **Advance Function Approximation Methods**: Provide efficient techniques for approximating high-dimensional multivariate functions.
- **Enhance Neural Network Architectures**: Introduce KST-based neural network models that outperform traditional architectures in specific tasks.

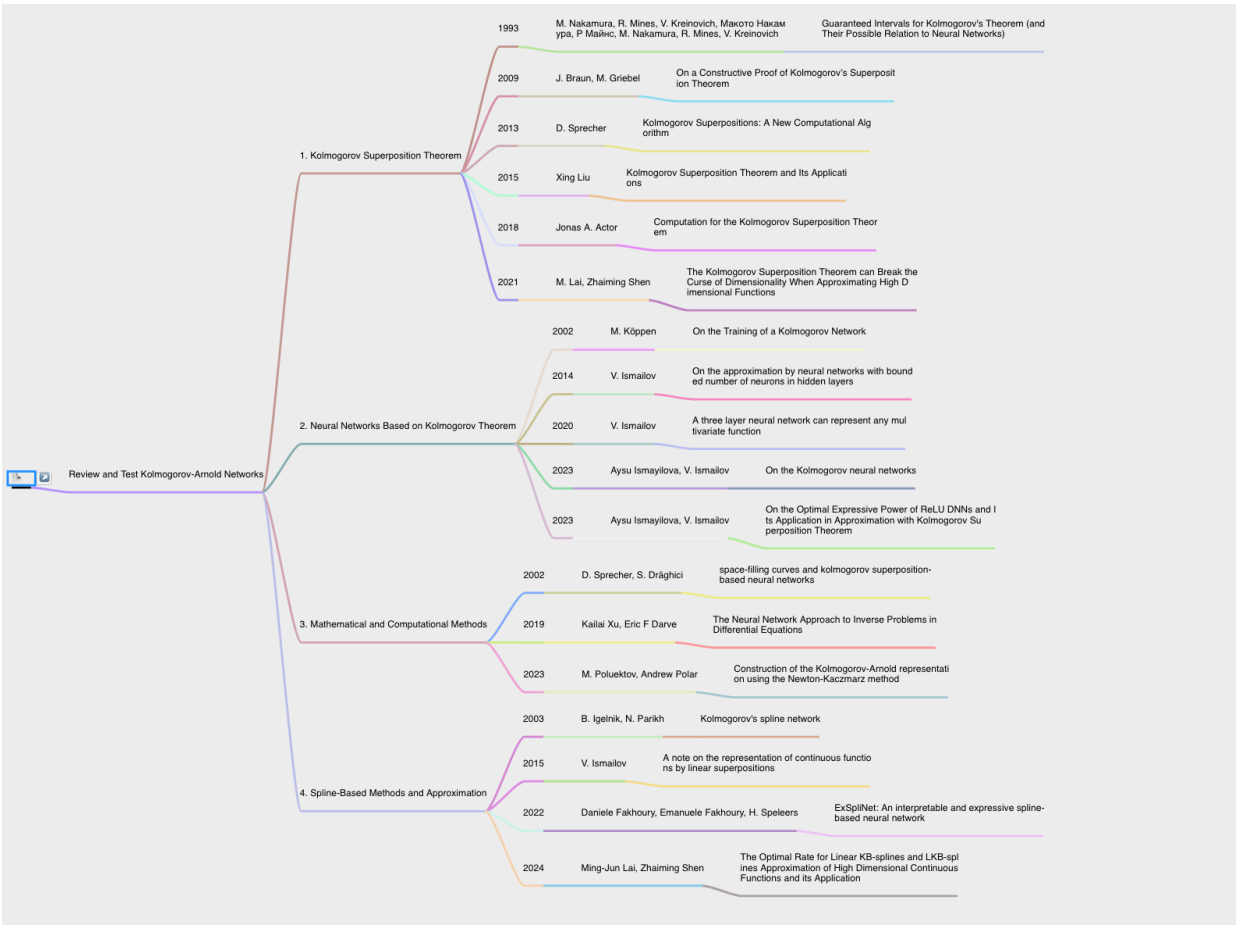

Fig. 1. Taxonomy of Literature

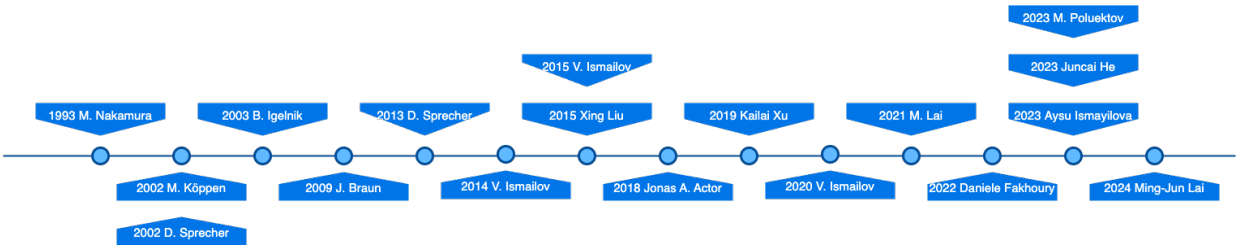

Fig. 2. Chronological Diagram of Literature

- **Address the Curse of Dimensionality**: Offer solutions that mitigate computational challenges associated with high-dimensional data.

These contributions could lead to improvements in neural network performance, especially in tasks involving complex, high-dimensional data such as image analysis and scientific simulations.

VII. Proposed Methods

The primary difference between a Multilayer Perceptron (MLP) and the Kolmogorov Superposition Theorem (KST) lies in how they approach function approximation. MLPs are neural networks that approximate multivariate functions using layers of interconnected neurons, leveraging weights and biases learned through backpropagation. In contrast, KST represents any continuous multivariate function as a finite sum of compositions of continuous univariate functions, simplifying high-dimensional problems by breaking them into univariate components. While MLPs are versatile and widely used, KST offers a mathematically exact approach to function representation but can be computationally intensive and challenging to implement directly in high-dimensional settings.

To evaluate the Kolmogorov Superposition Theorem (KST) and compare its performance with a traditional Multilayer Perceptron (MLP), we propose three specific architectures for this study. The MLP architecture features two dense hidden

TABLE I
CHRONOLOGICAL OVERVIEW OF LITERATURE

Year	Category	Authors	Title
1993	KST	M. Nakamura, R. Mines, V. Kreinovich	Guaranteed Intervals for Kolmogorov's Theorem
2002	NN-KST	M. Köppen	On the Training of a Kolmogorov Network
2002	MCM	D. Sprecher, S. Drăghici	Space-Filling Curves and Kolmogorov Superposition-Based Neural Networks
2003	SBMA	B. Igelnik, N. Parikh	Kolmogorov's Spline Network
2009	KST	J. Braun, M. Griebel	On a Constructive Proof of Kolmogorov's Superposition Theorem
2013	KST	D. Sprecher	Kolmogorov Superpositions: A New Computational Algorithm
2014	NN-KST	V. Ismailov	On the Approximation by Neural Networks with Bounded Number of Neurons in Hidden Layers
2015	KST	Xing Liu	Kolmogorov Superposition Theorem and Its Applications
2015	SBMA	V. Ismailov	A Note on the Representation of Continuous Functions by Linear Superpositions
2018	KST	J.A. Actor	Computation for the Kolmogorov Superposition Theorem
2019	MCM	K. Xu, E.F. Darve	The Neural Network Approach to Inverse Problems in Differential Equations
2020	NN-KST	V. Ismailov	A Three-Layer Neural Network Can Represent Any Multivariate Function
2021	KST	M. Lai, Z. Shen	The Kolmogorov Superposition Theorem Can Break the Curse of Dimensionality
2022	SBMA	D. Fakhoury, E. Fakhoury, H. Speleers	ExSpliNet: An Interpretable and Expressive Spline-Based Neural Network
2023	NN-KST	A. Ismayilova, V. Ismailov	On the Kolmogorov Neural Networks
2023	NN-KST	J. He	On the Optimal Expressive Power of ReLU DNNs and Its Application in Approximation with KST
2023	MCM	M. Poluektov, A. Polar	Construction of the Kolmogorov-Arnold Representation Using the Newton-Kaczmarz Method
2024	SBMA	M.-J. Lai, Z. Shen	The Optimal Rate for Linear KB-Splines and LKB-Splines Approximation of High-Dimensional Continuous Functions

Note: KST = Kolmogorov Superposition Theorem; NN-KST = Neural Networks Based on KST; MCM = Mathematical and Computational Methods; SBMA = Spline-Based Methods and Approximation.

layers of 128 nodes each, serving as a baseline for comparison. For the KST-based models, we design two architectures: one using polynomial fitting and another using spline fitting for the inner functions. Both KST models employ two smaller hidden layers with 20 nodes each, optimizing computational efficiency while maintaining approximation accuracy. Figure 4 illustrates these proposed architectures, highlighting the differences in layer size and structure between the MLP baseline and the two KST-based networks.

To shorten the training time for KST, a key solution is replacing the spline functions used in the inner function representation with polynomial fits. Spline functions, while effective for capturing complex variations, are computationally intensive and can significantly increase training time. By substituting splines with polynomial fits, the computational complexity is reduced, enabling faster training while maintaining comparable levels of accuracy. This approach leverages the simplicity and efficiency of polynomial computations, optimizing KST for practical applications where time efficiency is critical. However, polynomials provide a single global approximation over the entire domain, whereas splines use piecewise-defined functions that can adapt to local variations more effectively. Due to this global representation, polynomial fits may struggle to capture sharp changes or complex patterns in the data, leading to lower accuracy compared to splines.

To address this limitation, we implement a hybrid optimization strategy by combining the Adaptive Moment Estimation (Adam) optimizer for rapid convergence in the initial stages of training with the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm for fine-tuning the results. Given the limited-memory nature of L-BFGS, recursion is employed to optimize the loss function effectively within memory constraints. This combined approach aims to reduce errors and enhance the accuracy of the polynomial fit while maintaining its computational efficiency. This strategy ensures a balance between computational efficiency and model accuracy.

A. Addressing Function Decomposition and Representation

- **Polynomial Approximation**: By replacing spline functions with polynomial fits in the inner functions ψ_{ij}, we simplify the function decomposition, reducing computational overhead.
- **Optimization of Inner Functions**: Employ optimization techniques to fine-tune the univariate functions, enhancing the model's ability to capture complex patterns despite using global approximations.

B. Addressing Computational Complexity and Efficiency

- **Hybrid Optimization Strategy**: Utilize the Adam optimizer for rapid convergence during initial training phases and L-BFGS for precise fine-tuning, optimizing computational resources.
- **Efficient Computation**: Implement recursive strategies within L-BFGS to manage memory constraints and improve computational efficiency.

C. Addressing Integration into Neural Networks

- **KST-Based Architectures**: Develop neural network architectures that inherently incorporate the KST framework, leveraging its theoretical strengths while mitigating practical implementation challenges.
- **Code Adaptation**: Adapt existing implementations from resources such as ChebyKAN [20] and PyKAN [21] to build efficient KST-based neural networks.

D. Diagram of Proposed Methodology

Figure 3 illustrates our proposed methodology. The diagram outlines the flow of our approach, starting from the theoretical foundation of KST, moving through the development of optimized inner and outer functions, and culminating in the integration into neural network architectures. Key components include:

- **Theoretical Framework**: Establishing a solid understanding of KST and identifying areas for enhancement.
- **Algorithm Development**: Designing algorithms for efficient computation of univariate functions φ_i and ψ_{ij}.
- **Optimization Techniques**: Implementing optimization methods to improve computational efficiency and accuracy.
- **Neural Network Integration**: Developing KST-based neural network architectures and testing their performance on high-dimensional data.

VIII. Experiment Setup

In this experiment, we compare three types of neural networks: KST with polynomial fitting, KST with spline fitting, and a traditional Multilayer Perceptron (MLP). An accuracy comparison of these three neural networks is conducted to evaluate how well they can learn a challenging mathematical function. The target function is composed of different components, including sine waves, exponential growth, and logarithmic patterns. At the end of the target function, the function y is multiplied by 1000. This scaling creates an "extreme condition" to test how each neural network handles tough numerical challenges.

The code for KST is adapted from [20] and [21]. The total number of parameters for each neural network is calculated to illustrate their relative complexity. KST-based networks, whether using polynomial or spline fitting, are designed to leverage the Kolmogorov Superposition Theorem for function approximation. KST with polynomial neural networks simplifies computation, but may compromise accuracy in capturing sharp transitions. KST with spline neural networks offer better adaptability to local variations but at a higher computational cost due to their increased parameter count. For comparison, the MLP serves as a traditional baseline with its layer-wise architecture and significant parameter requirements, illustrating its scalability and adaptability in high-dimensional spaces.

Each model is trained using the defined target function, and performance is evaluated using the Normalized Mean Absolute Percentage Error (NMAPE). This metric provides a normalized view of the error throughout the training process, offering a consistent basis for comparison. Training times for each network are recorded to assess computational efficiency.

IX. Implementation Plan

To execute the proposed methods, we have devised a detailed six-week implementation plan, denoted in Table II.

TABLE II
Implementation Timeline

Week	Activities
Week 1	Study KST in-depth and its implications in neural networks. Assign tasks among team members for literature review and initial coding setup.
Week 2	Develop a specific strategy for implementing KST, including selecting appropriate datasets and defining coding frameworks. Begin initial coding of basic KST models.
Week 3	Continue coding KST-based models. Validate the theoretical approach through simple test cases. Begin exploring optimization techniques for inner functions.
Week 4	Improve algorithms based on initial test results. Implement polynomial and spline-based functions and test on more complex problems. Start integrating dimension reduction methods.
Week 5	Integrate KST models into neural network architectures. Test and compare performance with traditional networks. Optimize computational efficiency using hybrid optimization strategies.
Week 6	Finalize the project—complete coding, analyze results, and prepare documentation. Begin writing the final report and prepare for presentations.

X. Performance Analysis and Trade-offs

A. Model Architecture Characteristics

We compare three distinct neural network architectures using multiple efficiency metrics. The architectures and their parameter counts are presented in Table III.

TABLE III
Model Architectural Complexity

Architecture	Configuration	Parameters
MLP	[1, 128, 128, 1]	16,897
KST + Polynomial	[1, 20, 20, 1]	3,960
KST + Spline	[1, 20, 20, 1]	6,600

The KST architectures achieve significant parameter reduction compared to the MLP baseline: 76.6% for KST + Polynomial and 60.9% for KST + Spline.

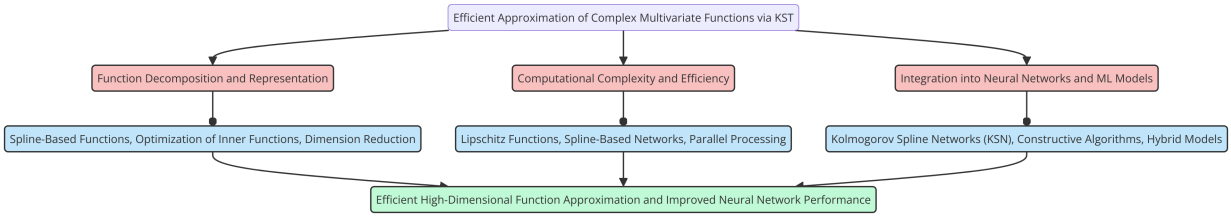

Fig. 3. Diagram of Proposed Methodology

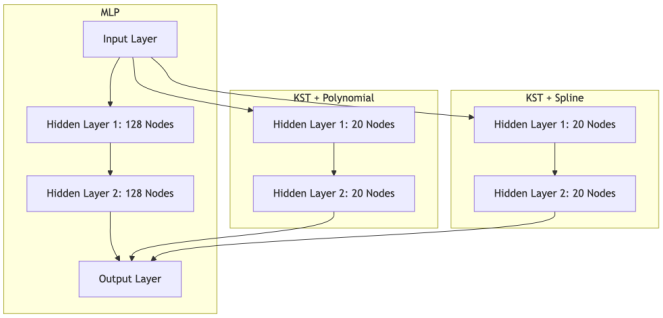

Fig. 4. Model structure comparison for the three neural network architectures: Multilayer Perceptron (MLP), KST with Polynomial fitting, and KST with Spline fitting. The MLP uses two dense layers with 128 nodes each, while KST-based models use smaller layers with 20 nodes, reducing computational complexity.

TABLE IV
TRAINING TIME PER EPOCH

Architecture	Time per Epoch (ms)
MLP	5.99
KST + Polynomial	14.70
KST + Spline	26.87

B. Computational Performance

The computational performance of the architectures is evaluated in terms of training time per epoch and total training duration. Results are shown in Tables IV and V.

While KST methods reduce parameter count, they require significantly more computation time per epoch, suggesting more complex operations per parameter.

TABLE V
TOTAL TRAINING DURATION

Architecture	Adam Phase (s)	L-BFGS Phase (s)
MLP	27.71	35.15
KST + Polynomial	54.58	99.78
KST + Spline	146.24	135.85

C. Error Metrics

The initial performance of the architectures is assessed using RMSE, MAE, and NMAPE. The results are summarized in Table VI.

TABLE VI
INITIAL PERFORMANCE ERROR METRICS

Metric	MLP	KST + Polynomial	KST + Spline
RMSE	7.111	53.693	24.846
MAE	4.361	32.031	12.975

The MLP achieves the best initial error metrics, while the KST methods require further improvement to approach MLP performance.

D. Hyperparameter Tuning

To improve accuracy, hyperparameter tuning was applied to the three neural networks. Adjustments were made to the network structure and learning rate. The spline function's order and the polynomial function's order were also optimized. Additionally, the KST function was modified to enhance performance. Detailed information on these changes is provided in Table VII below.

TABLE VII
NMAPE BEFORE AND AFTER HYPERPARAMETER TUNING

Architecture	Initial NMAPE	After H.P. Tuning NMAPE
MLP	7.81323612	1.38069351
KST + Polynomial	36.49292290	10.14589037
KST + Spline	9.94008929	4.11048754

After tuning, the MLP improved by 82.33%, KST + Polynomial by 72.21%, and KST + Spline by 58.65%. These results demonstrate that careful hyperparameter optimization can significantly enhance the accuracy of KST-based models.

E. NMAPE Before and After Optimization

Following hyperparameter tuning, we further employed a hybrid optimization strategy (Adam + L-BFGS). This step aimed to refine the model parameters for enhanced accuracy. Table X-E presents the Normalized Mean Absolute Percentage Error before and after this additional optimization phase.

TABLE VIII
NMAPE BEFORE AND AFTER OPTIMIZATION

Architecture	Initial NMAPE	After H.P. Tuning NMAPE
MLP	1.381	1.266
KST + Polynomial	10.146	4.678
KST + Spline	4.110	1.580

With these optimizations, the KST + Spline model nearly matches the MLP's final accuracy. The hybrid optimization strategy proves particularly effective for reducing NMAPE in KST-based architectures, albeit at a higher computational cost.

F. Optimization Effectiveness and Trade-offs

The hybrid optimization strategy demonstrates varying effectiveness. While KST methods achieve large improvements, they incur substantially more training time than the MLP. These findings highlight key trade-offs: KST-based methods can reach competitive accuracies given enough optimization effort, but at the expense of computational efficiency.

G. Limitations and Future Work

Despite the progress shown, several limitations remain. Metrics like FLOPS and memory utilization were not fully explored. Future work should profile computational performance more thoroughly, test generalization on diverse datasets, and investigate more advanced optimization strategies to further reduce training time while preserving accuracy.

XI. CONCLUSION

The Kolmogorov Superposition Theorem holds significant potential for advancing function approximation in high-dimensional spaces. By addressing the challenges of function decomposition, computational complexity, and integration into neural networks, this research seeks to unlock new possibilities in computational mathematics and machine learning. Our proposed methodology provides a structured approach to overcoming these challenges, and our experimental results demonstrate the practical viability of KST-based neural networks. We anticipate that our work will contribute valuable insight and tools to the field, fostering further research and development.

CODE AVAILABILITY

The source code used in this project can be accessed through our GitHub repository [1].

REFERENCES

[1] Ben Moore, Maxwell Lam, Matthew Robinson, and Justin Xie, "ML Project KAN Source Code," https://github.com/benhmoore/ML-Proj-KAN-Source, 2024.
[2] M. Nakamura, R. Mines, and V. Kreinovich, "Guaranteed Intervals for Kolmogorov's Theorem," *Kolmogorov Superposition Theorem*, 1993.
[3] M. Köppen, "On the Training of a Kolmogorov Network," *Neural Networks Based on KST*, 2002.
[4] D. Sprecher and S. Dräghici, "Space-Filling Curves and Kolmogorov Superposition-Based Neural Networks," *Mathematical and Computational Methods*, 2002.
[5] B. Igelnik and N. Parikh, "Kolmogorov's Spline Network," *Spline-Based Methods and Approximation*, 2003.
[6] J. Braun and M. Griebel, "On a Constructive Proof of Kolmogorov's Superposition Theorem," *Kolmogorov Superposition Theorem*, 2009.
[7] D. Sprecher, "Kolmogorov Superpositions: A New Computational Algorithm," *Kolmogorov Superposition Theorem*, 2013.
[8] V. Ismailov, "On the Approximation by Neural Networks with Bounded Number of Neurons in Hidden Layers," *Neural Networks Based on KST*, 2014.
[9] X. Liu, "Kolmogorov Superposition Theorem and Its Applications," *Kolmogorov Superposition Theorem*, 2015.
[10] V. Ismailov, "A Note on the Representation of Continuous Functions by Linear Superpositions," *Spline-Based Methods and Approximation*, 2015.
[11] J. A. Actor, "Computation for the Kolmogorov Superposition Theorem," *Kolmogorov Superposition Theorem*, 2018.
[12] K. Xu and E. F. Darve, "The Neural Network Approach to Inverse Problems in Differential Equations," *Mathematical and Computational Methods*, 2019.
[13] V. Ismailov, "A Three-Layer Neural Network Can Represent Any Multivariate Function," *Neural Networks Based on KST*, 2020.
[14] M. Lai and Z. Shen, "The Kolmogorov Superposition Theorem Can Break the Curse of Dimensionality," *Kolmogorov Superposition Theorem*, 2021.
[15] D. Fakhoury, E. Fakhoury, and H. Speleers, "ExSpliNet: An Interpretable and Expressive Spline-Based Neural Network," *Spline-Based Methods and Approximation*, 2022.
[16] A. Ismayilova and V. Ismailov, "On the Kolmogorov Neural Networks," *Neural Networks Based on KST*, 2023.
[17] J. He, "On the Optimal Expressive Power of ReLU DNNs and Its Application in Approximation with KST," *Neural Networks Based on KST*, 2023.
[18] M. Poluektov and A. Polar, "Construction of the Kolmogorov-Arnold Representation Using the Newton-Kaczmarz Method," *Mathematical and Computational Methods*, 2023.
[19] M.-J. Lai and Z. Shen, "The Optimal Rate for Linear KB-Splines and LKB-Splines Approximation of High-Dimensional Continuous Functions and Its Application," *Spline-Based Methods and Approximation*, 2024.
[20] SynodicMonth, "ChebyKAN," GitHub repository, [Online]. Available: https://github.com/SynodicMonth/ChebyKAN. [Accessed: Oct. 15, 2023].
[21] X. Lai, "PyKAN Examples," [Online]. Available: https://kindxiaoming.github.io/pykan/examples.html. [Accessed: Oct. 15, 2023].

XII. GROUP CONTRIBUTIONS

TABLE IX
GROUP 3 - KAN CONTRIBUTIONS

Name	Contribution (%)
Maxwell Lam	25%
Ben Moore	25%
Matthew Robinson	25%
Justin Xie	25%

XIII. LLM USAGE

Used ChatGPT for grammatical error detection, resolution, and to understand some mathematical theories.