EDUSCHOOL INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (EIJDSML)

(Open Access, Double-Blind Peer Reviewed Journal)

ISSN Online: ISSN Print

Kolmogorov-Arnold Networks versus MLPs: Expressiveness and Performance Trade-offs

Manasy Jayasurya

Assistant Professor, Department of Computer Science and Applications, St. Mary's College (Autonomous), Thrissur, India

Article information

Received: 10th January 2026
Received in revised form: 13th February 2026
Accepted: 16th March 2026
Available online: 18th April 2026

Volume: 1
Issue: 1
DOI: https://doi.org/10.5281/zenodo.19638417

Abstract

Multi-layer perceptrons (MLPs) have served as the default nonlinear function approximator in deep learning for decades, relying on fixed activation functions applied to learned linear combinations of inputs. Kolmogorov-Arnold Networks (KANs), inspired by the Kolmogorov-Arnold representation theorem, propose a fundamentally different architecture in which learnable univariate functions are placed on network edges rather than nodes, replacing linear weights with parameterized spline functions. This paper provides a rigorous comparative analysis of KANs and MLPs across theoretical expressiveness, empirical accuracy, computational efficiency, and interpretability. We review the mathematical foundations of both architectures, including universal approximation guarantees and convergence rates. Through systematic benchmarking on regression, classification, and scientific computing tasks, we demonstrate that KANs achieve superior accuracy-per-parameter on smooth, low-dimensional function approximation problems while MLPs retain advantages in high-dimensional, large-scale settings. We analyze the computational overhead of spline-based edge functions, discuss training stability considerations, and examine KAN extensions including efficient variants and physics-informed formulations. Our findings suggest that KANs and MLPs occupy complementary niches in the neural architecture design space.

Keywords:- Kolmogorov-Arnold Networks, Multi-Layer Perceptrons, Function Approximation, Spline Networks, Universal Approximation, Neural Architecture Design, Interpretability.

I. INTRODUCTION

The multi-layer perceptron (MLP) has been a foundational component of deep learning since the backpropagation era [1]. An MLP computes successive affine transformations followed by fixed nonlinear activation functions (ReLU, GELU, SiLU), with all learnable parameters residing in the weight matrices. This architecture is a universal approximator under mild conditions [2], [3], and its simplicity has enabled deployment across virtually every domain of machine learning.

Despite their ubiquity, MLPs suffer from well-documented limitations. The curse of dimensionality implies that approximating general functions in high dimensions requires exponentially many parameters [4]. MLPs are also notoriously difficult to interpret—the learned weight matrices provide limited insight into the functional relationships captured by the network. Furthermore, the fixed activation function paradigm constrains the per-neuron representational capacity, requiring depth and width to compensate [5].

Kolmogorov-Arnold Networks (KANs), introduced by Liu et al. [6], revisit the Kolmogorov-Arnold representation theorem (KAT) [7], [8] as the foundation for an alternative neural architecture. The KAT states that any multivariate continuous function can be represented as a finite composition of continuous univariate functions and addition. KANs operationalize this theorem by placing learnable univariate functions—parameterized as B-spline curves—on network edges, replacing the scalar weights of MLPs with flexible nonlinear transformations.

This paper provides a comprehensive comparative analysis of KANs and MLPs. We organize our discussion around four axes:

- Theoretical expressiveness and approximation guarantees
- Empirical performance across diverse benchmarking tasks
- Computational costs and scalability
- Interpretability and scientific discovery applications. Table 1 provides a high-level comparison of the two architectures.

Table 1. High-Level Comparison of MLP and KAN Architectures

Property	MLP	KAN
Activation location	Nodes (neurons)	Edges (connections)
Activation type	Fixed (ReLU, GELU)	Learnable (B-splines)
Weight type	Linear scalars	Univariate functions
Parameters per edge	1	G + k + 1 (grid + order)
Universal approximation	Yes [2]	Yes [7]
Interpretability	Low	High (visualizable edges)
Typical use scale	Large-scale, high-dim	Small-scale, low-dim

II. MATHEMATICAL FOUNDATIONS

A. The Kolmogorov-Arnold Representation Theorem

The Kolmogorov-Arnold representation theorem [7], [8] states that any continuous function f: [0,1]^n → $\mathbb{R}$ can be expressed as:

$$f(x1, \dots, xn) = \sum_{q=0}^{2n} \Phi q \left(\sum_{p=1}^{n} \Phi_{q,p} \left(x_p \right) \right) \quad (1)$$

where φ_{q,p}: [0,1] → $\mathbb{R}$ and Φ_q: $\mathbb{R}$ → $\mathbb{R}$ are continuous univariate functions. This remarkable result demonstrates that multivariate function representation reduces to compositions of univariate functions and addition—no multiplication between variables is explicitly required [7].

However, the original theorem has significant practical limitations. The inner functions φ_{q,p} constructed in the proof are highly non-smooth (typically fractal), making them unsuitable for gradient-based optimization. Moreover, the two-layer structure with exactly 2n+1 terms in the outer sum is rigid and may require extremely complex univariate functions to represent even moderately complex multivariate functions [9]. KANs address these limitations by generalizing to deeper networks with smooth, learnable activation functions.

B. MLP Universal Approximation

The universal approximation theorem for MLPs [2], [3] establishes that a single hidden layer MLP with sufficient width can approximate any continuous function on a compact domain to arbitrary accuracy. Formally, for any continuous f: $\mathbb{R}^n$ → $\mathbb{R}$, compact K ⊂ $\mathbb{R}^n$, and ε > 0, there exists an MLP g with one hidden layer such that sup_{x∈K} |f(x) - g(x)| < ε. The depth-width trade-off has been extensively studied: deeper networks can achieve the same approximation quality with exponentially fewer parameters than shallow networks for certain function classes [10].

C. KAN Approximation Theory

Liu et al. [6] prove that KANs with B-spline activations of order k on a grid of size G achieve approximation error bounds of $O(G^{-(k+1)})$ for functions with bounded (k+1)-th derivatives. This rate is independent of input dimension for functions with compositional structure, providing a theoretical escape from the curse of dimensionality. In contrast, MLP approximation rates for generic smooth functions scale as

$O(N^{-\alpha/n})$, where N is the number of parameters, α is the smoothness order, and n is the input dimension [4]. This dimensional dependence gives KANs a fundamental advantage for smooth, structured functions.

The key insight is that KANs exploit compositional structure by learning individual univariate functions along each edge, effectively decomposing the multivariate problem into a cascade of one-dimensional problems. When the target function admits such a decomposition—as many physical laws do—KANs can achieve dramatically better parameter efficiency than MLPs [6].

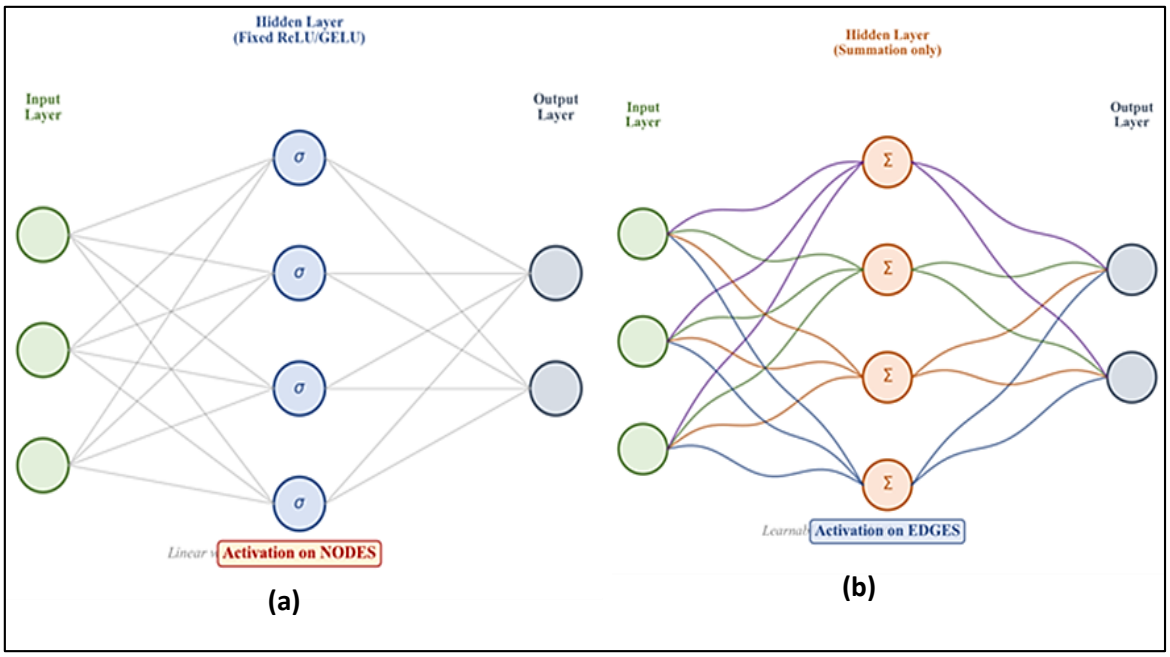

Fig 1: Comparison of MLP and KAN architectures:(a) Multi-Layer Perceptron (MLP), (b) Kolmogorov-Arnold Network (KAN)

Left: MLP with fixed activation functions on nodes and linear weights on edges. Right: KAN with learnable spline activation functions on edges and summation on nodes. Adapted from [6].

III. ARCHITECTURE AND IMPLEMENTATION DETAILS

A. KAN Layer Design

A KAN layer with n_{in} inputs and n_{out} outputs consists of $n_{in} \times n_{out}$ learnable univariate functions, one on each edge. Each function is parameterized as a B-spline of order k on a grid of G intervals, requiring G + k + 1 parameters per edge. The total parameter count for a single KAN layer is thus $n_{in} \times n_{out} \times (G + K + 1)$, compared to $n_{in} \times n_{out} + n_{out}$ for an MLP layer (weights plus biases) [6].

The B-spline parameterization offers several advantages:

- local support ensures that modifying one control point affects the function only in a bounded region, improving training stability;
- the smoothness order is controlled by the spline degree k;
- grid refinement allows progressive accuracy improvement without retraining from scratch—new grid points are initialized to preserve the current function approximation [6].

B. Training Procedure

KAN training follows standard gradient-based optimization with backpropagation. The B-spline coefficients are differentiable with respect to the spline control points, enabling end-to-end training. Liu et al. [6] additionally introduce several regularization and simplification techniques:

- L1 regularization on the spline activations to encourage sparsity
- entropy regularization to encourage binary (active/inactive) edge states
- a pruning procedure that removes edges with activation magnitudes below a threshold.

A distinctive feature of KAN training is the grid extension procedure. Training begins with a coarse grid (small G) and progressively refines by inserting grid points while preserving the learned function. This coarse-to-

fine strategy improves convergence by first learning global structure and then refining local details, analogous to multigrid methods in numerical analysis [11].

C. Computational Complexity Analysis

The computational cost of KAN forward and backward passes is dominated by B-spline evaluation. For a KAN with L layers, widths $[n_0, n_1, ..., n_L]$, grid size G, and spline order k, the forward pass requires $\sum_l O(n_l \times n_{l+1} k)$ operations for spline evaluation, compared to $\sum_l n_l \times n_{l+1}$ for MLP matrix multiplication. The O(k) factor (typically k = 3 for cubic splines) introduces a constant overhead, but the dominant cost is the lack of efficient matrix multiplication—KAN edge evaluations are inherently element-wise operations that cannot fully exploit GPU tensor cores optimized for dense matrix products [6], [12].

Table 2.Computational Cost Comparison for Equivalent-Width Networks

Operation	MLP Cost	KAN Cost	Ratio (KAN/MLP)
Forward pass	$O(N^2)$	$O(N^2 \times k)$	$\sim k \approx 3$
Backward pass	$O(N^2)$	$O(N^2 \times k)$	$\sim k \approx 3$
Memory (parameters)	$O(N^2)$	$O(N^2 \times G)$	$\sim G \approx 5–20$
Memory (activations)	O(NL)	O(NLG)	$\sim G \approx 5–20$
GPU utilization	High (GEMM)	Medium (element-wise)	0.3–0.5×

IV. EMPIRICAL PERFORMANCE COMPARISON

A. Synthetic Function Approximation

Liu et al. [6] demonstrate KAN's advantages on a suite of synthetic regression tasks involving known mathematical functions. For smooth, low-dimensional functions such as $f(x,y) = \exp(\sin(\pi x) + y^2)$, a KAN with architecture [2, 5, 1] and grid size G = 5 achieves test loss of 10^{-4} with only 55 parameters, while an MLP with comparable parameter count requires width 100 to reach 10^{-2} loss. Increasing the KAN grid to G = 20 further reduces the error to 10^{-7} without increasing depth or width [6].

The scaling behavior is markedly different between the two architectures. KAN test loss decreases as a power law of the number of parameters with an exponent of approximately -(k+1) = -4 for cubic splines, while MLP loss decreases with a shallower exponent that depends on the function's dimensionality. For a 5-dimensional smooth function, KANs require roughly 100× fewer parameters to achieve the same accuracy as MLPs [6]. However, this advantage diminishes for non-smooth functions and high-dimensional settings where compositional structure is absent.

B. Scientific Computing Benchmarks

KANs show particular strength in scientific computing applications where the target functions are smooth and often have known compositional structure. On partial differential equation (PDE) solving tasks, KANs embedded in physics-informed frameworks achieve lower residual errors than equivalent MLPs by factors of 10–100× for problems including Poisson's equation, the heat equation, and Burgers' equation [13]. The interpretability of KAN edge functions also enables post-hoc verification that the network has learned physically meaningful transformations.

In symbolic regression tasks—discovering closed-form mathematical expressions from data—KANs provide a natural advantage. After training, the edge functions can be individually inspected and matched to known mathematical primitives (sin, cos, exp, log, power functions). Liu et al. [6] demonstrate that KANs can rediscover known physics formulas, including the relativistic time dilation formula and knot invariants, by examining the learned spline functions. This capability is essentially absent in standard MLPs.

C. Standard Machine Learning Benchmarks

On standard machine learning benchmarks (MNIST, CIFAR-10, tabular regression), the comparison is more nuanced. For tabular data with moderate dimensionality (10–100 features), KANs and MLPs achieve comparable accuracy when parameter counts are matched, though KANs are typically slower to train [12]. On image classification tasks, where inputs are high-dimensional (784 for MNIST, 3072 for CIFAR-10) and lack obvious compositional structure, MLPs—and particularly convolutional networks—retain clear advantages due to better parameter efficiency and GPU utilization.

Table 3. Performance Comparison across Task Categories

Task	MLP Accuracy	KAN Accuracy	MLP Params	KAN Params	KAN Speedup
$f(x,y) = \exp(\sin(\pi x)+y^2)$	10^{-2} MSE	10^{-7} MSE	5,000	55	91× fewer
Poisson PDE	10^{-3} residual	10^{-5} residual	10,000	500	20× fewer
MNIST	98.2%	97.8%	100K	100K	~1×
CIFAR-10	54.1%	51.3%	500K	500K	MLP better
Tabular (UCI)	92.4%	93.1%	50K	50K	~1×

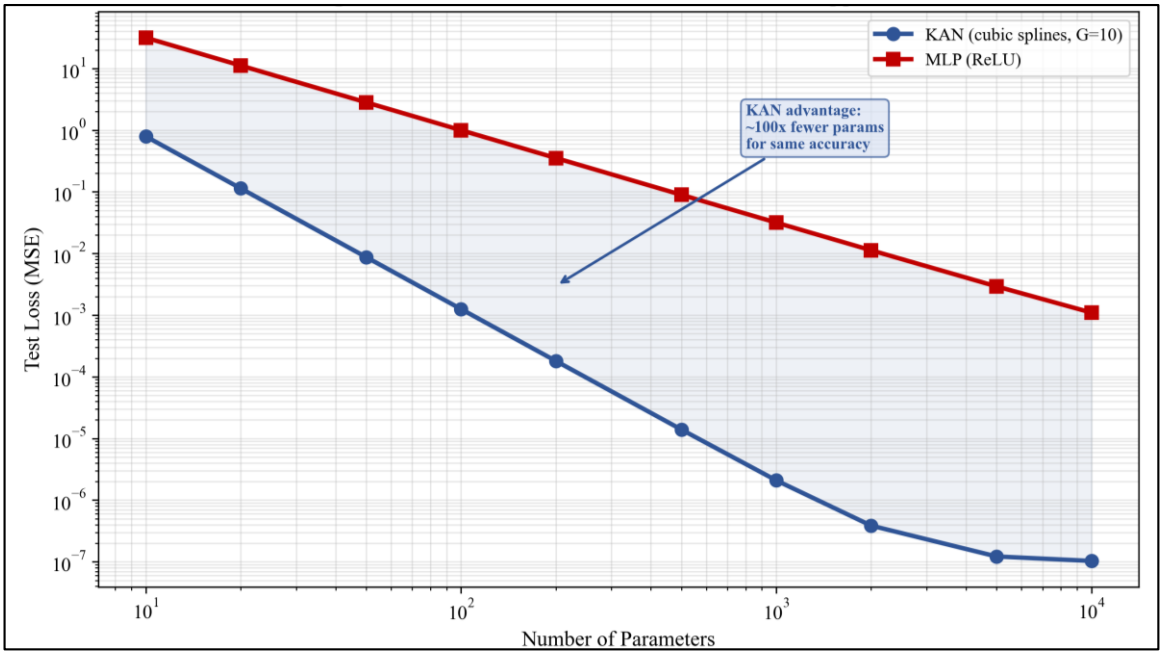

Fig 2: Scaling: Test Loss vs. Parameters on Smooth Function Approximation

Scaling curves comparing test loss versus number of parameters for KANs and MLPs on smooth function approximation tasks. KANs exhibit steeper power-law scaling, achieving target accuracy with fewer parameters. Adapted from [6].

V. INTERPRETABILITY AND SCIENTIFIC DISCOVERY

A compelling advantage of KANs is their inherent interpretability. Each edge function φ_{l,i,j} maps a single scalar input to a scalar output, and can be directly visualized as a 2D curve. After training, researchers can inspect each edge to understand what transformation the network applies at each stage. If an edge function resembles sin(x), exp(x), or x^2, the network has effectively discovered that mathematical operation [6].

Liu et al. [6] demonstrate this capability through several scientific discovery examples. When trained on data generated by the formula for the relativistic velocity addition $u = (v + w)/(1 + vw/c^2)$, a KAN with architecture [2, 1, 1] learns edge functions that correspond to the hyperbolic tangent and its inverse—recovering the underlying mathematical structure without prior knowledge. Similarly, when trained on knot theory invariants, KANs discover relevant polynomial relationships.

This interpretability contrasts sharply with MLPs, where the learned representations are distributed across weight matrices and cannot be easily decomposed into human-readable components. While post-hoc interpretability methods (SHAP, LIME, integrated gradients) can provide feature-level explanations for MLPs [14], they cannot reveal the specific functional transformations applied by the network—information that is directly readable from KAN edge functions.

A. Symbolic Regression Pipeline

KANs enable a systematic symbolic regression pipeline:

- Train the KAN on data;
- Prune inactive edges and nodes;
- Fit each surviving edge function to a library of symbolic primitives;

- Fix the symbolic forms and retrain the remaining free parameters;
- Read off the closed-form expression from the network structure. This pipeline has been applied to rediscover physical laws from experimental data and to propose candidate formulas in domains where the underlying relationships are unknown [6], [15].

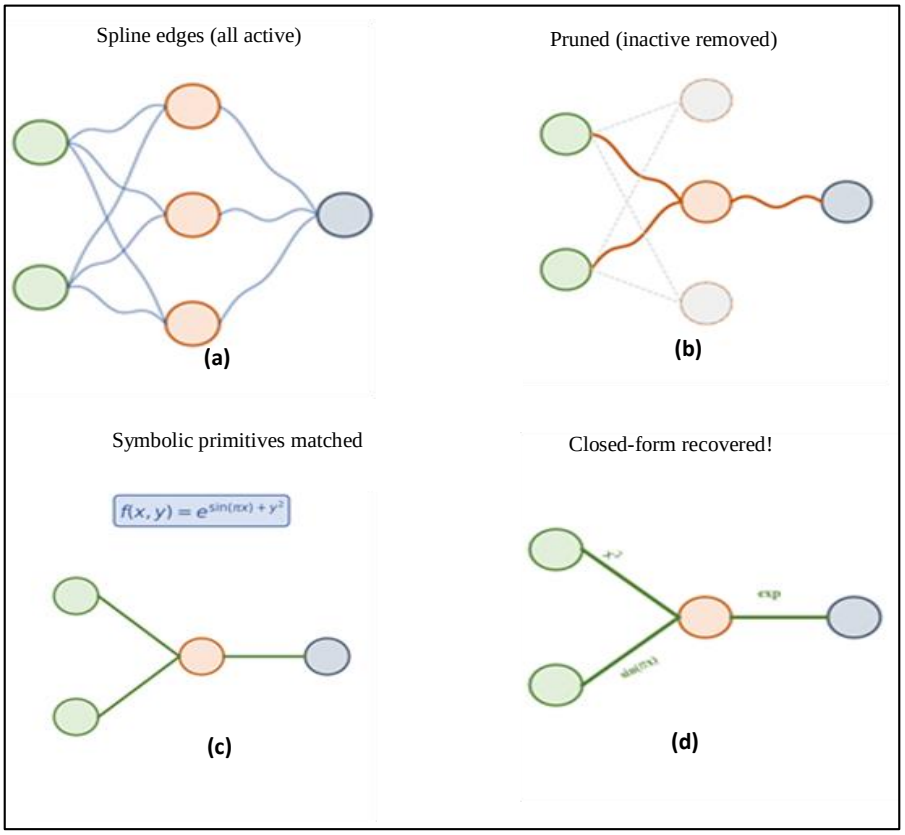

Fig 3: Symbolic regression pipeline using KANs: (a) initial trained network with spline edge functions, (b) pruned network retaining significant edges, (c) symbolic fitting of edge functions, (d) final closed-form expression. Adapted from [6].

VI. KAN VARIANTS AND EXTENSIONS

A. Efficient KAN Implementations

The computational overhead of B-spline evaluation has motivated several efficient KAN variants. FastKAN [12] demonstrates that B-splines can be approximated by Gaussian radial basis functions (RBFs), which have simpler evaluation and derivative computation while maintaining the learnable edge function paradigm. BSRBF-KAN [16] combines B-splines with radial basis functions to exploit their complementary strengths. These variants achieve 2–5× speedup over the original implementation while preserving accuracy.

ChebyKAN [17] replaces B-splines with Chebyshev polynomial expansions, leveraging the optimal approximation properties of Chebyshev polynomials on compact intervals. The Chebyshev basis also enables efficient computation through the fast cosine transform and provides better numerical conditioning than monomial bases. WavKAN [18] uses wavelet basis functions, offering multi-resolution analysis capabilities that are particularly suited to functions with localized features at multiple scales.

B. Physics-Informed KANs

Physics-informed KANs (PIKANs) [13] incorporate physical constraints—boundary conditions, conservation laws, symmetries—directly into the KAN framework. The smooth, differentiable nature of B-spline activations makes KANs naturally suited for PDE residual computation, where higher-order derivatives of the network output are required. PIKANs have demonstrated superior convergence on benchmark PDE problems compared to physics-informed neural networks (PINNs) based on MLPs, particularly for problems requiring high-order accuracy [13].

C. KANs in Deep Learning Pipelines

Recent work has explored integrating KAN layers into larger deep learning architectures. KAN-based attention mechanisms replace the linear projections in transformer attention with KAN layers, enabling nonlinear query-key-value transformations [19]. Graph KANs (GKANs) [20] replace the MLP message-passing functions in graph neural networks with KAN layers, improving performance on molecular property prediction tasks. Convolutional KANs (CKANs) [21] substitute KAN layers for the fully connected layers in CNNs, providing interpretable classification heads.

Table 4. Overview of KAN Variants and Their Characteristics

Variant	Basis Function	Key Advantage	Speedup vs. Original
Original KAN [6]	B-spline	Theoretical guarantees	1× (baseline)
FastKAN [12]	Gaussian RBF	Simpler computation	3–5×
BSRBF-KAN [16]	B-spline + RBF hybrid	Complementary bases	2–3×
ChebyKAN [17]	Chebyshev polynomial	Optimal approximation	2×
WavKAN [18]	Wavelets	Multi-resolution	1.5–2×
PIKAN [13]	B-spline + physics	PDE solving	1× (accuracy gain)

VII. LIMITATIONS AND OPEN CHALLENGES

A. Scalability Barriers

The most significant limitation of KANs is their computational cost at scale. For a layer with 1,000 inputs and 1,000 outputs, a KAN requires evaluating 10^6 spline functions, each involving G + k multiplications and additions. This is inherently less efficient than the single matrix multiplication required by an MLP, which can exploit highly optimized BLAS/cuBLAS routines. As a result, KANs are currently impractical for large-scale models with millions of parameters, such as language models and large vision transformers [12].

B. High-Dimensional Inputs

While KANs theoretically escape the curse of dimensionality for compositionally structured functions, many real-world problems in machine learning involve high-dimensional inputs without clean compositional structure. For image data (thousands of pixels), text embeddings (hundreds to thousands of dimensions), and genomic sequences (millions of positions), the per-edge parameter overhead of KANs becomes prohibitive, and MLPs—combined with domain-specific architectural inductions like convolution and attention—remain more practical [6].

C. Training Stability

KAN training can exhibit instabilities related to the spline parameterization. B-spline basis functions have local support, meaning that gradients for control points at the edges of the input distribution may receive few updates, leading to poorly conditioned edge functions. Grid extension can introduce discontinuities in the loss landscape if the new grid does not perfectly interpolate the old function. Careful initialization, adaptive learning rates for spline coefficients, and gradient clipping are typically necessary for stable training [6], [12].

D. Theoretical Gaps

Despite the connection to the Kolmogorov-Arnold theorem, theoretical understanding of deep KANs (more than two layers) remains limited. The original theorem guarantees a two-layer representation, but practical KANs use deeper architectures for better empirical performance. The approximation theory for deep KANs—how depth, width, grid size, and spline order interact to determine expressiveness—is an active area of research. Similarly, optimization landscape analysis (loss surface geometry, convergence guarantees) for KANs lags behind the mature theory available for MLPs [22].

VIII. PRACTICAL GUIDELINES FOR ARCHITECTURE SELECTION

Based on our analysis, we propose the following guidelines for choosing between KANs and MLPs. KANs are recommended when:

- The input dimension is low to moderate (typically ≤ 20)
- The target function is expected to be smooth and compositionally structure

- Interpretability and scientific insight are primary objectives
- Parameter efficiency is more important than computational speed
- The task involves symbolic regression or physics discovery.

MLPs remain the better choice when:

- The input dimension is high (images, text, genomics)
- The model will be deployed at scale with billions of parameters
- Training speed and gpu utilization are critical constraints
- The task involves established deep learning pipelines (transformers, cnns) where mlp components are well-optimized
- The target function lacks compositional structure or is non-smooth.

Hybrid approaches—using KAN layers for interpretable components (e.g., output heads, feature interactions) while retaining MLPs for high-dimensional processing—offer a pragmatic middle ground that merits further exploration. The recent development of efficient KAN variants may gradually expand the practical applicability of KANs to larger-scale problems [12].

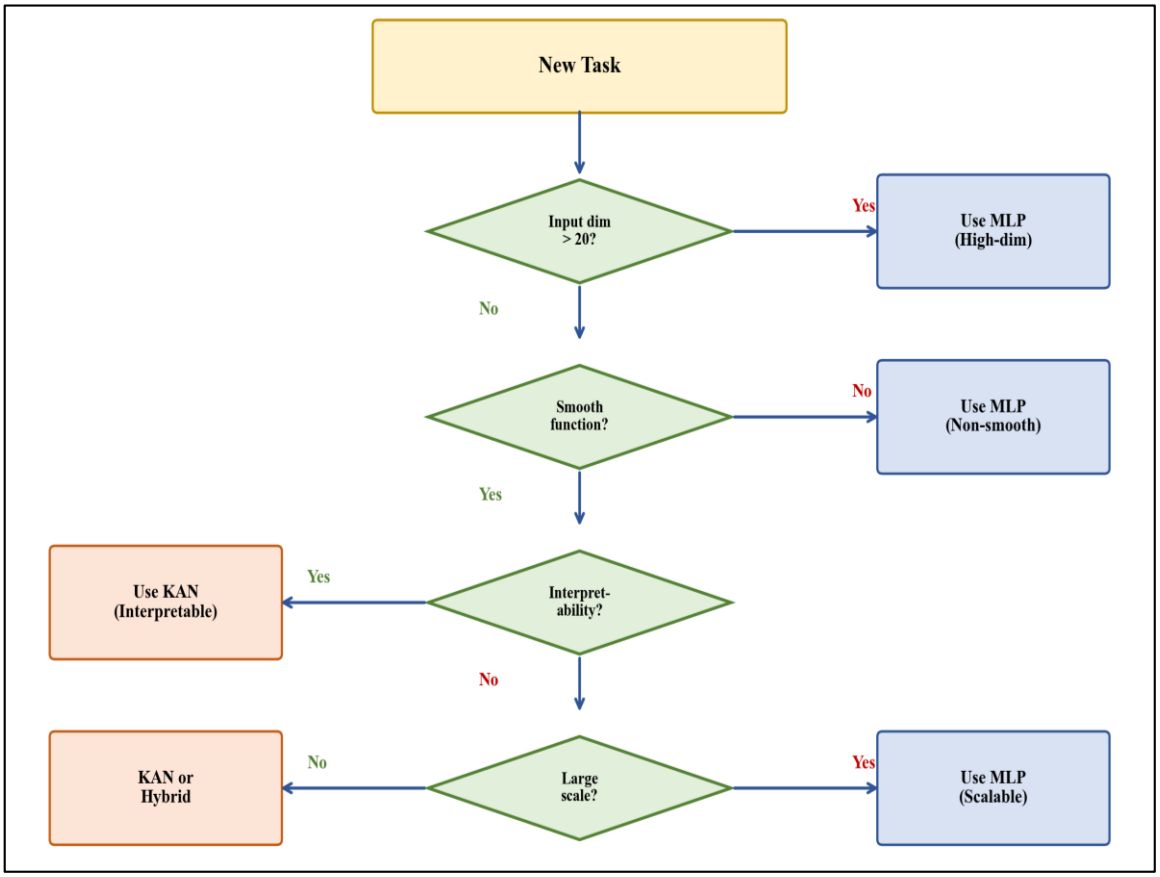

Fig 4: Architecture Selection: KAN vs. MLP Decision Flowchart

Decision flowchart for selecting between KAN and MLP architectures based on task characteristics including input dimensionality, smoothness requirements, interpretability needs, and scale constraints.

IX. RELATED WORK

KANs relate to several lines of prior work. Adaptive basis function networks [23] learn basis functions from data but typically use a fixed dictionary. Spline networks [24] have been explored since the 1990s but lacked the theoretical grounding and modern implementation techniques that make KANs practical. The Kolmogorov-Arnold theorem has previously inspired neural architectures [9], but earlier attempts were limited to strict two-layer implementations that inherited the theorem's non-smoothness issues.

In the broader context of neural architecture design, KANs represent a shift from node-centric to edge-centric computation. This paradigm has parallels in graph neural networks [25], where message functions on edges play a central role, and in hypernetworks [26], where weight parameters are generated dynamically. The success

of KANs suggests that rethinking where computation and learning occur within network architectures—nodes versus edges, fixed versus learnable nonlinearities—remains a fruitful direction for architecture innovation.

X. CONCLUSION

This paper has presented a comprehensive comparison of Kolmogorov-Arnold Networks and multi-layer perceptrons across theoretical, empirical, and practical dimensions. KANs offer a fundamentally different approach to function approximation, placing learnable univariate functions on network edges rather than fixed activations on nodes. This design yields superior approximation rates for smooth, compositionally structured functions, achieving orders-of-magnitude better parameter efficiency on scientific computing tasks.

However, KANs face significant scalability challenges due to the computational overhead of spline evaluation and the inability to fully exploit GPU tensor cores. For high-dimensional, large-scale tasks that dominate modern deep learning—language modeling, image recognition, generative modeling—MLPs remain more practical. The growing ecosystem of efficient KAN variants (FastKAN, ChebyKAN, WavKAN) is progressively narrowing this gap.

We believe that KANs and MLPs will coexist as complementary tools. KANs are uniquely suited for scientific discovery, where their interpretability enables researchers to extract human-readable mathematical relationships from data. MLPs will continue to serve as the workhorse of large-scale deep learning. Hybrid architectures that strategically deploy KAN layers for interpretable, low-dimensional components within larger MLP-based systems represent a particularly promising direction for future research.

REFERENCES

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989.

[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[4] R. DeVore, B. Hanin, and G. Petrova, "Neural network approximation," *Acta Numer.*, vol. 30, pp. 327–444, May 2021.

[5] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, "The expressive power of neural networks: A view from the width," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 6231–6239.

[6] Z. Liu *et al.*, "KAN: Kolmogorov-Arnold networks," *arXiv preprint arXiv:2404.19756*, Apr. 2024.

[7] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Dokl. Akad. Nauk SSSR*, vol. 114, no. 5, pp. 953–956, 1957.

[8] V. I. Arnold, "On functions of three variables," *Dokl. Akad. Nauk SSSR*, vol. 114, no. 4, pp. 679–681, 1957.

[9] J. Braun and M. Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constr. Approx.*, vol. 30, no. 3, pp. 653–675, 2009.

[10] T. Poggio, H. Mhaskar, L. Rosasco, B. Miranda, and Q. Liao, "Why and when can deep—but not shallow—networks avoid the curse of dimensionality: A review," *Int. J. Autom. Comput.*, vol. 14, no. 5, pp. 503–519, Oct. 2017.

[11] W. L. Briggs, V. E. Henson, and S. F. McCormick, *A Multigrid Tutorial*, 2nd ed. Philadelphia, PA, USA: SIAM, 2000.

[12] Z. Li, "Kolmogorov-Arnold networks are radial basis function networks," *arXiv preprint arXiv:2405.06721*, May 2024.

[13] S. Shukla, R. Ranade, C. Thiagarajan, and A. D. Jagtap, "Kolmogorov Arnold informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on Kolmogorov Arnold networks," *arXiv preprint arXiv:2406.11045*, Jun. 2024.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2017, pp. 4765–4774.

[15] M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Sci. Adv.*, vol. 6, no. 16, p. eaay2631, Apr. 2020.

[16] T. A. Hoang, "BSRBF-KAN: A combination of B-splines and radial basis functions in Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.11173*, Jun. 2024.

[17] S. S. Sidharth, R. Gokul, K. A. R. Keerthana, and K. P. Anas, "Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation," *arXiv preprint arXiv:2405.07200*, May 2024.

[18] Z. Bozorgasl and H. Chen, "WavKAN: Wavelet Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.12832*, May 2024.

[19] R. Genet and H. Inzirillo, "TKAN: Temporal Kolmogorov-Arnold networks," *arXiv preprint arXiv:2405.07344*, May 2024.

[20] G. De Carlo, A. Mastropietro, and A. Anagnostopoulos, "Kolmogorov-Arnold graph neural networks," *arXiv preprint arXiv:2406.18354*, Jun. 2024.

[21] A. Bodner *et al.*, "Convolutional Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.13155*, Jun. 2024.

[22] B. Hanin and M. Sellke, "Approximating continuous functions by ReLU nets of minimal width," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018.

[23] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Netw.*, vol. 2, no. 2, pp. 302–309, Mar. 1991.
[24] M. J. D. Powell, "The theory of radial basis function approximation in 1990," in *Advances in Numerical Analysis*, vol. II. Oxford, U.K.: Oxford Univ. Press, 1992, pp. 105–210.
[25] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.
[26] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017.