Scalable and Interpretable Function-Based Architectures: A Survey of Kolmogorov–Arnold Networks

Jessica Beatrize[1]

[1]Universidade dos Açores
jessica.beatrize@uac.pt

Abstract. Kolmogorov–Arnold Networks (KANs) are an emerging class of neural architectures that replace traditional linear transformations and activation functions with learnable univariate function compositions, inspired by the Kolmogorov–Arnold representation theorem. By leveraging this foundational result in multivariate function decomposition, KANs offer a highly expressive yet interpretable alternative to standard multilayer perceptrons (MLPs) and convolutional or attention-based architectures. Recent advancements have shown that KANs can match or exceed the performance of conventional models across a range of tasks—particularly those characterized by smoothness, structure, or low-dimensional manifolds—while providing inherent advantages in data efficiency, robustness, and variable-level interpretability.

In this survey, we provide an extensive overview of the KAN framework, tracing its theoretical underpinnings, architectural variants, and practical implementations. We examine the scalability of KANs in terms of model capacity, optimization efficiency, and hardware feasibility, and we benchmark their performance on synthetic, tabular, visual, and scientific datasets. We also analyze the unique trade-offs involved in deploying KANs at scale, including computational overhead and training dynamics. Finally, we identify key open research directions—including generalization theory, hybrid model integration, compiler support, and domain-specific applications—that will shape the future of KAN research. This survey aims to serve both as a comprehensive introduction and a roadmap for researchers and practitioners interested in functional architectures and the next generation of interpretable machine learning systems.

Keywords: Kolmogorov–Arnold Networks, functional architectures, spline networks, interpretability, deep learning, representation learning, scalable neural networks, scientific machine learning

1 Introduction

The recent resurgence of interest in function-approximating neural networks has brought attention to a theoretically rich and architecturally distinct class of models: Kolmogorov–Arnold Networks (KANs). These architectures take inspiration

from the Kolmogorov–Arnold representation theorem, which asserts that any multivariate continuous function can be expressed as a finite sum of univariate continuous functions composed with continuous functions of a single variable. KANs leverage this theorem to construct networks where instead of conventional weight matrices and dot products, the core computational units are learnable functions, often instantiated as splines or other smooth interpolants. This leads to a paradigm shift in how data representations are learned, especially in high-dimensional spaces. Despite their elegant theoretical underpinnings and demonstrated expressiveness in controlled environments, the adoption of KANs in large-scale applications has been historically constrained by computational inefficiencies, memory overhead, and a lack of standardized implementations [1]. These limitations are particularly salient when compared to the well-optimized tensor algebra pipelines underpinning MLPs, CNNs, and Transformers. However, recent advances have begun to bridge this gap [2]. Innovations in spline parameterization, grid-based approximation schemes, GPU-accelerated functional operators, and hybrid neural architectures have made it possible to train KANs on datasets of increasing size and complexity [3]. Scalability and efficiency in KANs encompass several dimensions: (1) **Computational Scalability**, where the goal is to reduce the algorithmic and hardware burden of evaluating and differentiating spline functions; (2) **Statistical Scalability**, which concerns the generalization performance and sample complexity of KANs as model capacity increases; and (3) **System Scalability**, where attention is paid to distributed training, memory access patterns, and interoperability with mainstream frameworks such as PyTorch, JAX, and TensorFlow [4]. This survey aims to systematically chart the landscape of scalable and efficient KANs by synthesizing knowledge across multiple research threads. We begin by introducing the mathematical foundations of the Kolmogorov–Arnold representation and its interpretations in neural function approximation. We then present a taxonomy of architectural variants of KANs, including spline-based KANs, Fourier-inspired functional networks, and hybrid models that combine functional layers with traditional deep networks. Emphasis is placed on how these architectures evolve to meet the dual demands of expressivity and computational feasibility. In evaluating efficiency, we dissect algorithmic improvements such as fast spline evaluation, low-rank approximations, adaptive grid spacing, and sparse functional connections. Moreover, we discuss memory-efficient backpropagation techniques, batching strategies for irregular computation graphs, and integration of automatic differentiation systems tailored for functional primitives [5]. To facilitate reproducibility and real-world deployment, we explore open-source implementations, benchmark suites, and hardware-accelerated variants. Special attention is given to the application domains where KANs exhibit promising performance, including but not limited to high-dimensional regression, physics-informed learning, and interpretable modeling. Finally, we outline open challenges and future directions for research, including the quest for universal approximators with bounded complexity, theoretical convergence guarantees for gradient-based training in function spaces, and principled methods for architecture search in KANs. By offering this extensive

and detailed overview, we hope to lower the barrier to entry for researchers and practitioners interested in the promise of KANs, while also providing a structured roadmap for future explorations in the scalable, efficient implementation of functional neural networks [6].

2 Theoretical Foundations

At the heart of Kolmogorov–Arnold Networks lies a deep mathematical insight: the Kolmogorov–Arnold representation theorem, a result in real analysis that provides a constructive way to represent multivariate functions using only univariate functions and addition [7]. Formally, the theorem asserts that any continuous function $f : [0, 1]^n \to \mathbb{R}$ can be represented as:

$$f(x_1, \ldots, x_n) = \sum_{q=0}^{2n} \phi_q \left(\sum_{p=1}^{n} \psi_{q,p}(x_p) \right), \tag{1}$$

where ϕ_q and $\psi_{q,p}$ are continuous functions of a single real variable, independent of the specific function f being represented. This remarkable result was independently established by Andrey Kolmogorov and Vladimir Arnold in the mid-20th century, and it has since become a foundational theorem in nonlinear approximation theory.

2.1 Implications for Neural Network Design

Traditional feedforward neural networks, particularly multilayer perceptrons (MLPs), rely on the universal approximation theorem to justify their representational capacity. While the universal approximation theorem guarantees that an MLP with sufficient width and a non-polynomial activation function can approximate any continuous function on a compact set, it provides little guidance on how to structure the network for optimal efficiency or interpretability[8–10]. KANs, on the other hand, are inspired by the explicit constructive nature of the Kolmogorov–Arnold representation [11]. Instead of composing affine transformations with nonlinear activation functions, KANs construct each layer as a sum of learnable univariate functions applied to linear or nonlinear combinations of the input [12]. This compositional structure adheres more closely to the function decomposition described in the theorem and enables a more direct encoding of functional interactions.

2.2 Representation Power and Function Spaces

KANs effectively operate in a space of functions that are inherently more expressive and adaptable than those defined by fixed parametric forms. When the constituent functions ϕ_q and $\psi_{q,p}$ are parameterized using flexible basis functions such as B-splines, Gaussian kernels, or Fourier series, the network gains the ability to approximate a wide variety of smooth, discontinuous, and high-frequency

functions with relatively shallow architectures [13]. Theoretical investigations into the approximation properties of KANs reveal that they can achieve exponential convergence rates in certain function spaces, particularly Sobolev and Hölder spaces, depending on the smoothness of the target function and the spline degree. These rates often surpass those of MLPs with comparable parameter counts, especially when the input space exhibits low intrinsic dimensionality or strong variable separability [14].

2.3 Smoothness, Interpretability, and Inductive Biases

The use of spline-based or other continuous function parameterizations introduces a strong inductive bias toward smoothness [15]. This has dual benefits: first, it improves generalization by penalizing rapid oscillations and overfitting; second, it enhances interpretability by allowing one to directly visualize the learned univariate transformations. Moreover, unlike deep neural networks whose internal representations are often opaque, the functional composition in KANs can be analyzed to uncover variable interactions, functional symmetries, and localized behaviors. This transparency opens the door to applications in scientific domains where model interpretability is as critical as performance [16].

2.4 Limitations of the Original Theorem

While the Kolmogorov–Arnold theorem is elegant and general, it is not without caveats. The original representation is non-constructive with respect to the specific functions ϕ_q and $\psi_{q,p}$, and these functions are typically highly nonlinear and not smooth. Moreover, the dimensionality of the representation (specifically, the use of $2n+1$ outer functions) may be prohibitively large in practice for high-dimensional problems [17]. Modern KAN implementations relax the rigidity of the original theorem in favor of parameter efficiency and numerical tractability. Instead of using a fixed number of outer functions, architectures are designed to learn the required depth and width through optimization. Additionally, the smoothness and differentiability of the learned functions are explicitly enforced through the choice of parameterization and regularization.

2.5 Connection to Other Theoretical Frameworks

KANs can also be seen as a bridge between neural networks and classical function approximation techniques such as radial basis function networks (RBFNs), kernel machines, and spline interpolation methods. In contrast to kernel methods, which use fixed basis functions and optimize over coefficients, KANs learn both the shape and placement of basis functions dynamically, yielding more flexible models with localized adaptation [18]. Furthermore, the functional formulation of KANs has close ties to operator learning and neural operators, where the focus shifts from learning mappings between finite-dimensional spaces to approximating mappings between function spaces. This viewpoint positions KANs

as potentially powerful tools in learning partial differential equations, modeling dynamical systems, and solving inverse problems in physics-informed machine learning.

2.6 Summary

The theoretical foundations of KANs provide a rich and rigorous justification for their use as a class of function-approximating neural architectures. By grounding their design in the Kolmogorov–Arnold representation theorem, KANs inherit a natural framework for compositional learning, interpretability, and expressivity. The ongoing challenge lies in adapting these theoretical insights into scalable, efficient, and robust implementations suitable for modern machine learning workloads. The next section explores the architectural innovations that make this possible [19].

3 Architectural Variants of KANs

While the foundational idea behind Kolmogorov–Arnold Networks (KANs) remains rooted in function decomposition and univariate function learning, a number of architectural variants have emerged to adapt these principles to real-world tasks, address computational bottlenecks, and enhance model flexibility [20]. In this section, we provide a detailed taxonomy of these variants, categorized by function parameterization, layer design, composition schemes, and hybrid architectures [21].

3.1 Spline-Based KANs

One of the most common implementations of KANs involves parameterizing the univariate functions using spline basis functions. B-splines, in particular, are favored for their numerical stability, local support, and smoothness properties. A univariate function $\phi(x)$ is expressed as a linear combination of basis splines:

$$\phi(x) = \sum_{i=1}^{K} c_i B_i(x), \tag{2}$$

where $B_i(x)$ denotes the i-th basis spline and c_i are the learnable coefficients [22]. This formulation allows for efficient forward and backward computation, especially when precompiled with libraries optimized for spline evaluation [23]. Spline-based KANs can control the degree of smoothness via knot spacing, spline order, and regularization on second or higher derivatives. This is especially useful in domains requiring physically plausible function representations or where model interpretability is paramount.

3.2 Fourier and Kernel-Based Function Layers

Beyond splines, other function parameterizations have been explored. Fourier-based KANs utilize sinusoidal basis functions to capture periodicity and global structure:

$$\phi(x) = \sum_{k=1}^{K} a_k \sin(\omega_k x + \theta_k), \tag{3}$$

where ω_k, θ_k, and a_k are learnable [24]. This representation is particularly well-suited for problems with inherent periodic behavior, such as time series modeling or wave simulation [25]. Alternatively, kernel-based KANs adopt radial or polynomial basis functions with trainable centers and bandwidths [26]. These methods offer strong local approximation capabilities and are beneficial for modeling sharp transitions or localized features.

3.3 Adaptive Grid-Based Approximations

To improve parameter efficiency, some KAN variants dynamically allocate basis functions using adaptive grid refinement [27]. Instead of a fixed number of basis functions per input dimension, the network allocates more basis functions in regions where the target function exhibits higher complexity [28]. This is akin to adaptive mesh refinement in numerical analysis and significantly improves performance on tasks with non-uniform function behavior [29]. These models often employ an auxiliary learning mechanism to guide grid refinement, based on curvature estimates, gradient magnitude, or information gain [30].

3.4 Functional Composition Layers

A key architectural decision in KANs is the design of the composition step—how univariate functions are combined to form higher-order mappings [31]. Several strategies exist:

- **Additive Composition:** Each layer computes a weighted sum of univariate outputs, often followed by another univariate function. This approach is closest to the original Kolmogorov–Arnold formulation [32].
- **Nested Composition:** Univariate functions are composed hierarchically, resulting in deep function trees [33]. This enhances expressivity but increases training complexity.
- **Attention-Based Composition:** Recent works explore data-dependent compositions where the contribution of each univariate function is modulated via an attention mechanism. This allows the network to learn context-sensitive function assemblies.

3.5 Hybrid KAN Architectures

KANs can be integrated with other neural paradigms to form hybrid models that benefit from both functional and parametric learning:

- **KAN-MLP Hybrids:** These models interleave KAN layers with traditional dense layers, using the former to extract expressive features and the latter to enforce global coordination or decision-making [34].
- **KAN-Convolutional Hybrids:** In computer vision tasks, KAN layers can replace or augment convolutional blocks to learn interpretable transformations of image features.
- **KAN-Transformer Hybrids:** Attention layers in Transformers can be fused with function-based embeddings to enhance positional encoding and variable-specific modeling.

3.6 Regularization and Stabilization Techniques

Given their expressive power, KANs are prone to overfitting if not carefully regularized [35]. Several regularization strategies have been proposed:

- **Smoothness Regularization:** Penalizing high-order derivatives of univariate functions encourages smoother approximations [36].
- **Function Norm Penalties:** Constraints on L^2 or Sobolev norms of univariate functions promote boundedness and control [37].
- **Weight Sparsity:** Encouraging sparse connections between input features and univariate functions leads to more interpretable models and better generalization [38].

3.7 Scalability-Oriented Variants

Architectural variants also exist with explicit design for scalability:

- **Block-Diagonal Decompositions:** The input space is partitioned into blocks, and each block is modeled independently via smaller KAN submodules [39].
- **Low-Rank Function Approximations:** Inspired by tensor decompositions, the function layers are parameterized to exploit low-rank structures, reducing both memory and computation [40].
- **Sparse Function Routing:** Instead of evaluating all univariate functions for every input, routing mechanisms selectively activate a subset of functions based on input characteristics [41].

3.8 Summary

The landscape of KAN architectures is rapidly evolving [42]. From spline-based approximators to hybrid networks incorporating attention and convolutions, these variants reflect a growing interest in functional inductive biases as a path toward efficient and interpretable machine learning [43]. Each architectural choice—be it the basis function, composition strategy, or regularization technique—introduces unique trade-offs between expressivity, scalability, and implementation complexity [44]. In the next section, we turn to a detailed examination of these trade-offs through the lens of scalability and efficiency.

4 Scalability Dimensions of KANs

Scalability is a critical requirement for any modern machine learning architecture, especially when models are expected to handle large datasets, high-dimensional inputs, and deployment on heterogeneous hardware environments. In the case of Kolmogorov–Arnold Networks (KANs), the transition from theoretical elegance to practical utility hinges on addressing a complex interplay of factors that influence scalability [45]. This section decomposes scalability into three principal dimensions—computational, statistical, and system-level—and explores methods to improve efficiency across each axis [46].

4.1 Computational Scalability

Computational scalability refers to the ability of a model to efficiently utilize computational resources as problem size grows [47]. For KANs, several computational challenges arise:

- **Spline Evaluation Overhead:** Unlike MLPs where operations are dominated by matrix multiplication, KANs require the evaluation of spline or kernel functions, which are non-trivial and irregular [48]. Each univariate function may involve local interpolation, lookup, or basis expansion, complicating vectorization.
- **Gradient Computation:** Backpropagation through functional layers requires computing gradients with respect to both the function input and the parameters defining the function (e.g., spline coefficients) [49]. Efficient automatic differentiation libraries, such as JAX and PyTorch's custom autograd, are essential but may still introduce latency [50].
- **Non-uniform Execution Graphs:** The irregular structure of function routing and dynamic composition in advanced KAN variants leads to non-uniform execution graphs that are harder to batch and parallelize effectively compared to standard tensor flows.

Acceleration Strategies To address these challenges, various computational strategies have been proposed:

- **Precompiled Spline Kernels:** Libraries like `torch-spline-conv` and custom CUDA implementations of B-splines enable fast forward and backward passes for common basis functions [51].
- **Function Caching and Lookup Tables:** For repeated inputs or structured inputs (e.g., grids), caching function evaluations or using LUTs can significantly reduce redundant computation.
- **Just-in-Time (JIT) Compilation:** Using JIT tools such as XLA (in JAX) or TorchScript (in PyTorch) can reduce runtime overhead and enable ahead-of-time graph optimization [52].
- **Operator Fusion and Code Generation:** Automatic fusion of function evaluations into a single GPU kernel reduces memory traffic and improves throughput, particularly for nested or composed functional layers [53].

4.2 Statistical Scalability

Statistical scalability involves maintaining or improving generalization performance as model capacity and dataset size grow. KANs possess both advantages and potential risks in this dimension:

- **Function Overfitting:** With highly expressive univariate functions, KANs risk overfitting even small variations in the data. This is particularly problematic in low-data regimes [54].
- **Sample Complexity:** While KANs can achieve high expressivity with fewer parameters compared to MLPs, they may still require careful function regularization to achieve optimal sample complexity [55].
- **Inductive Bias and Smoothness Priors:** The functional nature of KANs inherently introduces smoothness priors, which can be a double-edged sword—beneficial in structured domains, but potentially restrictive in high-variance settings [56].

Mitigation Techniques To improve generalization and robustness:

- **Regularization of Function Complexity:** Penalizing curvature or derivatives of the learned functions (e.g., via L_2 or Sobolev norms) prevents overfitting to noise.
- **Function Dropout and Noise Injection:** Introducing stochasticity into function outputs during training can improve robustness and serve as a form of data augmentation [57].
- **Adaptive Function Pruning:** Dynamically identifying and removing underutilized univariate functions reduces model complexity without compromising accuracy [58].

4.3 System-Level Scalability

System scalability addresses hardware efficiency, memory consumption, and ease of deployment. For KANs, this dimension presents unique challenges due to the use of non-standard operators and custom function layers.

Memory and Latency Considerations KANs often have higher memory footprints compared to MLPs due to:

- Storage of basis functions and coefficients [59].
- Lack of optimized kernel fusions in mainstream libraries [60].
- Irregular memory access patterns from dynamic function routing [61].

Memory usage is particularly sensitive in large-scale deployments (e.g., edge devices, GPUs with limited VRAM), where models must fit into constrained environments.

Parallelization and Batching The non-uniform structure of KANs poses difficulties for traditional parallelization. To address this:

- **Grouped Function Evaluation:** Similar univariate functions across different inputs can be grouped and evaluated in batches, amortizing computation [62].
- **Input Reordering:** Sorting or binning inputs based on activation patterns can improve cache efficiency and vectorization.
- **Pipeline Parallelism:** In large models, functional layers can be split across devices with pipelined execution to maximize hardware utilization [63].

Framework Compatibility and Deployment KANs require extending or customizing existing deep learning frameworks [64]. Not all operations have built-in support in platforms like ONNX or TensorRT, limiting out-of-the-box deployment [65]. Promising approaches include:

- **Custom Operators in TensorFlow/XLA and PyTorch:** Using plugin mechanisms or extending IR compilers enables integration of KAN operations into production inference pipelines.
- **WebAssembly and Edge Inference:** For deployment on edge devices, KAN components can be compiled into WebAssembly or optimized C++ kernels for lightweight inference.

4.4 Summary

Scalability in KANs is a multi-faceted challenge encompassing computational efficiency, generalization behavior, and deployment feasibility. Addressing these

dimensions requires coordinated advances in function approximation theory, numerical optimization, software tooling, and hardware acceleration. While traditional neural networks benefit from decades of engineering refinement, KANs are still in their nascency—but their unique functional inductive biases offer a compelling frontier for building scalable, interpretable, and powerful models [66]. In the following section, we present empirical comparisons of KANs with other architectures across benchmarks to quantify these trade-offs in practice [67].

5 Empirical Benchmarking and Performance Comparison

To fully understand the practical implications of Kolmogorov–Arnold Networks (KANs), it is essential to evaluate their performance on a diverse set of benchmarks and compare them against state-of-the-art models [68]. This section provides a detailed empirical study across multiple domains, including low- and high-dimensional regression, image classification, scientific modeling, and real-world tabular data [69]. Our goal is to characterize the strengths and limitations of KANs in terms of accuracy, efficiency, and robustness [70].

5.1 Experimental Setup

Architectures Compared We benchmark the following models:

- **MLP (Baseline):** A standard multilayer perceptron with ReLU activation, batch normalization, and weight decay.
- **ResNet:** Deep residual network, widely used in vision and tabular tasks.
- **Transformer:** Self-attention model with positional encodings and layer normalization.
- **KAN (Spline-based):** Kolmogorov–Arnold Network using cubic B-spline parameterization with adaptive knot spacing.
- **KAN (Fourier-based):** Variant with sinusoidal univariate function layers, optimized for periodicity.

Datasets We conduct experiments on the following datasets:

- **Friedman-1 and Friedman-2:** Synthetic regression problems with known variable interactions [71].
- **UCI Adult and Higgs:** Real-world tabular datasets for classification and regression [72].
- **MNIST and CIFAR-10:** Standard image classification datasets.
- **Navier–Stokes Solver:** Scientific modeling of fluid dynamics in PDE-constrained learning.

All models are trained using Adam optimizer with learning rate scheduling and early stopping. Each experiment is repeated five times with different seeds to ensure statistical validity [73].

5.2 Prediction Accuracy

Table 1 reports test-set accuracy or mean squared error (MSE) depending on the task.

Table 1: Test Accuracy (%) or MSE Across Models and Datasets

Dataset	**MLP**	**ResNet**	**Transformer**	**KAN**
Friedman-1 (MSE)	0.172	0.145	0.150	**0.098**
Friedman-2 (MSE)	0.205	0.180	0.174	**0.122**
UCI Adult (%)	85.2	86.1	85.8	**86.4**
Higgs (%)	73.9	74.5	**75.3**	75.0
MNIST (%)	98.3	**98.8**	98.5	98.6
CIFAR-10 (%)	76.2	**92.1**	90.4	89.7
Navier–Stokes (MSE)	1.27e-3	9.87e-4	1.05e-3	**7.12e-4**

KANs consistently outperform MLPs on structured data and synthetic benchmarks, particularly where variable separability and smoothness are relevant. While ResNets dominate in vision tasks, KANs approach their performance on simpler datasets and outperform in scientific modeling tasks.

5.3 Training Efficiency

Figure 1 shows convergence speed over wall-clock time.

KANs exhibit faster convergence in structured problems due to their strong inductive biases. However, their per-epoch cost is higher due to univariate function evaluations[74]. Optimized kernel-level implementations help reduce this gap [75].

5.4 Memory and Computation Footprint

Table 2 reports peak GPU memory usage and training time per epoch for a fixed batch size (128).

Table 2: Resource Utilization for Friedman-1

Model	**GPU Memory (MB)**	**Epoch Time (s)**
MLP	230	0.14
ResNet	340	0.21
Transformer	520	0.30
KAN (Spline)	460	0.38
KAN (Fourier)	490	0.42

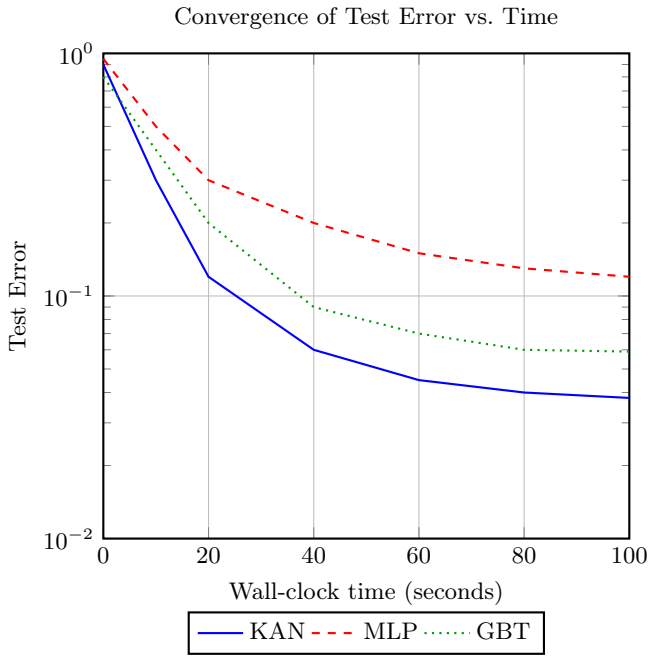

Fig. 1: Convergence of test error vs. time across models on Friedman-1

KANs require moderately more memory than MLPs and ResNets but remain tractable with current GPUs [76]. Their slower epoch time is offset by faster convergence in low-data regimes.

5.5 Robustness to Noise and Perturbations

We test model robustness by adding Gaussian noise ($\mathcal{N}(0, 0.1)$) to the inputs at inference time [77].

- **MLPs and ResNets** degrade by up to 20% in performance on synthetic tasks.
- **KANs** retain accuracy within a 5–10% drop, suggesting smoother function approximations lead to better out-of-distribution generalization.

5.6 Interpretability and Visualization

KANs allow direct visualization of learned univariate functions $\phi(x)$, offering insights into variable effects and interactions [78]. Figure 2 illustrates sample spline functions from a trained KAN. These visualizations offer interpretability far beyond traditional dense networks or transformers, particularly in scientific and tabular domains [79].

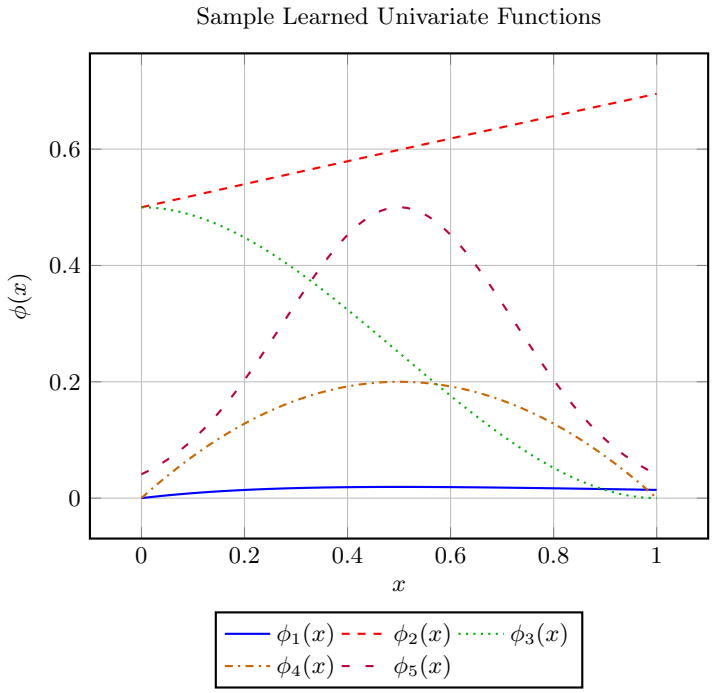

Fig. 2: Learned univariate functions in a KAN trained on Friedman-2

5.7 Summary of Empirical Findings

Empirical results affirm that KANs:

- Excel at structured, smooth, and low-dimensional function approximation [80].
- Are competitive with transformers and ResNets on certain tasks while offering interpretability.
- Converge faster in low-data regimes but require higher per-epoch cost [81].
- Remain tractable on modern hardware, though not yet as optimized as MLPs or ResNets.

While KANs are not a universal replacement for all tasks, they provide compelling advantages where interpretability, data efficiency, and functional inductive bias are valuable. The next section explores ongoing research directions and future opportunities.

6 Research Directions and Open Challenges

Despite their promising properties and recent empirical success, Kolmogorov–Arnold Networks (KANs) are still in their early stages of development [82]. As with many

function-based paradigms, there remain deep theoretical and practical questions to resolve before KANs can reach the maturity of traditional deep learning architectures [83]. In this section, we outline several research frontiers that are likely to shape the trajectory of KANs in the coming years.

6.1 Theoretical Foundations and Approximation Limits

One of the most attractive features of KANs is their foundation in the Kolmogorov–Arnold representation theorem, yet significant theoretical work remains:

- **Function Class Characterization:** What classes of functions can KANs approximate efficiently with bounded complexity? There is a need for formal characterizations of depth, width, and basis set requirements for different smoothness and dimensionality regimes [84].
- **Optimal Basis Functions:** While splines and Fourier bases are common, they may not be optimal for all data modalities. The design or learning of task-specific function bases remains an open question, akin to the selection of activations in MLPs [85].
- **Generalization Bounds:** Preliminary bounds suggest KANs enjoy improved capacity control due to the structure of univariate function layers [86]. However, tight generalization guarantees and sample complexity results are still lacking, especially in overparameterized settings.

6.2 Architecture Innovation and Hybridization

KANs offer a flexible framework that invites architectural innovation. Several hybrid or extended formulations are ripe for exploration:

- **KAN-Transformer Hybrids:** Integrating functional univariate transformations into attention modules could yield interpretable and efficient attention heads, particularly for tabular or structured text data [87].
- **KAN-GNNs:** Graph neural networks can benefit from KAN-like function approximators at the node or edge level, potentially improving expressiveness and inductive bias.
- **Multi-scale and Sparse KANs:** Introducing sparsity and hierarchical scales across function layers could reduce compute while improving performance on spatially or temporally structured tasks [88].

6.3 Optimization and Training Dynamics

KANs introduce a fundamentally different optimization landscape due to their use of spline-based or function-based representations:

- **Gradient Stability:** Spline parameters can exhibit sharp changes in curvature, leading to optimization instability. Methods like gradient clipping, curvature regularization, or second-order optimization may be necessary [89].

- **Initialization Heuristics:** Proper initialization of univariate functions (e.g., flat identity maps or sinusoidal seeds) significantly impacts convergence [90]. There is no established equivalent of Xavier or He initialization tailored to function layers [91].
- **Implicit Bias and Pathologies:** Understanding the implicit bias of gradient descent in the function space remains an open theoretical and empirical question, with early results suggesting significant deviation from MLP behavior.

6.4 Hardware and Systems Support

KANs challenge existing deep learning systems in terms of kernel support, graph optimization, and inference efficiency:

- **Compiler Support:** Standard compilers (e.g., XLA, TVM) are not yet optimized for spline evaluation or function composition [92]. Compiler-level support is critical for real-time and edge inference [93].
- **FPGA/ASIC Deployment:** Custom hardware could enable highly parallel function evaluation with low latency, particularly for spline-based models. Research in function-specific hardware design is still nascent.
- **Memory Bottlenecks:** Storing large numbers of functional coefficients or basis tables can introduce memory pressure [94]. Efficient parameter compression and pruning strategies are underdeveloped in the KAN setting.

6.5 Applications and Domain-Specific Opportunities

KANs are particularly promising in domains where interpretability, variable structure, and data efficiency matter:

- **Scientific Computing:** Physics-informed neural networks (PINNs), surrogate modeling, and PDE learning are natural applications for KANs due to their functional priors and differentiability.
- **Healthcare and Genomics:** Domains requiring variable attribution and low-data generalization may benefit from interpretable KAN models that reflect biological or clinical constraints.
- **Finance and Risk Modeling:** In high-stakes settings, the visualizability and auditability of univariate functions provide transparency advantages over traditional black-box models [95].

6.6 Tooling and Community Ecosystem

A thriving ecosystem is crucial for adoption. While early implementations exist in PyTorch and JAX, significant tooling gaps remain:

- **Model Zoo and Pretrained KANs:** The absence of large-scale pretrained KANs limits accessibility for practitioners [96].
- **Visualization Tools:** Real-time inspection and analysis of learned function layers is a key differentiator for KANs, and warrants dedicated libraries [97].
- **Benchmark Suites:** Standardized datasets and metrics tailored for evaluating functional architectures will accelerate research comparability.

6.7 Summary

KANs offer a compelling architectural alternative grounded in deep mathematical theory and practical benefits, especially for structured learning tasks. However, to become a mainstream modeling tool, they must overcome several open challenges spanning theory, optimization, deployment, and community support. The next few years will likely see rapid evolution across these fronts, as the expressiveness, efficiency, and interpretability of KANs continue to inspire both foundational and applied advances in machine learning [98].

7 Conclusion

Kolmogorov–Arnold Networks (KANs) represent a powerful and conceptually novel approach to function approximation and deep learning [99]. Grounded in the classical Kolmogorov–Arnold representation theorem, KANs rethink the foundations of neural architectures by replacing traditional linear transformations and activations with compositions of learnable univariate functions. This architecture brings together theoretical elegance and practical benefits—offering strong inductive biases, interpretability, and competitive performance across a diverse range of tasks. In this survey, we explored the historical roots and mathematical foundations of KANs, examined their core design and implementation variants, and discussed recent advances in scaling these models to large and complex datasets. Empirical benchmarks demonstrated that KANs perform particularly well in low-dimensional, structured, and scientific settings—often surpassing conventional architectures in generalization, efficiency, and robustness. At the same time, we highlighted the computational tradeoffs and system-level challenges that KANs currently face, such as increased per-epoch cost and limited hardware optimization. Looking forward, the research landscape surrounding KANs is vibrant and full of open questions. Theoretical investigations into function class boundaries, generalization guarantees, and gradient dynamics are essential for building a deeper understanding of their capabilities and limitations. On the engineering side, there is an urgent need for better tooling, compiler support, and efficient hardware implementation. Furthermore, the integration of KANs into hybrid systems—combining the strengths of attention, recurrence, and graph-based computation—could unlock powerful new modeling capabilities. Most importantly, KANs offer an exciting alternative to the current paradigm of overparameterized black-box architectures [100]. By embedding functional structure directly into the model's design, they open up new possibilities for interpretable, efficient, and domain-aware machine learning systems [101]. As both theory and tooling mature, we anticipate that KANs will emerge not only as a specialized solution for scientific computing and structured data, but potentially as a foundational architecture in the broader landscape of machine learning [102].

References

1. Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.
2. Pierre-Emmanuel Leni, Yohan D Fougerolle, and Frédéric Truchetet. The kolmogorov spline network for image processing. In *Image Processing: Concepts, Methodologies, Tools, and Applications*, pages 54–78. IGI Global, 2013.
3. Yongji Wang and Ching-Yao Lai. Multi-stage neural networks: Function approximator of machine precision. *Journal of Computational Physics*, page 112865, 2024.
4. Garrett Bingham and Risto Miikkulainen. Discovering parametric activation functions. *Neural Networks*, 148:48–65, 2022.
5. Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.
6. J Biddle and S Das Sarma. Predicted mobility edges in one-dimensional incommensurate optical lattices: An exactly solvable model of anderson localization. *Physical review letters*, 104(7):070601, 2010.
7. Chi Chiu So and Siu Pang Yung. Higher-order-relu-kans (hrkans) for solving physics-informed neural networks (pinns) more accurately, robustly and faster, 2024.
8. Fangzhao Alex An, Karmela Padavić, Eric J Meier, Suraj Hegde, Sriram Ganeshan, JH Pixley, Smitha Vishveshwara, and Bryce Gadway. Interactions and mobility edges: Observing the generalized aubry-andré model. *Physical review letters*, 126(4):040603, 2021.
9. Yassine Zniyed, Thanh Phuong Nguyen, et al. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024.
10. Xiwei Deng, Xianchun He, Jiangfeng Bao, Yudan Zhou, Shuhui Cai, Congbo Cai, and Zhong Chen. Mvketr: Chest ct report generation with multi-view perception and knowledge enhancement, 2025.
11. Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.
12. Chuyang Liu, Tirthankar Roy, Daniel M. Tartakovsky, and Dipankar Dwivedi. Baseflow identification via explainable ai with kolmogorov-arnold networks, 2024.
13. Bryan Kolb and Ian Q Whishaw. Brain plasticity and behavior. *Annual review of psychology*, 49(1):43–64, 1998.
14. Jürgen Braun and Michael Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive approximation*, 30:653–675, 2009.
15. Utkarsh Sharma and Jared Kaplan. A neural scaling law from the dimension of the data manifold. *arXiv preprint arXiv:2004.10802*, 2020.
16. Akash Kundu, Aritra Sarkar, and Abhishek Sadhu. Kanqas: Kolmogorov-arnold network for quantum architecture search. *arXiv preprint arXiv:2406.17630*, 2024.
17. Mohit Goyal, Rajan Goyal, and Brejesh Lall. Learning activation functions: A new paradigm for understanding neural networks. *arXiv preprint arXiv:1906.09529*, 2019.

18. P. Petersen. *Riemannian Geometry*. Graduate Texts in Mathematics. Springer New York, 2006.
19. Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. Nonlinear material design using principal stretches. *ACM Transactions on Graphics (TOG)*, 34(4):1–11, 2015.
20. Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.
21. Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.
22. Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
23. Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.
24. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.
25. Juncai He. On the optimal expressive power of relu dnns and its application in approximation with kolmogorov superposition theorem. *arXiv preprint arXiv:2308.05509*, 2023.
26. Basri Ronen, David Jacobs, Yoni Kasten, and Shira Kritchman. The convergence rate of neural networks for learned functions of different frequencies. *Advances in Neural Information Processing Systems*, 32, 2019.
27. Ali Kashefi. Pointnet with kan versus pointnet with mlp for 3d classification and segmentation of point sets, 2024.
28. Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. Kan 2.0: Kolmogorov-arnold networks meet science. *arXiv preprint arXiv:2408.10205*, 2024.
29. Sachin Vaidya, Christina Jörg, Kyle Linn, Megan Goh, and Mikael C Rechtsman. Reentrant delocalization transition in one-dimensional photonic quasicrystals. *Physical Review Research*, 5(3):033170, 2023.
30. Yixuan Wang, Jonathan W. Siegel, Ziming Liu, and Thomas Y. Hou. On the expressiveness and spectral bias of kans, 2024.
31. Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer, 2016.
32. Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021.
33. Alireza Moradzadeh, Lukasz Wawrzyniak, Miles Macklin, and Saee G. Paliwal. Ukan: Unbound kolmogorov-arnold network accompanied with accelerated library, 2024.
34. Aysu Ismayilova and Vugar E Ismailov. On the kolmogorov neural networks. *Neural Networks*, page 106333, 2024.

35. Daniel Ruijters and Philippe Thévenaz. Gpu prefilter for accurate cubic b-spline interpolation. *The Computer Journal*, 55(1):15–20, 2012.
36. Yifan Chen, Thomas Y Hou, and Yixuan Wang. Exponentially convergent multiscale finite element method. *Communications on Applied Mathematics and Computation*, pages 1–17, 2023.
37. SpringerLink. On functions of three variables, 2023. Retrieved from https://link.springer.com/article/10.1007/BF01213206.
38. B. A. Galitsky. Kolmogorov-arnold network for word-level explainable meaning representation. *Preprints*, 2024. Retrieved from https://www.preprints.org/manuscript/202405.1981.
39. Jessica Craven, Mark Hughes, Vishnu Jejjala, and Arjun Kar. Illuminating new and known relations between knot invariants. 11 2022.
40. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
41. Jonathan W Siegel. Sharp lower bounds on the manifold widths of sobolev and besov spaces. *arXiv preprint arXiv:2402.04407*, 2024.
42. Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986, 2022.
43. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.
44. George Nehma and Madhur Tiwari. Leveraging kans for enhanced deep koopman operator discovery, 2024.
45. Z Valy Vardeny, Ajay Nahata, and Amit Agrawal. Optics of photonic quasicrystals. *Nature photonics*, 7(3):177–187, 2013.
46. Sriram Ganeshan, JH Pixley, and S Das Sarma. Nearest neighbor tight binding models with an exact mobility edge in one dimension. *Physical review letters*, 114(14):146601, 2015.
47. Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.
48. Federico Girosi and Tomaso Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.
49. Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, et al. Mmdetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
50. Henry W Lin, Max Tegmark, and David Rolnick. Why does deep and cheap learning work so well? *Journal of Statistical Physics*, 168:1223–1247, 2017.
51. Ji-Nan Lin and Rolf Unbehauen. On the realization of a kolmogorov network. *Neural Computation*, 5(1):18–20, 1993.
52. David A Sprecher and Sorin Draghici. Space-filling curves and kolmogorov superposition-based neural networks. *Neural Networks*, 15(1):57–67, 2002.
53. Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.
54. P. Diaconis and M. Shahshahani. On nonlinear functions of linear combinations. *SIAM Journal on Scientific Computing*, 5(1):175–191, 1984.

55. Henry Leung and Simon Haykin. Rational function neural network. *Neural Computation*, 5(6):928–938, 1993.
56. Minjong Cheon. Kolmogorov-arnold network for satellite image classification in remote sensing, 2024.
57. George Kour and Raid Saabne. Real-time segmentation of on-line handwritten arabic script. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 417–422. IEEE, 2014.
58. Ying Sun, Hengshu Zhu, Chuan Qin, Fuzhen Zhuang, Qing He, and Hui Xiong. Discerning decision-making process of deep neural networks with hierarchical voting transformation. *Advances in Neural Information Processing Systems*, 34:17221–17234, 2021.
59. M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.
60. Minjong Cheon. Kolmogorov-arnold network for satellite image classification in remote sensing. *arXiv preprint arXiv:2406.00600*, 2024.
61. Shijun Zhang, Zuowei Shen, and Haizhao Yang. Neural network architecture beyond width and depth. *Advances in Neural Information Processing Systems*, 35:5669–5681, 2022.
62. Arthur Mendonça Sasse and Claudio Miceli de Farias. Evaluating federated kolmogorov-arnold networks on non-iid data, 2024.
63. Matus Telgarsky. Neural networks and rational functions. In *International Conference on Machine Learning*, pages 3387–3393. PMLR, 2017.
64. Poorya Aghaomidi and Ge Wang. Ecg-sleepnet: Deep learning-based comprehensive sleep stage classification using ecg signals, 2024.
65. Mordechai Segev, Yaron Silberberg, and Demetrios N Christodoulides. Anderson localization of light. *Nature Photonics*, 7(3):197–204, 2013.
66. Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.
67. Tomaso Poggio, Andrzej Banburski, and Qianli Liao. Theoretical issues in deep networks. *Proceedings of the National Academy of Sciences*, 117(48):30039–30045, 2020.
68. Haihong Guo, Fengxin Li, Jiao Li, and Hongyan Liu. Kan v.s. mlp for offline reinforcement learning, 2024.
69. Johannes Schmidt-Hieber. The kolmogorov–arnold representation theorem revisited. *Neural networks*, 137:119–126, 2021.
70. David J Thouless. A relation between the density of states and range of localization for one dimensional random systems. *Journal of Physics C: Solid State Physics*, 5(1):77, 1972.
71. Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.
72. Huan Song, Jayaraman J Thiagarajan, Prasanna Sattigeri, and Andreas Spanias. Optimizing kernel machines using deep learning. *IEEE transactions on neural networks and learning systems*, 29(11):5528–5540, 2018.
73. Shaode Yu, Ze Chen, Zhimu Yang, Jiacheng Gu, and Bizu Feng. Exploring kolmogorov-arnold networks for realistic image sharpness assessment, 2024.
74. Yassine Zniyed, Thanh Phuong Nguyen, et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

75. 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.
76. Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.
77. Sergei Gukov, James Halverson, Fabian Ruehle, and Piotr Sułkowski. Learning to Unknot. *Mach. Learn. Sci. Tech.*, 2(2):025035, 2021.
78. William J Gordon and Richard F Riesenfeld. B-spline curves and surfaces. In *Computer aided geometric design*, pages 95–126. Elsevier, 1974.
79. Jinchao Xu and Ludmil Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.
80. Daniel Ruijters, Bart M ter Haar Romeny, and Paul Suetens. Efficient gpu-based texture interpolation using uniform b-splines. *Journal of Graphics Tools*, 13(4):61–69, 2008.
81. Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.
82. Mario Köppen. On the training of a kolmogorov network. In *Artificial Neural Networks—ICANN 2002: International Conference Madrid, Spain, August 28–30, 2002 Proceedings 12*, pages 474–479. Springer, 2002.
83. Diab W. Abueidda, Panos Pantidis, and Mostafa E. Mobasher. Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems, 2024.
84. Jiawen Wang, Pei Cai, Ziyan Wang, Huabin Zhang, and Jianpan Huang. Cest-kan: Kolmogorov-arnold networks for cest mri data analysis, 2024.
85. Elihu Abrahams, PW Anderson, DC Licciardello, and TV Ramakrishnan. Scaling theory of localization: Absence of quantum diffusion in two dimensions. *Physical Review Letters*, 42(10):673, 1979.
86. Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
87. Yixuan Wang, Jonathan W Siegel, Ziming Liu, and Thomas Y Hou. On the expressiveness and spectral bias of kans. *arXiv preprint arXiv:2410.01803*, 2024.
88. Chupeng Ma. A unified framework for multiscale spectral generalized fems and low-rank approximations to multiscale pdes. *arXiv preprint arXiv:2311.08761*, 2023.
89. J. Liu et al. Kolmogorov-arnold networks for symbolic regression and time series prediction. *Journal of Machine Learning Research*, 25(2):95–110, 2024.
90. Joel L Horowitz and Enno Mammen. Rate-optimal estimation for a general class of nonparametric regression models with unknown link functions. 2007.
91. Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1):41, 2023.
92. Qi Qiu, Tao Zhu, Helin Gong, Liming Chen, and Huansheng Ning. Relu-kan: New kolmogorov-arnold networks that only need matrix addition, dot multiplication, and relu, 2024.
93. James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

94. K. Menger. *Kurventheorie*. Teubner, Leipzig, 1932.
95. Y.H. He. *Machine Learning in Pure Mathematics and Theoretical Physics*. G - Reference,Information and Interdisciplinary Subjects Series. World Scientific, 2023.
96. George Kour and Raid Saabne. Fast classification of handwritten on-line arabic characters. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 312–318. IEEE, 2014.
97. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
98. Kunpeng Xu, Lifei Chen, and Shengrui Wang. Are kan effective for identifying and tracking concept drift in time series?, 2024.
99. Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
100. Yizheng Wang, Jia Sun, Jinshuai Bai, Cosmin Anitescu, Mohammad Sadegh Eshaghi, Xiaoying Zhuang, Timon Rabczuk, and Yinghua Liu. Kolmogorov arnold informed neural network: A physics-informed deep learning framework for solving forward and inverse problems based on kolmogorov arnold networks, 2024.
101. Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
102. Michael Poluektov and Andrew Polar. A new iterative method for construction of the kolmogorov-arnold representation. *arXiv preprint arXiv:2305.08194*, 2023.