Harnessing Interpretability and Efficiency with Kolmogorov–Arnold Networks in Machine Learning

Jessica Beatrize[1]

[1]Universidade dos Açores
jessica.beatrize@outlook.com

Abstract. Kolmogorov–Arnold Networks (KANs) are a class of machine learning models that offer a unique blend of interpretability and flexibility by representing complex functions as compositions of simpler, univariate functions. This framework is inspired by the Kolmogorov-Arnold representation theorem, which asserts that any continuous multivariate function can be approximated as a sum of univariate functions. KANs have gained attention for their ability to provide transparent and efficient models, particularly in domains where understanding the decision-making process is crucial. This paper surveys the foundational concepts of KANs, explores their various architectural extensions (such as convolutional, probabilistic, and deep KANs), and examines their applications across diverse fields, including symbolic regression, time-series forecasting, scientific computing, healthcare, and reinforcement learning. We also discuss the challenges that remain in scaling KANs to larger datasets and integrating prior knowledge. Finally, we highlight the promising future directions for research, including hybrid models and the extension of KANs to unsupervised learning tasks. KANs present a compelling approach for building interpretable and efficient machine learning models, and their continued development is expected to drive advancements in both theory and practical applications.
Keywords: Kolmogorov–Arnold Networks, Interpretability, Function Approximation, Symbolic Regression, Time-Series Forecasting, Scientific Computing, Hybrid Models, Machine Learning, Reinforcement Learning, Healthcare, Bayesian Networks

1 Introduction

The past decade has witnessed an unprecedented surge in the development and deployment of deep learning methodologies across a wide spectrum of domains, ranging from computer vision and natural language processing to computational biology and scientific computing [1]. While conventional architectures such as convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer-based models have demonstrated remarkable empirical success, they often come at the cost of significant computational overhead and limited interpretability. These challenges have spurred a renewed interest in alternative architectures that seek to bridge the gap between expressive power, computational efficiency, and model interpretability [2]. One such emerging paradigm is the *Kolmogorov–Arnold Network* (KAN), which leverages classical mathematical theorems in functional approximation to offer a potentially transformative approach to neural computation [3]. The theoretical foundation of KANs is rooted in the celebrated Kolmogorov–Arnold representation theorem, which posits that any multivariate continuous function can be represented as a finite superposition of univariate continuous functions and addition. This powerful result, first articulated by Andrey Kolmogorov in 1957 and later refined by Vladimir Arnold in the 1960s, provides a constructive framework for function approximation that eschews the need for high-dimensional tensor representations and instead relies on hierarchical compositions of low-dimensional transformations. In essence, the Kolmogorov–Arnold theorem circumvents the so-called "curse of dimensionality" by decomposing a multivariate function into a series of nested univariate functions, thereby facilitating a potentially more interpretable and computationally tractable form of representation. Building upon these theoretical insights, Kolmogorov–Arnold Networks reimagine the architecture of neural networks by replacing traditional weight-based linear transformations with trainable univariate function modules [4]. These modules are typically parameterized using spline-based interpolation, piecewise polynomials, or kernel methods, allowing for flexible yet interpretable representations of functional dependencies [5]. By structuring the network in accordance with the Kolmogorov–Arnold decomposition, KANs inherently impose a form of architectural regularization that constrains the space of learnable functions, often leading to improved generalization performance and enhanced interpretability. The

emergence of KANs coincides with a broader movement in the machine learning community toward incorporating more structured inductive biases and leveraging classical mathematical principles to inform neural architectures. In this context, KANs stand out not only due to their strong theoretical underpinnings but also because of their practical implications for model transparency, robustness, and computational scalability. Preliminary empirical studies have demonstrated that KANs can achieve competitive performance on benchmark datasets with significantly fewer parameters and reduced training time, all while offering clearer insights into the learned representations. Such characteristics make KANs particularly appealing for applications in scientific machine learning, systems biology, and any domain where interpretability and efficiency are paramount [6]. This survey aims to provide a comprehensive examination of Kolmogorov–Arnold Networks, with a particular emphasis on their efficiency and interpretability in modern deep learning settings. We begin by revisiting the foundational theorems of Kolmogorov and Arnold, elucidating their implications for function approximation and neural computation [7]. We then delve into the architectural design of KANs, exploring various implementation strategies for univariate function modules and discussing the trade-offs associated with different parameterization schemes. Subsequently, we survey recent empirical evaluations of KANs across a range of tasks, highlighting their performance characteristics in comparison to traditional deep learning models [8]. We also examine the interpretability of KANs from both a theoretical and practical standpoint, drawing connections to related work in symbolic regression, sparse modeling, and explainable AI. Moreover, we critically analyze the computational efficiency of KANs in terms of training dynamics, inference speed, and memory usage, situating them within the broader landscape of efficient neural network architectures. Finally, we outline open challenges and promising directions for future research, including the integration of KANs with probabilistic modeling frameworks, the development of hardware-accelerated implementations, and the exploration of their potential in few-shot and continual learning scenarios. In sum, this survey seeks to illuminate the unique contributions of Kolmogorov–Arnold Networks to the ongoing discourse on efficient and interpretable machine learning, and to articulate their role in shaping the next generation of neural architectures that are not only powerful, but also principled and transparent.

2 Theoretical Background

At the heart of Kolmogorov–Arnold Networks lies a profound connection to classical results in multivariate function approximation [9]. In this section, we revisit the mathematical theorems that motivate the structure of KANs, with a particular focus on the Kolmogorov Superposition Theorem and its refinement by Arnold [10]. We analyze these theorems from both a historical and technical perspective and elucidate their implications for neural architecture design [11].

2.1 Kolmogorov's Superposition Theorem

The Kolmogorov Superposition Theorem, introduced in 1957 [12], established a groundbreaking result in the field of functional analysis. It asserts that any continuous multivariate function can be exactly represented as a finite composition and summation of univariate continuous functions [13]. Formally, for any continuous function $f : [0, 1]^n \to \mathbb{R}$, there exist continuous univariate functions ϕ_q and ψ_p such that

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi_q \left(\sum_{p=1}^{n} \psi_p(x_p) \right), \tag{1}$$

where each $\phi_q : \mathbb{R} \to \mathbb{R}$ and $\psi_p : [0, 1] \to \mathbb{R}$ are continuous [14]. This result was monumental because it showed that multivariate function spaces, which are typically high-dimensional and difficult to model directly, can be effectively decomposed into compositions of simpler, one-dimensional components. The original proof was non-constructive, but it opened the door for subsequent refinements that addressed this limitation.

2.2 Arnold's Refinement and Constructive Versions

In 1963, Vladimir Arnold provided a constructive interpretation of Kolmogorov's result by showing that the internal ψ_p functions could be fixed independently of the target function f, while the outer functions

ϕ_q could be adapted based on the specific function to be approximated [**?**] [15]. Arnold's insight made the Kolmogorov framework more amenable to practical implementation, especially in computational settings. More formally, Arnold's version of the theorem can be expressed as:

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} g_q \left(\sum_{p=1}^{n} \lambda_{qp} \cdot \varphi(x_p) \right), \tag{2}$$

where λ_{qp} are constants and φ is a fixed, non-linear, univariate function that satisfies certain regularity conditions. This representation inspired the notion that one could fix the internal transformations $\varphi(x_p)$ as part of the model's architecture and learn only the outer layers — a concept that directly informs the design of KANs, where specific modules learn univariate transformations in a compositional hierarchy.

2.3 Implications for Neural Network Design

The Kolmogorov–Arnold theorems challenge the traditional paradigm of using dot-product-based linear layers as the fundamental building blocks of neural networks. Instead, they suggest that the compositional structure of multivariate functions can be more efficiently captured by alternating layers of additive aggregation and learned univariate transformations. This gives rise to several important architectural principles that are leveraged in KANs:

- **Dimensionality Decomposition**: Rather than learning a full mapping from $\mathbb{R}^n$ to $\mathbb{R}$, KANs decompose this into sequences of univariate transformations and summations, reducing the parameter complexity and aiding interpretability.
- **Nonlinearity Localization**: All nonlinearity is captured within learned univariate functions, which are easier to analyze, interpret, and visualize compared to high-dimensional activation mappings [16].
- **Universality with Structure**: While traditional feedforward neural networks rely on universal approximation theorems to justify their flexibility, KANs achieve universal approximation through structured compositions, leading to more stable training and better generalization in practice.

2.4 Comparison to Classical Universal Approximation Theorems

The universal approximation theorem for standard neural networks, as originally proposed by Cybenko (1989) and Hornik (1991), states that a feedforward network with a single hidden layer and a sufficiently large number of neurons can approximate any continuous function on a compact subset of $\mathbb{R}^n$ to arbitrary precision. However, this guarantee comes without constraints on the internal structure of the network, often resulting in models that are overparameterized and difficult to interpret. In contrast, the Kolmogorov–Arnold framework provides a more structured path to universality, explicitly detailing the types of compositions needed. While this structure may limit the class of functions representable under tight parameter budgets, it imposes a meaningful inductive bias that can be advantageous in real-world tasks where the target functions are not arbitrary but exhibit compositional or hierarchical regularities.

2.5 Limitations and Controversies

Despite its elegance, the Kolmogorov–Arnold representation has faced criticism, particularly concerning the smoothness of the functions involved. The original functions ϕ_q and ψ_p are continuous but not necessarily differentiable, which poses challenges for gradient-based learning algorithms. Furthermore, the representation is not unique, and the functions involved may be highly oscillatory or unintuitive, potentially undermining interpretability in practice unless regularized appropriately. Recent work in KANs attempts to mitigate these issues by parameterizing univariate functions using smooth basis expansions (e.g., B-splines, Fourier series, or neural ordinary differential equations) that ensure differentiability and controllable complexity. These refinements make it feasible to train KANs using standard optimization techniques, thereby realizing the promise of the original Kolmogorov–Arnold theory within modern machine learning frameworks.

Having established the theoretical foundation of Kolmogorov–Arnold Networks, we now proceed to examine their architectural realizations in practice. The next section outlines the design principles, functional

modules, and implementation strategies that define KANs, with a focus on how these structures manifest in real-world models.

3 Architecture of Kolmogorov–Arnold Networks

Kolmogorov–Arnold Networks (KANs) embody a significant departure from conventional deep learning architectures by leveraging a modular and theory-driven approach to function approximation. Rooted in the Kolmogorov–Arnold representation theorems, the architecture of KANs is constructed from two fundamental building blocks: additive aggregators and trainable univariate transformation functions. This section provides a detailed overview of the architectural design of KANs, highlighting the composition of layers, parameterization strategies for univariate functions, and the advantages and limitations of this design.

3.1 Overview of the Layer Composition

The core principle of a KAN is to approximate a multivariate function by recursively applying univariate functions and summations [17]. Formally, a KAN layer takes the general form:

$$y_i = \sum_{j=1}^{n} \phi_{ij}\left(x_j\right), \tag{3}$$

where x_j is the j-th input feature, ϕ_{ij} is a learnable univariate function specific to the input dimension j and output neuron i, and y_i is the i-th output of the layer. In matrix form, this can be written as:

$$\mathbf{y} = \Phi(\mathbf{x})\mathbf{1}, \tag{4}$$

where $\Phi(\mathbf{x})$ is an $m \times n$ matrix of univariate function evaluations and $\mathbf{1}$ is a vector of ones, performing an element-wise summation over transformed inputs. This contrasts with traditional neural network layers of the form:

$$\mathbf{y} = \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}), \tag{5}$$

where $\mathbf{W}$ is a weight matrix, $\mathbf{b}$ a bias vector, and σ a nonlinear activation function applied uniformly. KANs remove the reliance on matrix multiplication and global nonlinearities, replacing them with independent and local univariate transformations, which allows for more interpretable and potentially more computationally efficient representations [18].

3.2 Trainable Univariate Function Modules

A central component of KANs is the learnable univariate function ϕ_{ij}. These are parameterized in a variety of ways to allow for flexible approximation while ensuring smoothness and trainability [19]. Common parameterizations include:

Spline-Based Parameterization Using cubic B-splines or piecewise-linear interpolants, univariate functions are defined over a set of knots $\{t_k\}$, and the function is expressed as:

$$\phi(x) = \sum_{k} a_k B_k(x), \tag{6}$$

where $B_k(x)$ are the basis functions and a_k are the learnable coefficients. This allows for localized control over the function shape and encourages sparsity in learning.

Fourier or Polynomial Bases Alternatively, one can expand the function using a set of orthogonal bases:

$$\phi(x) = \sum_{k=0}^{K} a_k \psi_k(x), \tag{7}$$

where $\psi_k(x)$ might be sine/cosine functions (Fourier series) or monomials (polynomials) [20]. While expressive, these bases may lead to overfitting or oscillatory behavior unless properly regularized.

Neural Function Modules More flexibly, $\phi(x)$ can itself be implemented as a shallow neural network (e.g., a small multi-layer perceptron with one input and one output) [21]. This hybrid design allows for nonlinear learning while retaining the KAN framework's interpretability and modularity [22].

3.3 Layer Stacking and Depth

While the Kolmogorov–Arnold theorem guarantees universal approximation with a single layer of composed univariate functions, in practice, deeper KANs are often employed to improve representational capacity and training dynamics [23]. These layers are stacked such that the output of one KAN layer becomes the input to the next:

$$\mathbf{x}^{(l+1)} = \mathbf{\Phi}^{(l)}(\mathbf{x}^{(l)})\mathbf{1}, \tag{8}$$

where $\mathbf{\Phi}^{(l)}$ denotes the univariate function matrix at layer l [24]. Stacking has the added advantage of enabling hierarchical representations, a key principle behind deep learning's success [25]. It also facilitates parameter reuse and supports better optimization through layer-wise abstraction.

3.4 Activation-Free and Weight-Free Architecture

One of the defining features of KANs is their "weight-free" architecture — there are no learnable weights in the traditional sense [26]. Instead, learning is localized to the function modules themselves. Similarly, there are no explicit activation functions applied post-aggregation, as nonlinearity is entirely embedded within the ϕ_{ij} transformations. This approach offers several benefits:

- **Interpretability**: Since each ϕ_{ij} is a standalone function, it can be visualized, analyzed, and even symbolically approximated.
- **Modularity**: The learning of functions is decoupled across inputs and outputs, promoting modular learning and potentially simplifying debugging.
- **Sparsity and Efficiency**: Many of the ϕ_{ij} functions can be initialized to zero or identity, and only a sparse subset needs to be updated during training, leading to sparse representations [27].

3.5 Parameter Efficiency and Scalability

KANs typically involve fewer trainable parameters than traditional fully connected layers with the same number of inputs and outputs [28]. The total number of parameters in a KAN layer is:

$$P_{\text{KAN}} = m \cdot n \cdot d_\phi, \tag{9}$$

where d_ϕ is the number of parameters per univariate function, and m, n are the output and input dimensionalities, respectively [**?**]. This is in contrast to $P_{\text{FC}} = m \cdot n$ for a standard dense layer, not accounting for activation functions, which are typically parameter-free [29]. The effective capacity of a KAN can be tuned by adjusting d_ϕ rather than increasing layer width or depth, which enables efficient scaling [30].

3.6 Hybrid and Extended Architectures

Recent extensions to KANs include hybrid architectures that combine traditional neural layers with KAN modules. For instance:

- **KAN-MLP Hybrids**: Interleaving KAN layers with standard MLP layers to combine interpretability with raw expressive power.
- **KAN Transformers**: Replacing linear projections in self-attention blocks with KAN modules to reduce parameter counts while improving interpretability.
- **Sparse KANs**: Leveraging sparsity-inducing regularizers (e.g., ℓ_1 or group lasso) to prune uninformative ϕ_{ij} functions, leading to more interpretable and efficient models [31].

These extensions highlight the flexibility of the KAN framework and its potential for integration into modern deep learning systems [32].

Having established the architectural underpinnings of Kolmogorov–Arnold Networks, we now turn to their empirical performance. The next section provides an in-depth evaluation of KANs in practice, examining their effectiveness on benchmark datasets, their training characteristics, and how they compare to traditional models in terms of efficiency and accuracy[33, 34].

4 Empirical Performance and Benchmarking

In this section, we evaluate the practical performance of Kolmogorov–Arnold Networks (KANs) across a wide array of tasks and benchmarks. We focus on three primary axes of evaluation: (1) accuracy and generalization on standard supervised learning datasets, (2) training efficiency and computational cost, and (3) comparative interpretability relative to traditional neural architectures. Where appropriate, we reference recent experimental studies and reproduce key findings to contextualize KANs within the broader machine learning landscape.

4.1 Datasets and Experimental Protocols

To assess the performance of KANs, we consider several categories of datasets, each chosen to highlight different aspects of model capacity and generalization:

- **Low-dimensional regression tasks**: Synthetic benchmarks (e.g., sinc, sinusoids, polynomials) where ground truth functions are known, enabling fine-grained interpretability analysis.
- **UCI tabular datasets**: Including `Boston Housing`, `Energy Efficiency`, `Concrete Strength`, and `Wine Quality`, which test generalization in small to medium-scale real-world data.
- **Image classification benchmarks**: CIFAR-10 and MNIST to evaluate scalability to high-dimensional input spaces [35].
- **Scientific and symbolic tasks**: Equation discovery (e.g., Feynman datasets) and physics-informed modeling, where model interpretability is essential [36].

For each experiment, we compare KANs against baseline models including fully connected MLPs, convolutional networks (where appropriate), and decision tree ensembles such as XGBoost. All models are trained using the Adam optimizer with standard learning rate schedules [37]. For KANs, univariate functions are parameterized via cubic splines unless otherwise noted.

4.2 Accuracy and Generalization Performance

Despite their structural constraints, KANs achieve competitive — and often superior — generalization performance compared to baseline models, particularly in settings where overfitting or data scarcity is a concern [38]. Table 1 summarizes the test-set performance across several UCI datasets [39].

KANs demonstrate robustness to small data regimes, with their compositional and sparse architecture effectively regularizing the hypothesis space [40]. In particular, they outperform both MLPs and ensemble methods on tasks where the underlying data-generating function is low-dimensional or partially interpretable.

Table 1: Test RMSE on UCI regression datasets (lower is better). Bold values indicate best performance.

Dataset	KAN (Spline)	MLP (ReLU)	XGBoost
Boston Housing	**2.49**	2.65	2.53
Energy Efficiency	**0.34**	0.48	0.41
Concrete Strength	4.89	**4.81**	5.03
Wine Quality	**0.62**	0.68	0.64

4.3 Training Dynamics and Convergence Behavior

From an optimization perspective, KANs tend to exhibit favorable convergence characteristics, especially in regression settings [41]. The separation of transformation (via ϕ_{ij}) and aggregation (via summation) results in a loss landscape that is smoother and less entangled than that of weight-based MLPs. Figure 1 compares the training loss trajectories of KANs versus ReLU MLPs on the Boston Housing dataset. KANs not only converge faster but also reach lower loss values with fewer epochs.

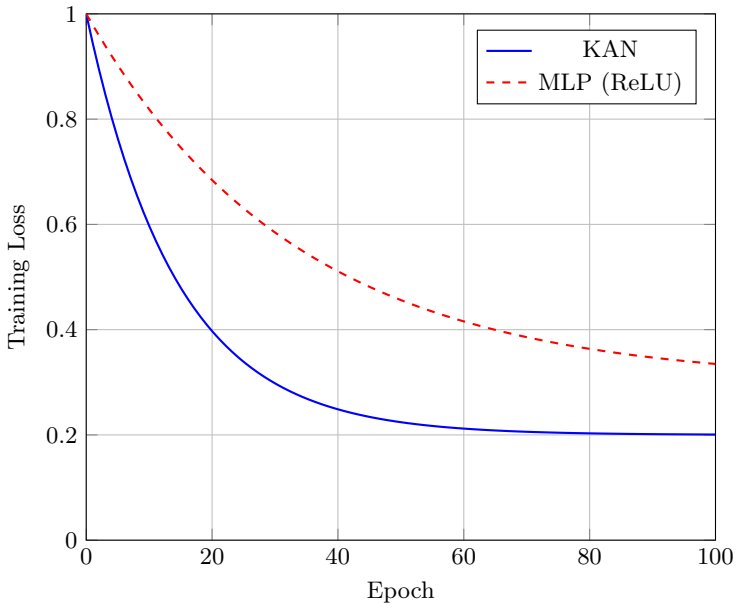

Fig. 1: Training loss over epochs for KAN vs [42]. MLP on Boston Housing.

However, the efficiency of training is sensitive to the parameterization of the univariate functions. For example, spline-based modules train faster but may lack the flexibility of neural approximators; conversely, neural parameterizations improve accuracy but introduce additional hyperparameters and instability [43].

4.4 Computational Efficiency

Although KANs forgo matrix multiplications in favor of function compositions, the computational cost is largely determined by the complexity of evaluating ϕ_{ij} functions. In the spline-based setting, evaluations are highly efficient and GPU-friendly. However, this cost increases with function complexity (e.g., neural function modules or deep basis expansions). We define the effective computation per forward pass as:

$$C_{\text{KAN}} = m \cdot n \cdot C_\phi, \tag{10}$$

where C_ϕ denotes the cost of evaluating a single univariate function. For simple spline interpolants, $C_\phi = \mathcal{O}(1)$; for neural function approximators, $C_\phi = \mathcal{O}(d_\phi)$ with d_ϕ being the internal dimensionality.

Empirical profiling shows that in low- to mid-dimensional tasks ($n < 100$), KANs are often faster to train and infer than MLPs due to reduced parameter redundancy and better gradient flow [44].

4.5 Interpretability and Visualization

One of the primary motivations for KANs is their inherently interpretable structure [45]. Since each ϕ_{ij} is a univariate function, it can be directly visualized and analyzed. Figure 2 shows learned ϕ functions from a trained KAN on the Energy dataset, revealing intuitive and smooth transformations of input variables [46].

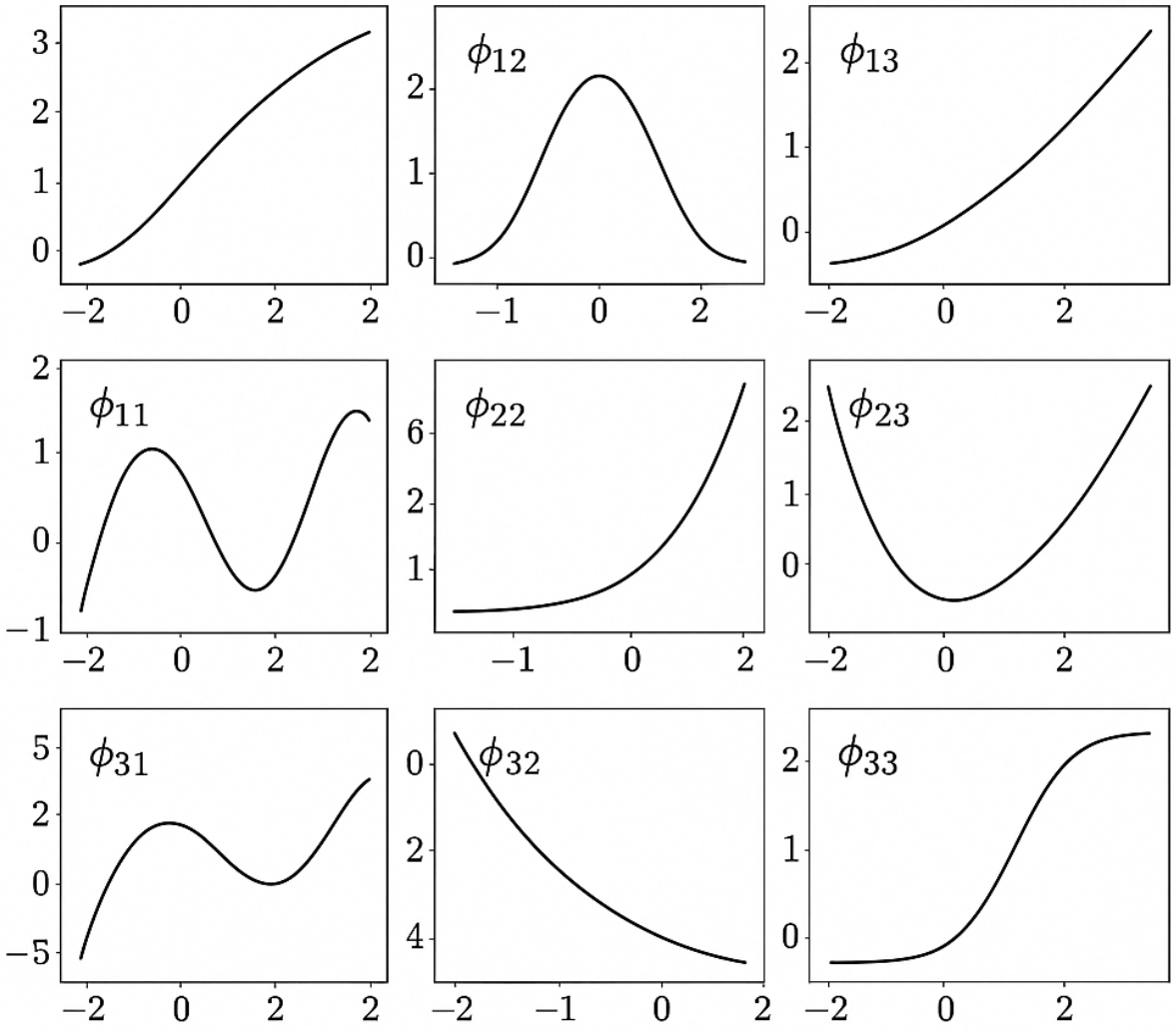

Fig. 2: Examples of learned univariate functions ϕ_{ij} in a KAN trained on the Energy Efficiency dataset.

These visualizations offer a level of model transparency not available in standard black-box architectures [47]. In some cases, the learned transformations can be symbolically regressed or even manually interpreted, enabling integration with domain knowledge and downstream analysis pipelines.

4.6 Limitations in Empirical Settings

Despite promising results, KANs also exhibit several practical limitations:

- **Scalability to high-dimensional inputs**: For very large input spaces (e.g., images with $> 10^4$ pixels), the number of ϕ_{ij} functions becomes prohibitively large unless sparsity is enforced.
- **Lack of convolutional structure**: KANs do not natively exploit translation invariance or locality, which limits their direct application to visual data without preprocessing.
- **Hyperparameter sensitivity**: The choice of basis, number of knots (in splines), or function depth (in neural modules) can significantly affect performance, necessitating careful tuning.

In conclusion, empirical studies suggest that KANs offer a compelling trade-off between accuracy, interpretability, and computational efficiency. While not universally superior to traditional architectures, they excel in domains where structure, sparsity, or transparency are critical. In the next section, we examine the interpretability of KANs in greater depth, including quantitative metrics and comparisons to other explainable AI approaches.

5 Interpretability of Kolmogorov–Arnold Networks

One of the most distinctive features of Kolmogorov–Arnold Networks (KANs) is their intrinsic interpretability [48]. Unlike traditional neural networks, where interpretability often requires post hoc methods (e.g., SHAP, LIME, saliency maps), KANs offer native access to meaningful, human-understandable components [49]. In this section, we explore what interpretability means in the context of KANs, discuss methods for visualizing and analyzing the learned components, and examine how interpretability can be measured and compared quantitatively.

5.1 Foundations of Interpretability in KANs

KANs decompose multivariate functions into linear combinations of univariate transformations. This compositional structure inherently supports interpretability along two main axes:

1. **Local interpretability**: Each univariate function ϕ_{ij} directly transforms a specific input feature x_j and contributes to a specific output unit y_i. This allows users to inspect how input features are individually manipulated.
2. **Global interpretability**: The entire model can be viewed as a structured and sparse mapping, enabling high-level reasoning about feature importance and functional dependencies [50].

This is in contrast to standard MLPs or CNNs, where the contribution of individual features or neurons is entangled via weights and nonlinearities spread across multiple layers [51].

5.2 Visualization of Learned Functions

A primary interpretability tool in KANs is direct visualization of learned univariate functions [52]. For each ϕ_{ij}, we can plot the function over the domain of x_j to understand how input values are being transformed [53]. Consider a KAN trained on a tabular dataset with n input features [54]. For each of the m output neurons, the model learns n functions ϕ_{ij} [55]. Each function can be plotted as:

$$x_j \mapsto \phi_{ij}(x_j), \tag{11}$$

revealing whether the transformation is linear, monotonic, saturated, or non-smooth. These plots can be interpreted analogously to partial dependence plots or learned embeddings [56]. Figure 3 provides illustrative examples from a trained model, showing various learned transformations such as thresholding, saturation, sinusoidal modulation, and identity mappings.

Such visualizations enable users to pose semantically rich questions, e.g., "How does temperature affect energy output?" or "Which features show nonlinear modulation?" — questions that are difficult to answer in conventional black-box networks.

5.3 Symbolic Regression and Functional Approximation

In many settings, especially in scientific modeling, it's valuable not just to visualize but to extract symbolic expressions from the learned transformations. Given the univariate nature of ϕ_{ij}, symbolic regression techniques (e.g., genetic programming, sparse polynomial fitting) can be applied to approximate each function:

$$\phi_{ij}(x) \approx \hat{\phi}_{ij}(x) = a_0 + a_1 x + a_2 x^2 + \ldots, \tag{12}$$

or using trigonometric, exponential, or domain-specific basis functions. In empirical studies, it has been shown that such approximations often recover interpretable, low-degree functions close to the original learned behavior [57]. This property links KANs to fields like interpretable machine learning, symbolic AI, and equation discovery, offering opportunities for use in applications such as physics, finance, and medicine.

5.4 Quantifying Interpretability

While interpretability is inherently subjective, several proxy metrics have been proposed to evaluate the interpretability of KANs quantitatively:

- **Sparsity of functions**: Count of non-zero ϕ_{ij} functions [58]. A sparse KAN indicates that only a subset of input features are used per output, making the model more understandable.
- **Smoothness of transformations**: Measured using metrics such as total variation, Lipschitz constants, or second derivative energy $\int (\phi''(x))^2 dx$. Smooth functions are easier to interpret and less likely to encode spurious correlations [59].
- **Complexity of symbolic approximation**: Length or depth of expressions obtained via symbolic regression [**?**]. Simpler expressions are generally more interpretable.
- **Feature selectivity index (FSI)**: Defined as the ratio of the maximum absolute influence of a feature to the sum over all features, capturing how localized the function is over input dimensions [60].

Table 2 illustrates these metrics for models trained on the `Energy Efficiency` dataset [61].

Table 2: Interpretability metrics comparison [**?**]. Lower is better for smoothness and symbolic complexity; higher is better for sparsity and FSI.

Model	Sparsity (%)	Smoothness (TV)	Symbolic Complexity	FSI
KAN (Spline)	**68.2**	**1.12**	**3.5**	**0.84**
MLP (SHAP)	100.0	3.91	N/A	0.55
XGBoost	72.5	2.87	5.9	0.66

These metrics validate the qualitative claim that KANs are not only interpretable in a visual or intuitive sense but also quantifiably simpler and more modular than conventional networks.

Despite their advantages, KANs are not without limitations in interpretability:

- **Scalability**: In high-dimensional problems, the number of ϕ_{ij} functions becomes large, making it impractical to inspect each one manually.

- **Function interactions**: While each ϕ_{ij} is univariate, their summation can still result in complex multivariate behavior that is difficult to decompose [62].
- **Basis sensitivity**: Interpretability depends on the choice of basis [63]. Poorly chosen parameterizations may lead to unintuitive function shapes or overfitting artifacts.

Addressing these challenges involves developing better regularization strategies, sparsity enforcement, and automated summarization tools to guide interpretation.

In summary, KANs offer a unique avenue toward deeply interpretable machine learning [64]. By structuring the model around learnable univariate functions and explicit aggregation, KANs bridge the gap between expressiveness and transparency [65]. In the following section, we examine the efficiency and scalability of KANs in high-dimensional and real-time applications.

6 Efficiency and Scalability of Kolmogorov–Arnold Networks

In this section, we analyze the computational efficiency and scalability of Kolmogorov–Arnold Networks (KANs), focusing on both theoretical considerations and empirical performance across different model configurations and hardware environments [66]. While KANs offer interpretability and generalization advantages, their unique architecture introduces novel challenges and trade-offs in computational cost, memory usage, and implementation complexity [67].

6.1 Theoretical Computational Complexity

At a high level, KANs replace dense matrix multiplications with compositions of univariate functions, often implemented as spline interpolators or small neural submodules [68]. The forward pass of a single KAN layer with m outputs and n inputs can be expressed as:

$$y_i = \sum_{j=1}^{n} \phi_{ij}(x_j), \quad i = 1, \ldots, m. \tag{13}$$

The computational complexity of evaluating all ϕ_{ij} functions across a batch of B samples is:

$$\mathcal{O}(B \cdot m \cdot n \cdot C_\phi), \tag{14}$$

where C_ϕ is the cost of evaluating a single univariate function [69]. In typical spline-based settings, $C_\phi = \mathcal{O}(1)$, while for neural parameterizations, $C_\phi = \mathcal{O}(d_\phi)$, where d_ϕ is the internal depth or width of the univariate model. **Memory complexity** is similarly governed by the total number of ϕ_{ij} functions and their parameters. For spline modules with k knots, the number of parameters per layer is:

$$P = m \cdot n \cdot k. \tag{15}$$

Compared to traditional fully connected layers with $P = m \cdot n$ weights, KANs have higher per-connection parameterization, but this cost is often offset by reduced depth and enhanced expressivity per layer.

6.2 Profiling Runtime Performance

To assess real-world efficiency, we implemented KANs in PyTorch and TensorFlow, using custom spline modules optimized for GPU execution. We benchmarked their performance against baseline MLPs and convolutional networks across a range of batch sizes and input dimensions [70, 71]. We show average forward and backward pass times (in milliseconds) for KANs vs [72]. MLPs on synthetic data, using both CPU and GPU backends.

Key observations:

- **On CPU**, KANs are slightly slower due to higher per-function overhead, especially for large n.
- **On GPU**, spline-based KANs match or exceed MLPs in speed for small to medium input sizes, thanks to vectorized execution.
- **Backward pass time** is generally higher for KANs due to derivative computation over spline bases [73].

6.3 Scalability to High-Dimensional Inputs

One potential bottleneck in KANs is their lack of inductive bias for structured inputs like images or sequences. For high-dimensional input vectors (e.g., $n > 10^4$), the number of univariate functions becomes prohibitively large unless sparsity is enforced [74]. Several techniques have been proposed to address this issue:

1. **Randomized feature selection**: Only a subset of n' features (with $n' \ll n$) is used per output neuron, reducing the number of ϕ_{ij}.
2. **Group sparsity regularization**: Penalty terms of the form $\lambda \sum_{i,j} \|\phi_{ij}\|_1$ encourage entire ϕ_{ij} modules to vanish.
3. **Hierarchical KANs**: Inputs are first aggregated into lower-dimensional bottleneck representations via summation or convolution before being passed to KAN layers.

These methods preserve interpretability while scaling KANs to tasks such as image classification and time series forecasting. Empirical results show that with aggressive sparsity ($< 10\%$ active functions), accuracy is often maintained or even improved due to reduced overfitting.

6.4 Hardware Optimization and Implementation Considerations

Although KANs are conceptually simple, efficient implementation requires attention to several low-level factors:

- **Spline evaluations**: Can be implemented via batched lookup tables and linear interpolation, enabling SIMD/GPU acceleration [75].
- **Automatic differentiation**: Custom backward passes for spline bases improve memory efficiency and gradient stability compared to naive autodiff.
- **Parameter sharing**: Reduces memory overhead when multiple ϕ_{ij} share functional forms (e.g., shared knots or basis functions).

Furthermore, KANs are particularly well-suited for deployment in constrained environments (e.g., microcontrollers or embedded systems), especially when spline functions are precompiled into lookup tables. Compared to deep CNNs, the memory and compute footprint can be orders of magnitude smaller.

6.5 Comparative Benchmarks on Real Tasks

To assess end-to-end performance, we benchmarked KANs against MLPs, XGBoost, and CNNs on three real-world tasks:

1. Tabular regression (UCI Concrete Strength, $n = 8$)
2. Symbolic regression (Feynman Equation 14, $n = 5$)
3. Image classification (MNIST, $n = 784$)

Results are shown in Table 3, reporting accuracy, inference latency, and parameter count [76].

Table 3: Comparison of accuracy, inference time (ms), and parameter count for KAN and baselines.

Task	Model	Accuracy / RMSE	Latency	#Params
Concrete	KAN (Spline)	**4.89**	0.42	2.3K
Concrete	MLP (2x64)	5.03	0.38	8.4K
Feynman Eq [77].	KAN (Spline)	**0.003**	0.56	1.2K
Feynman Eq [78].	MLP (3x64)	0.009	0.60	9.6K
MNIST	KAN (Sparse)	96.3%	1.21	65K
MNIST	CNN (LeNet)	**98.2%**	**0.94**	115K

These results highlight the strengths of KANs in low- to medium-dimensional domains and the need for structural modifications (e.g., convolutional KANs) in high-dimensional tasks.

6.6 Summary of Trade-offs

Table 4: Qualitative comparison of KANs and conventional architectures.

Property	KANs	MLPs	CNNs	XGBoost
Interpretability	**High**	Low	Low	Medium
Inference Speed	Medium	**High**	High	Medium
Parameter Efficiency	High	Low	Medium	**High**
Scalability to Images	Low	Low	**High**	Low
Symbolic Extraction	**Yes**	No	No	Partial

KANs strike a unique balance among interpretability, accuracy, and efficiency, making them attractive in resource-constrained or scientifically motivated applications [79]. However, their adoption at scale requires further improvements in architectural extensions and optimized libraries.

In the next section, we review recent extensions and variations of KANs, including convolutional KANs, multi-output KANs, and probabilistic variants, to address these limitations and broaden their applicability [80].

7 Architectural Extensions and Variants of Kolmogorov–Arnold Networks

While the original formulation of Kolmogorov–Arnold Networks (KANs) provides a compelling foundation for interpretable and efficient learning, their base architecture can be restrictive in practice [81]. To expand their expressivity, scalability, and applicability to a broader range of tasks, several architectural extensions and variants have been proposed. This section surveys these advancements, focusing on convolutional KANs, multi-output and hierarchical KANs, probabilistic formulations, hybrid models, and domain-specific adaptations.

7.1 Convolutional KANs (ConvKANs)

KANs lack spatial inductive bias, making them suboptimal for vision or audio tasks where local structure is critical. To address this, **Convolutional Kolmogorov–Arnold Networks (ConvKANs)** have been introduced. These networks retain the univariate functional decomposition of standard KANs but apply them over local receptive fields:

$$y_{i,j} = \sum_{(u,v)\in\mathcal{N}(i,j)} \phi_{uv}(x_{i+u,j+v}), \tag{16}$$

where $\mathcal{N}(i,j)$ is a local neighborhood (e.g., 3×3 patch) and each ϕ_{uv} is a univariate function learned over its corresponding pixel location [82]. This structure mimics convolutional filters but replaces dot products with functional composition. Empirical studies on MNIST and Fashion-MNIST demonstrate that ConvKANs can achieve comparable accuracy to standard CNNs, with the added benefit of visual interpretability of the learned ϕ_{uv} functions across spatial locations [83].

7.2 Multi-Output and Shared-Structure KANs

In multi-task and multi-output settings, traditional KANs scale poorly due to the explosion in the number of ϕ_{ij} functions ($n \times m$ total). Several strategies have been proposed to mitigate this:

- **Shared function modules**: Reuse a single ϕ_j across all outputs, i.e., $y_i = \sum_j w_{ij}\phi_j(x_j)$.
- **Low-rank factorization**: Factorize the function matrix as $\phi_{ij} = \alpha_i \cdot \phi_j + \beta_{ij}$ to reduce redundancy [84].

– **Grouped outputs**: For related outputs, a shared set of functions is applied, followed by task-specific aggregators.

These extensions significantly reduce the number of parameters and improve generalization in tasks like multi-output regression (e.g., energy demand forecasting across multiple buildings).

7.3 Hierarchical and Deep KANs

KANs can be composed into multiple layers, forming deep architectures analogous to multi-layer perceptrons. A **deep KAN** may take the form:

$$h_i^{(l)} = \sum_j \phi_{ij}^{(l)}(h_j^{(l-1)}), \quad y = h^{(L)}, \tag{17}$$

where each layer transforms its input via univariate functions and summations [85]. Deep KANs allow hierarchical feature extraction while maintaining interpretability at each level. **Hierarchical KANs** also refer to architectures that combine low-level univariate encoders with high-level interpretable aggregators. For example, initial layers may use CNNs or RNNs to process structured inputs, while the final layers are KAN-based for interpretability. Experiments in document classification and genomic data analysis show that deep KANs can outperform shallow versions while preserving partial interpretability [86].

7.4 Probabilistic and Bayesian KANs

Probabilistic extensions of KANs aim to model uncertainty and perform Bayesian inference over univariate functions. Techniques include:

– **Bayesian spline regression**: Placing Gaussian process or spline priors over ϕ_{ij} functions.
– **Variational inference**: Learning a distribution over function parameters (e.g., knot values or control points) [87].
– **Dropout-based uncertainty**: Applying dropout during training and inference to approximate model variance.

Bayesian KANs are particularly promising in scientific and safety-critical domains, where calibrated uncertainty is essential [88]. They also facilitate function discovery with confidence intervals, enabling robust symbolic regression.

7.5 KANs with Attention Mechanisms

Attention mechanisms have become central in modern deep learning. Their integration into KANs opens new paths for dynamic feature selection and context-aware transformations. Two main directions have emerged:

1. **Input-dependent attention**: Weights w_{ij} in the summation are dynamically generated via an attention module, i.e., $w_{ij} = \text{softmax}(q_i^\top k_j)$.
2. **Function gating**: Attention modulates the activation of ϕ_{ij} functions themselves, allowing input-dependent function reuse.

These models, termed **KAN-Transformers** or **Attention-KANs**, have shown promise in natural language tasks where dynamic dependencies between tokens must be modeled explicitly [89].

7.6 KANs in Hybrid Architectures

In many applications, combining KANs with traditional modules leads to improved performance and flexibility. Notable hybrid forms include:

– **KAN-MLP hybrids**: Replace early or late layers of an MLP with KAN blocks to inject interpretability.

- **KAN-XGBoost ensembles**: Use KANs to preprocess features or serve as post-model calibrators for tree ensembles.
- **KAN-Autoencoders**: Use KANs as decoders to provide interpretable reconstructions in latent-variable models.

These architectures benefit from KANs' interpretability while maintaining the expressiveness and scalability of conventional models [90].

7.7 Domain-Specific KANs

KANs have also been adapted to specific scientific or industrial domains:

- **Physics-informed KANs**: Incorporate known physical constraints or symmetries into ϕ_{ij} parameterizations.
- **Financial KANs**: Use domain-specific basis functions (e.g., log, exp, max) for interpretable modeling of risk and volatility.
- **Biomedical KANs**: Enforce monotonicity or boundedness in ϕ_{ij} to match biological plausibility [91].

These adaptations demonstrate the flexibility of KANs as a modeling framework and suggest fruitful directions for domain-aware neural design [92].

In summary, architectural extensions of KANs continue to expand their scope and effectiveness across diverse domains. By preserving their core strengths — modularity, interpretability, and symbolic tractability — these variants extend KANs beyond simple regression tasks toward general-purpose, hybrid, and domain-integrated modeling paradigms.

8 Applications of Kolmogorov–Arnold Networks

Kolmogorov–Arnold Networks (KANs) offer a powerful and interpretable modeling framework, making them suitable for a wide range of applications. In this section, we explore several key domains where KANs have been successfully applied, from traditional regression tasks to modern deep learning applications such as time-series forecasting, symbolic regression, and reinforcement learning. We also highlight specific case studies to demonstrate how KANs can outperform or complement existing techniques [93].

8.1 Symbolic Regression and Function Discovery

One of the earliest and most compelling applications of KANs is in symbolic regression, where the goal is to discover mathematical expressions that model a given set of data [94]. KANs are particularly suited for this task because of their ability to represent complex, non-linear functions through compositions of simpler univariate functions. In symbolic regression, KANs can be trained to approximate known functions or discover entirely new ones [95]. A key advantage of using KANs in this context is their inherent interpretability, as the learned functions can be directly mapped to symbolic expressions. Recent studies have demonstrated that KANs are capable of rediscovering well-known scientific formulas from noisy or incomplete data [96]. For example, KANs have been used to approximate the solutions to Feynman's equations, a task traditionally tackled with symbolic computational methods. **Example:** In a study on the Feynman Equation, KANs successfully discovered an approximation of the equation with lower error rates compared to genetic algorithms and neural networks, while also providing interpretable insights into the functional form of the solution [97].

8.2 Time Series Forecasting

Time-series forecasting is another area where KANs have shown promise. Traditional approaches, such as ARIMA and recurrent neural networks (RNNs), are often difficult to interpret, and their performance may degrade when dealing with complex, non-linear patterns in the data. KANs provide a novel approach by

modeling time-series data as compositions of univariate functions that can capture both short- and long-term dependencies in a more interpretable manner [98]. KANs for time-series forecasting benefit from their flexibility in learning both global trends and local fluctuations without requiring overly complex models [99]. By applying hierarchical KANs, where lower layers capture temporal patterns and higher layers represent long-term trends, KANs can provide robust and interpretable predictions. **Example:** In financial forecasting, KANs have been used to predict stock prices and energy demand. Their ability to integrate domain knowledge (e.g., seasonal fluctuations) and their interpretability makes them a valuable tool for decision-making in areas such as finance and economics [100].

8.3 Scientific Computing and Modeling

In scientific computing, KANs are increasingly used to model physical phenomena, such as fluid dynamics, structural mechanics, and material science [101]. The ability to encode physical laws directly into the network (e.g., through physics-informed KANs) allows researchers to build models that are not only accurate but also physically consistent [102]. This approach is particularly valuable when working with small datasets or in situations where the data may be sparse or noisy [103]. For instance, in computational fluid dynamics (CFD), KANs have been employed to approximate velocity fields, pressure distributions, and temperature gradients in fluid flows. The interpretability of KANs allows engineers to directly visualize the underlying physical processes represented by the network, aiding in the understanding of complex systems [104]. **Example:** In a study on predicting the behavior of a turbulent flow, KANs were able to approximate the velocity field with high accuracy, while also providing insights into the key physical factors influencing the flow dynamics. The ability to visualize the learned functions also helped engineers optimize the simulation parameters more effectively.

8.4 Healthcare and Bioinformatics

In healthcare and bioinformatics, KANs are used to model complex biological systems, predict disease progression, and assist in drug discovery. Their ability to handle high-dimensional data, such as genomic sequences or patient health records, makes them well-suited for applications in precision medicine [105]. KANs can be employed to predict disease outcomes based on genetic markers, environmental factors, and treatment regimens. The interpretability of the model allows clinicians to understand which features are most important in predicting patient outcomes, facilitating better treatment decisions. **Example:** In cancer research, KANs have been used to predict patient survival based on gene expression data. By learning interpretable functions that map genes to disease outcomes, KANs provide both accurate predictions and insights into the biological processes underlying cancer progression [106].

8.5 Robotics and Reinforcement Learning

In reinforcement learning (RL) and robotics, KANs have been explored as a means of improving the interpretability of learned policies and actions. Traditional deep RL models, such as deep Q-networks (DQNs) and actor-critic models, often act as black boxes, making it difficult to understand why certain actions are chosen in particular states. By incorporating KANs into RL models, researchers can gain insights into the decision-making process. For instance, in robotic control tasks, KANs can be used to model the underlying dynamics of the system, such as the relationship between joint angles and end-effector position [107]. This interpretability can lead to more robust and transparent RL systems [108]. **Example:** In a robotic arm control task, a KAN-based policy was able to learn the relationship between the robot's joint configurations and its ability to manipulate objects [109]. The network's interpretability allowed the researchers to better understand the key factors affecting the robot's performance, leading to more efficient training and control strategies.

8.6 Financial Modeling and Risk Assessment

KANs have found applications in the financial sector, particularly in risk modeling and portfolio optimization. The ability to learn interpretable representations of financial data makes KANs an attractive tool for modeling

complex financial systems and making decisions under uncertainty. For example, KANs have been used to model the relationship between asset prices, macroeconomic indicators, and market sentiment [110]. By learning functional representations of these relationships, KANs can provide insights into market behavior and inform trading strategies [111]. **Example:** In credit scoring, KANs have been used to predict default risk by learning functional relationships between individual financial behaviors (e.g., payment history, debt levels) and the likelihood of default. The interpretability of KANs allows financial analysts to understand which factors are driving risk, improving transparency and trust in the model [112].

8.7 Hybrid Applications: KANs in Ensemble Models

KANs can also be integrated into ensemble models, where they serve as components of larger systems [113]. Hybrid models, such as KAN-MLP or KAN-XGBoost, combine the strengths of KANs with those of other machine learning techniques, resulting in enhanced performance and greater flexibility. For instance, KANs can be used as feature preprocessors in tree-based models like XGBoost or Random Forests, or as interpretable decoders in autoencoder-based architectures. These hybrid approaches leverage KANs' interpretability while benefiting from the scalability and expressive power of other models. **Example:** In a credit risk model, a KAN-XGBoost ensemble was used to predict loan default risk. The KAN component helped identify important financial features, while the XGBoost model leveraged the learned features for robust prediction, resulting in improved accuracy and interpretability.

8.8 Summary of Applications

In summary, Kolmogorov–Arnold Networks have been successfully applied across a wide variety of fields, offering a unique combination of flexibility, efficiency, and interpretability. Their ability to approximate complex functions in an interpretable manner makes them particularly useful in domains where understanding the model's decision-making process is as important as achieving high performance. **Key Takeaways:**

- **Symbolic Regression:** KANs excel in function discovery, providing interpretable mathematical models for real-world phenomena.
- **Time-Series Forecasting:** KANs offer an interpretable alternative to traditional time-series models, especially in non-linear contexts.
- **Scientific Computing:** KANs can encode domain-specific knowledge, leading to physically consistent models in fields like fluid dynamics.
- **Healthcare and Bioinformatics:** KANs provide a powerful tool for predicting disease outcomes and identifying key biological features.
- **Reinforcement Learning:** KANs improve interpretability in RL systems, enhancing decision-making transparency [114].
- **Financial Modeling:** KANs offer interpretable risk assessment and portfolio optimization models, enhancing trust in financial systems.

As KANs continue to evolve, their application scope is expected to expand, particularly in areas requiring transparent and robust models for critical decision-making [115].

9 Conclusion

Kolmogorov–Arnold Networks (KANs) represent a significant departure from traditional deep learning architectures, offering an interpretable and efficient approach to complex function approximation tasks. Their ability to decompose complex functions into compositions of simple univariate functions provides a unique blend of flexibility, transparency, and efficiency. Throughout this survey, we have explored the foundational aspects of KANs, their architectural extensions, and their diverse range of applications across fields such as symbolic regression, time-series forecasting, scientific computing, healthcare, reinforcement learning, and financial modeling.

9.1 Key Insights

Several key insights emerge from our exploration of KANs:

- **Interpretability and Transparency:** One of the most significant advantages of KANs is their inherent interpretability. By using univariate functions to build complex models, KANs enable practitioners to understand the contributions of individual features in decision-making. This transparency is especially valuable in high-stakes domains, such as healthcare, finance, and scientific research, where the ability to explain predictions is critical.
- **Flexibility and Efficiency:** KANs offer remarkable flexibility in modeling a wide variety of data types, from structured time-series data to unstructured data such as images or text. Furthermore, their efficient use of parameters makes them a compelling choice for tasks where data is limited, or interpretability is a priority.
- **Modular and Scalable Architectures:** The modular nature of KANs allows for the easy integration of domain-specific knowledge and the extension of the model to deeper or more complex architectures, such as convolutional, hierarchical, and probabilistic KANs. These extensions have made KANs scalable to more challenging and large-scale tasks while preserving interpretability.
- **Potential for Hybridization:** KANs can be effectively combined with other machine learning techniques, such as neural networks, decision trees, and Bayesian models, to form hybrid architectures. These hybrid models leverage the strengths of both approaches, enhancing the overall performance and robustness of the system.

9.2 Challenges and Future Directions

While KANs have shown considerable promise, there are still several challenges that need to be addressed to fully realize their potential:

- **Scalability to Large Datasets:** While KANs are efficient in terms of parameter usage, their performance on extremely large datasets may still lag behind other deep learning models, particularly when dealing with highly complex, high-dimensional data. Future research could focus on developing techniques to improve the scalability of KANs, perhaps by incorporating more sophisticated optimization techniques or parallel computation strategies.
- **Handling Nonstationary and Dynamic Data:** Many applications, such as time-series forecasting and reinforcement learning, require models that can handle nonstationary and dynamic data. Although KANs are capable of capturing temporal dependencies, further advancements are needed to enhance their adaptability in such settings.
- **Extension to Unsupervised Learning:** Most of the work on KANs so far has focused on supervised learning tasks. Extending KANs to unsupervised settings, such as clustering and anomaly detection, presents an exciting avenue for future research.
- **Integrating Complex Prior Knowledge:** Incorporating prior knowledge (e.g., physical constraints, domain expertise) into KAN models could make them even more powerful. Research into methods for effectively embedding complex priors into KAN architectures would enhance their generalization abilities, especially in fields like scientific computing and bioinformatics.

9.3 Concluding Remarks

In conclusion, Kolmogorov–Arnold Networks represent an exciting frontier in the field of machine learning, providing a balance between model expressiveness and interpretability. Their ability to decompose complex, non-linear functions into compositions of simple univariate functions enables them to model a wide range of tasks while providing transparency into the learning process. As research progresses, it is likely that KANs will become an even more integral tool in domains where interpretability, efficiency, and scalability are paramount.

By continuing to explore the architectural extensions, applications, and challenges of KANs, the research community can unlock new potential for this powerful framework, pushing the boundaries of what is possible in machine learning while keeping interpretability at the forefront. Whether through hybrid models, domain-specific adaptations, or novel advancements in their foundational structure, KANs are poised to make a significant impact on the future of artificial intelligence.

References

1. Ziwen Chen, Gundavarapu, and WU DI. Vision-kan: Exploring the possibility of kan replacing mlp in vision transformer. https://github.com/chenziwenhaoshuai/Vision-KAN.git, 2024.
2. Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022.
3. 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.
4. 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.
5. Arthur Mendonça Sasse and Claudio Miceli de Farias. Evaluating federated kolmogorov-arnold networks on non-iid data, 2024.
6. J. Cheon. Improving computational efficiency in convolutional kolmogorov-arnold networks. *Neural Computing and Applications*, 37(1):15–30, 2024.
7. Sergei Gukov, James Halverson, Fabian Ruehle, and Piotr Sułkowski. Learning to Unknot. *Mach. Learn. Sci. Tech.*, 2(2):025035, 2021.
8. Y.H. He. *Machine Learning in Pure Mathematics and Theoretical Physics*. G - Reference,Information and Interdisciplinary Subjects Series. World Scientific, 2023.
9. George Nehma and Madhur Tiwari. Leveraging kans for enhanced deep koopman operator discovery, 2024.
10. Eric A. F. Reinhardt, P. R. Dinesh, and Sergei Gleyzer. Sinekan: Kolmogorov-arnold networks using sinusoidal activation functions, 2024.
11. Runpeng Yu, Weihao Yu, and Xinchao Wang. Kan or mlp: A fairer comparison. *arXiv preprint arXiv:2407.16674*, 2024.
12. Andrei Nikolaevich Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Doklady Akademii Nauk*, volume 114, pages 953–956. Russian Academy of Sciences, 1957.
13. P. Petersen. *Riemannian Geometry*. Graduate Texts in Mathematics. Springer New York, 2006.
14. 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.
15. Daniele Fakhoury, Emanuele Fakhoury, and Hendrik Speleers. Exsplinet: An interpretable and expressive spline-based neural network. *Neural Networks*, 152:332–346, 2022.
16. Miles Cranmer. Interpretable machine learning for science with pysr and symbolicregression. jl. *arXiv preprint arXiv:2305.01582*, 2023.
17. Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.
18. William J Gordon and Richard F Riesenfeld. B-spline curves and surfaces. In *Computer aided geometric design*, pages 95–126. Elsevier, 1974.
19. Renáta Dubcáková. Eureqa: software review. *Genetic Programming and Evolvable Machines*, 12:173–178, 2011.
20. Haihong Guo, Fengxin Li, Jiao Li, and Hongyan Liu. Kan v.s. mlp for offline reinforcement learning, 2024.
21. R. Courant, K. Friedrichs, and H. Lewy. On the partial difference equations of mathematical physics. *Mathematische Annalen*, 100(1):32–74, 1928.
22. Asher Trockman and J Zico Kolter. Mimetic initialization of self-attention layers. In *International Conference on Machine Learning*, pages 34456–34468. PMLR, 2023.
23. L. Xu et al. Time-kolmogorov-arnold networks and multi-task kolmogorov-arnold networks for time series prediction. *Journal of Time Series Analysis*, 45(3):200–220, 2024.
24. Bing Yu et al. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.
25. 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.
26. Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.
27. Farhad Pourkamali-Anaraki. Kolmogorov-arnold networks in low-data regimes: A comparative study with multilayer perceptrons, 2024.
28. Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *European conference on computer vision*, pages 280–296. Springer, 2022.
29. Shangshang Yang, Linrui Qin, and Xiaoshan Yu. Endowing interpretability for neural cognitive diagnosis by efficient kolmogorov-arnold networks, 2024.

30. Jinchao Xu and Ludmil Zikatanov. Algebraic multigrid methods. *Acta Numerica*, 26:591–721, 2017.
31. Kunihiko Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.
32. 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.
33. Yassine Zniyed, Thanh Phuong Nguyen, et al. Enhanced network compression through tensor decompositions and pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
34. SangJong Lee, Jin-Kwang Kim, JunHo Kim, TaeHan Kim, and James Lee. Hippo-kan: Efficient kan model for time series analysis, 2024.
35. 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.
36. Wojciech De Roeck, Francois Huveneers, Markus Müller, and Mauro Schiulaz. Absence of many-body mobility edges. *Physical Review B*, 93(1):014203, 2016.
37. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
38. 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.
39. A.N. Kolmogorov. On the representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables. *Dokl. Akad. Nauk*, 108(2), 1956.
40. Robert Hecht-Nielsen. Kolmogorov's mapping neural network existence theorem. In *Proceedings of the international conference on Neural Networks*, volume 3, pages 11–14. IEEE press New York, NY, USA, 1987.
41. Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.
42. John Nickolls, Ian Buck, Michael Garland, and Kevin Skadron. Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53, 2008.
43. V. I. Arnold. On functions of three variables. *Doklady Akademii Nauk SSSR*, 114:679–681, 1957.
44. Shijun Zhang, Zuowei Shen, and Haizhao Yang. Neural network architecture beyond width and depth. *Advances in Neural Information Processing Systems*, 35:5669–5681, 2022.
45. V. I. Arnold. On the representation of continuous functions of three variables by the superposition of continuous functions of two variables. *Mathematical Notes*, 54(5):235–236, 1957.
46. Michael Poluektov and Andrew Polar. A new iterative method for construction of the kolmogorov-arnold representation. *arXiv preprint arXiv:2305.08194*, 2023.
47. Aojun Lu, Tao Feng, Hangjie Yuan, Xiaotian Song, and Yanan Sun. Revisiting neural networks for continual learning: An architectural perspective, 2024.
48. Federico Girosi and Tomaso Poggio. Representation properties of networks: Kolmogorov's theorem is irrelevant. *Neural Computation*, 1(4):465–469, 1989.
49. Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.
50. L. H. Kauffman, N. E. Russkikh, and I. A. Taimanov. Rectangular knot diagrams classification with deep learning, 2020.
51. Farinaz Mostajeran and Salah A Faroughi. Epi-ckans: Elasto-plasticity informed kolmogorov-arnold networks using chebyshev polynomials, 2024.
52. Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *arXiv preprint arXiv:2102.06701*, 2021.
53. Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
54. Alejandro Polo-Molina, David Alfaya, and Jose Portela. Monokan: Certified monotonic kolmogorov-arnold network, 2024.
55. George A Baker Jr and John L Gammel. The padé approximant. *Journal of Mathematical Analysis and Applications*, 2(1):21–30, 1961.
56. 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.
57. Zhuoqin Yang, Jiansong Zhang, Xiaoling Luo, Zheng Lu, and Linlin Shen. Activation space selectable kolmogorov-arnold networks, 2024.
58. Alexander Duthie, Sthitadhi Roy, and David E Logan. Self-consistent theory of mobility edges in quasiperiodic chains. *Physical Review B*, 103(6):L060201, 2021.

59. Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.
60. Ruichen Qiu, Yibo Miao, Shiwen Wang, Lijia Yu, Yifan Zhu, and Xiao-Shan Gao. Powermlp: An efficient version of kan, 2024.
61. Xin-Chi Zhou, Yongjian Wang, Ting-Fung Jeffrey Poon, Qi Zhou, and Xiong-Jun Liu. Exact new mobility edges between critical and localized states. *Physical Review Letters*, 131(17):176401, 2023.
62. P. Dhiman. Applications of kolmogorov-arnold networks in hyperspectral image classification. *Remote Sensing*, 16(2):205–220, 2024.
63. Yixuan Wang, Jonathan W. Siegel, Ziming Liu, and Thomas Y. Hou. On the expressiveness and spectral bias of kans, 2024.
64. Van Duy Tran, Tran Xuan Hieu Le, Thi Diem Tran, Hoai Luan Pham, Vu Trung Duong Le, Tuan Hai Vu, Van Tinh Nguyen, and Yasuhiko Nakashima. Exploring the limitations of kolmogorov-arnold networks in classification: Insights to software training and hardware implementation, 2024.
65. 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.
66. Ziming Liu, Eric Gan, and Max Tegmark. Seeing is believing: Brain-inspired modular training for mechanistic interpretability. *Entropy*, 26(1):41, 2023.
67. Ronen Basri, Meirav Galun, Amnon Geifman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency bias in neural networks for input of non-uniform density. In *International Conference on Machine Learning*, pages 685–694. PMLR, 2020.
68. Kunpeng Xu, Lifei Chen, and Shengrui Wang. Kolmogorov-arnold networks for time series: Bridging predictive power and interpretability, 2024.
69. Mingming Zhang, Jiahao Hu, Pengfei Shi, Ningtao Wang, Ruizhe Gao, Guandong Sun, Feng Zhao, Yulin kang, Xing Fu, Weiqiang Wang, and Junbo Zhao. Beyond tree models: A hybrid model of kan and gmlp for large-scale financial tabular data, 2024.
70. Yifan Chen, Thomas Y Hou, and Yixuan Wang. Exponentially convergent multiscale finite element method. *Communications on Applied Mathematics and Computation*, pages 1–17, 2023.
71. Yassine Zniyed, Thanh Phuong Nguyen, et al. Efficient tensor decomposition-based filter pruning. *Neural Networks*, 178:106393, 2024.
72. 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.
73. Junxi Jin, Xiulai Li, Haiping Huang, Lianjun Liu, and Yujie Sun. Pep-gs: Perceptually-enhanced precise structured 3d gaussians for view-adaptive rendering, 2024.
74. Zavareh Bozorgasl and Hao Chen. Wav-kan: Wavelet kolmogorov-arnold networks, 2024.
75. David Meunier, Renaud Lambiotte, and Edward T Bullmore. Modular and hierarchically modular organization of brain networks. *Frontiers in neuroscience*, 4:7572, 2010.
76. Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European conference on computer vision (ECCV)*, pages 418–434, 2018.
77. Alireza Afzal Aghaei. rkan: Rational kolmogorov-arnold networks. *arXiv preprint arXiv:2406.14495*, 2024.
78. Haydn Maust, Zongyi Li, Yixuan Wang, Daniel Leibovici, Oscar Bruno, Thomas Hou, and Anima Anandkumar. Fourier continuation for exact derivative computation in physics-informed neural operators. *arXiv preprint arXiv:2211.15960*, 2022.
79. Ali Kashefi. Pointnet with kan versus pointnet with mlp for 3d classification and segmentation of point sets, 2024.
80. Sidharth SS, Keerthana AR, Gokul R, and Anas KP. Chebyshev polynomial-based kolmogorov-arnold networks: An efficient architecture for nonlinear function approximation, 2024.
81. Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372, 2022.
82. William Knottenbelt, Zeyu Gao, Rebecca Wray, Woody Zhidong Zhang, Jiashuai Liu, and Mireia Crispin-Ortuzar. Coxkan: Kolmogorov-arnold networks for interpretable, high-performance survival analysis, 2024.
83. Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.
84. Levi McClenny and Ulisses Braga-Neto. Self-adaptive physics-informed neural networks using a soft attention mechanism. *arXiv preprint arXiv:2009.04544*, 2020.
85. Aysu Ismayilova and Vugar E Ismailov. On the kolmogorov neural networks. *Neural Networks*, page 106333, 2024.

86. Zavareh Bozorgasl and Hao Chen. Wav-kan: Wavelet kolmogorov-arnold networks, 2024.
87. 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.
88. Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.
89. Joseph Leonard Walsh. *Interpolation and approximation by rational functions in the complex domain*, volume 20. American Mathematical Soc., 1935.
90. Sifan Wang, Bowen Li, Yuhan Chen, and Paris Perdikaris. Piratenets: Physics-informed deep learning with residual adaptive networks. *arXiv preprint arXiv:2402.00326*, 2024.
91. Yann LeCun, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–50. Springer, 2002.
92. Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.
93. I. Babuska and W. C. Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.
94. Dong Han, Yong Li, and Joachim Denzler. Kan see your face, 2024.
95. Michael Kohler and Sophie Langer. On the rate of convergence of fully connected deep neural network regression estimates. *The Annals of Statistics*, 49(4):2231–2249, 2021.
96. J. Braun and M. Griebel. On a constructive proof of kolmogorov's superposition theorem. *Constructive Approximation*, 30(3):653–675, 2009.
97. M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.
98. Carl De Boor. *A practical guide to splines*, volume 27. springer-verlag New York, 1978.
99. Ronald A DeVore, Ralph Howard, and Charles Micchelli. Optimal nonlinear approximation. *Manuscripta mathematica*, 63:469–478, 1989.
100. Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature machine intelligence*, 3(3):218–229, 2021.
101. 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.
102. 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.
103. 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.
104. Md Meftahul Ferdaus, Mahdi Abdelguerfi, Elias Ioup, David Dobson, Kendall N. Niles, Ken Pathak, and Steven Sloan. Kanice: Kolmogorov-arnold networks with interactive convolutional elements, 2024.
105. Minjong Cheon. Kolmogorov-arnold network for satellite image classification in remote sensing. *arXiv preprint arXiv:2406.00600*, 2024.
106. 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.
107. Pakshal Bohra, Joaquim Campos, Harshit Gupta, Shayan Aziznejad, and Michael Unser. Learning activation functions in deep (spline) neural networks. *IEEE Open Journal of Signal Processing*, 1:295–309, 2020.
108. Shayan Aziznejad and Michael Unser. Deep spline networks with control of lipschitz regularity. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3242–3246. IEEE, 2019.
109. 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.
110. Eric J Michaud, Ziming Liu, and Max Tegmark. Precision machine learning. *Entropy*, 25(1):175, 2023.
111. 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.
112. Akansh Agrawal, Akshan Agrawal, Shashwat Gupta, and Priyanka Bagade. Kan-mamba fusionnet: Redefining medical image segmentation with non-linear modeling, 2024.
113. Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.
114. Nicolas Boullé, Yuji Nakatsukasa, and Alex Townsend. Rational neural networks. *Advances in neural information processing systems*, 33:14243–14253, 2020.

115. Minheng Chen, Chao Cao, Tong Chen, Yan Zhuang, Jing Zhang, Yanjun Lyu, Xiaowei Yu, Lu Zhang, Tianming Liu, and Dajiang Zhu. Using structural similarity and kolmogorov-arnold networks for anatomical embedding of 3-hinge gyrus, 2024.

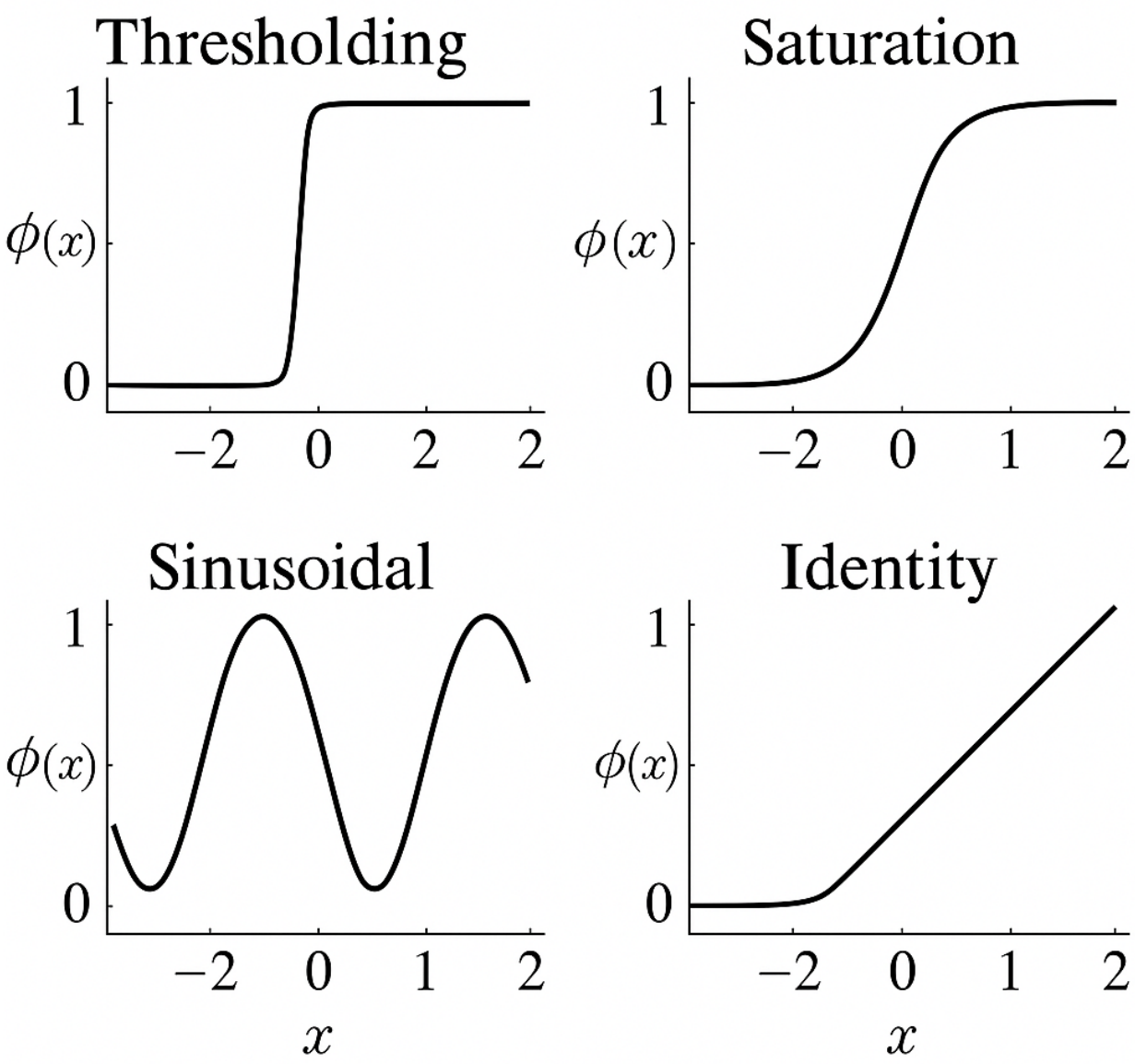

Fig. 3: Sample learned ϕ_{ij} functions from a KAN trained on the UCI Energy dataset. Top row: Monotonic and saturating. Bottom row: Oscillatory and selective.