Kolmogorov-Arnold Networks

JS Vishnu Teja **L Sai Harsh** **K Bhargav Sashank** **Kunjan Manoj Kumar S**

Abstract

Kolmogorov-Arnold Networks (KANs) present a novel alternative to traditional Multi-Layer Perceptrons (MLPs), drawing inspiration from the Kolmogorov-Arnold Representation theorem rather than the universal approximation theorem (3). KANs mainly differ from MLPs by replacing fixed activation functions at nodes with learnable univariate functions on edges, implemented via splines. This shift leads to a significant improvements in both accuracy and interpretability which shows KANs are better than MLPs in certain aspects.

1 Introduction

The KAN paper (1) claims that theoretically and empirically, KANs possess faster neural scaling laws than MLPs (1). For interpretability, KANs can be intuitively visualized and can easily interact with human users (1). KANs seem to be a promising alternative for MLPs.Multi-Layer Perceptrons (MLPs) are inspired by the universal approximation theorem. We instead focus on the Kolmogorov-Arnold representation theorem, which can be realized by a new type of neural network called Kolmogorov-Arnold networks (KAN). We review the Kolmogorov-Arnold theorem, to inspire the design of Kolmogorov-Arnold Networks.

Kolmogorov–Arnold Representation Theorem (KART). *If $f : [0,1]^n \longrightarrow \mathbb{R}$ is a multivariate continuous function, then f can be represented as a composition of continuous univariate functions (1),*

$$f(\mathbf{x}) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right),$$

where $\mathbf{x} = (x_1, x_2, \cdots, x_n)$; $\phi_{q,p} : [0,1] \longrightarrow \mathbb{R}$ and $\Phi_q : \mathbb{R} \longrightarrow \mathbb{R}$ are continuous functions.

2 Methodology

KANs utilize univariate continuous learnable activation functions in place of traditional fixed nonlinearities. These activations are modeled using B-spline functions, enabling the network to flexibly approximate complex functional relationships while maintaining interpretability.

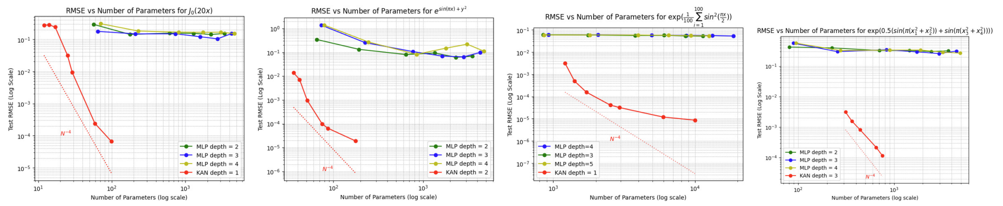
2.1 B-splines as Activation Functions in KANs

In KANs, each edge in the computational graph is assigned an activation function parameterized as a linear combination of B-spline basis functions (2) . That is, a univariate function $\phi(x)$ is used in place of weights, given by:

$$\phi(x) = \textstyle\sum_i c_i B_{i,k}(x)$$

where c_i are the learnable coefficients and $B_{i,k}(x)$ are the spline basis functions of order k. Optionally, KANs add a residual nonlinear function (e.g., SiLU) scaled by a learnable parameter:

$$\phi(x) = w_b.\text{silu}(x) + w_s \textstyle\sum_i c_i B_{i,k}(x)$$



(a) $f_1(x)$ (b) $f_2(x, y)$ (c) $f_4(x_1, x_2, \ldots, x_{100})$ (d) $f_5(x_1, x_2, x_3, x_4)$

Figure 1: Compared KANs to MLPs on five toy examples. KANs can saturate at a faster scaling bound as predicted by our theory ($\alpha = 4$), while MLPs scales slowly and plateau quickly.

The choice of B-splines A.1 offers several advantages: smoothness, local control (due to compact support), and differentiability, making them ideal for use within backpropagation-based training with LBFGS optimizer A.2.

3 Experiments

We validated KANs empirically by replicating the toy experiments from Section 3.1 of the original paper, comparing their function-fitting performance with MLPs, with a focus on scaling, generalization, and compositional structure.

3.1 Neural Scaling Laws

The resulting log-log plots reveal that KANs consistently exhibit better neural scaling behavior than MLPs, particularly as input dimensionality increases. This indicates that KANs are more efficient in utilizing additional parameters and are less affected by the curse of dimensionality compared to traditional MLPs. To investigate neural scaling laws, we implemented both KANs and standard MLPs across a series of toy datasets. Our objective was to study how the model performance scales with the number of parameters and to compare the neural scaling behavior of KANs and MLPs (Figure 1,5).

3.1.1 Experimental Setup

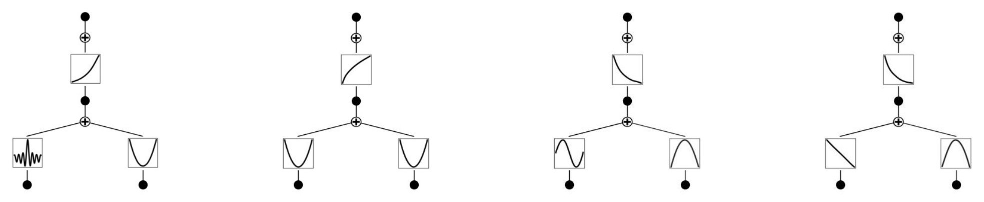
For each dataset, we trained both KAN and MLP models with varying numbers of parameters. We then measured the root mean squared error (RMSE) on a held-out test set and plotted $\log(\text{RMSE})$ against $\log(\text{Number of Parameters})$ to examine the neural scaling behavior. We considered a suite of synthetic functions with varying dimensionality and compositional complexity including the functions in A.3.

3.2 Interpretability in KANs

KANs stand out for their inherent **interpretability**, setting them apart from MLPs. They leverage the **KART**, which expresses any multivariate continuous function as a composition of univariate functions and addition. As illustrated in Figure 2,6,7,8, KANs construct functions via:

- **Univariate functional transformations**: Inputs are transformed using learned 1D functions (i.e. B-splines) .
- **Additive combinations**: These univariate outputs are then summed together at intermediate nodes.
- **Hierarchial composition**: Complex functions are recursively built from simpler, interpretable components.

Each learned function in the network can be visualized and interpreted independently, allowing us to understand how the model transforms and combines inputs to produce outputs. This is in stark contrast to traditional neural networks where the internal feature representations are opaque and difficult to analyze. To evaluate the interpretability of KANs, we implemented KAN models on a series of toy datasets, each representing known univariate or multivariate functional forms.



(a) $f_6(x, y)$ (b) $f_7(x, y)$ (c) $f_8(x, y)$ (d) $f_9(x, y)$

Figure 2: Visualization of function composition in KANs. Each branch represents a learned univariate function, and the nodes represent summation.

The objective was to assess whether KANs can accurately learn and replicate these functions in an interpretable manner. We trained the models on the functions f_6 to f_{11} in A.3.
Each KAN was trained using a small dataset sampled from the respective function's domain. After training, we visualized the learned univariate functions to demonstrate the inherent interpretability of KANs by verifying whether the model captured the functional forms accurately and represented them as compositions of simpler univariate functions. These experiments validate the potential of KANs to not only approximate complex functions, but also to do so in a way that preserves interpretability—a key advantage over traditional black-box neural networks.

3.3 Are KANs promising alternatives for MLPs ?

We aimed to demonstrate that **KANs** are promising alternatives to traditional **MLPs**, especially in terms of interpretability and efficiency. To support this claim, we conducted the following experiments:

1. **Function Approximation:** We implemented and trained KANs on selected *Feynman functions*, as referenced in the original KAN paper. These functions serve as benchmarks for symbolic regression tasks.
2. **Classification Task:** We conducted a series of experiments on a subset of MNIST dataset (given in Question 2 of Assignment 2). The dataset consists of 10,000(8,000 for training and 2,000 for testing) grayscale images of handwritten digits, categorized into 10 distinct classes (digits 0 through 9).

The results in Table 1 indicate that **KANs can achieve similar or better performance with fewer parameters and improved interpretability**, particularly in symbolic or low-dimensional domains.

Empirical formula	KAN Shape	KAN Params	MLP Params	KAN Error	MLP Error	KAN Time(sec)	MLP Time(sec)
$\sqrt{1+a^2+b^2}$	[2,1,1]	144	4417	1.17×10^{-3}	4.17×10^{-4}	45.17	6.31
$a + \alpha b$	[3,3,1]	204	4481	7.32×10^{-5}	2.55×10^{-5}	40.25	8.01
$n \ln(a)$	[2,2,1]	72	4417	3.29×10^{-5}	4.17×10^{-5}	16.86	32.26
$a(\frac{1}{b} - 1)$	[2,2,1]	72	4417	6.34×10^{-5}	4.51×10^{-5}	15.85	10.69

Table 1: Comparison between KAN and MLP on different Feymann Datasets

Though KANs have lesser number of parameters than MLPs the results are comparable to comparable to that of MLPs, but the time taken to train for KANs is much more than that for MLPs.

MNIST We implemented KANs and MLPs on a subset of MNIST dataset. To reduce computational complexity and improve efficiency, we applied Principal Component Analysis (PCA) to project the original 28×28 pixel images to a 100-dimensional space before training the models. We have used the a 3-layer KAN with different widths and compared it with 2-layer and 3-layer MLPs of different width and compared the accuracy by plotting in Figure 3 accuracy v/s number of parameters.
KANs do not seem to perform well in classification tasks, especially ones that do not have any "perfect" mathematical structure to them.

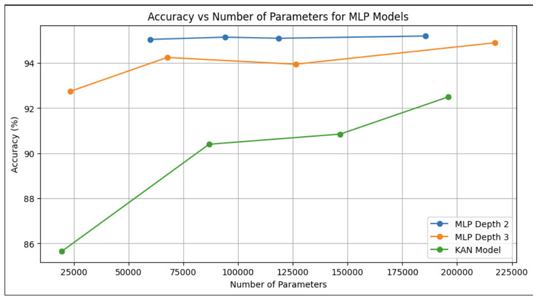
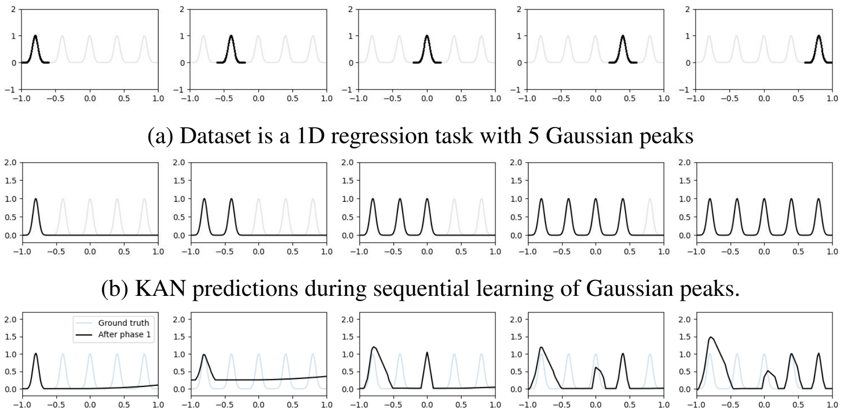


Figure 3: Accuracy v/s Number of parameters on a subset of MNIST dataset



(a) Dataset is a 1D regression task with 5 Gaussian peaks

(b) KAN predictions during sequential learning of Gaussian peaks.

(c) MLP predictions during the same task, showing catastrophic forgetting.

Figure 4: Comparison of KAN and MLP predictions on a sequential Gaussian peak learning task, demonstrating catastrophic forgetting in MLPs and locality-preserving behavior in KANs.

3.4 Catastrophic Forgetting

Catastrophic forgetting is a major challenge in traditional neural networks like MLPs, where learning new tasks leads to forgetting previously acquired knowledge (1). In contrast, KANs leverage the local nature of spline basis functions, which ensures that updates during training affect only limited regions of the parameter space. As shown in Figure 4b and Figure 4c, KANs retain previously learned regions even after sequential exposure to new data, unlike MLPs which tend to overwrite prior knowledge globally. This local plasticity of KANs makes them more biologically plausible and robust to sequential learning.
To illustrate catastrophic forgetting, a 1D regression task with 5 Gaussian peaks is constructed. Instead of presenting all data simultaneously, data from each peak is shown sequentially. After each training phase, model predictions are evaluated. This setup highlights how KANs preserve previously learned regions, whereas MLPs exhibit catastrophic forgetting by altering predictions even in regions with no new data. Further details can be found in A.4

4 Conclusion

In this project, we explored the suitability of KANs versus MLPs across different scenarios. From the results, we understand that KANs excel when the task involves compositional structure, complex functions, and they offer better interpretability. KANs are better than MLPs especially in tasks that deal with continual learning or require small model size. Though KANs are better in many of the tasks, MLPs still remain competitive in terms of speed of training and general-purpose use. Hence, the choice between KANs and MLPs depends on the specific demands of accuracy, interpretability, and efficiency, as summarized in the decision flow 9.
All the implementations are in the github repository

References

[1] Ziming Liu, Yixuan Wang, Sachin Vaidya1 Fabian Ruehle, James Halverson, Marin Soljačić , Thomas Y. Hou, Max Tegmark *Kolmogorov–Arnold Networks*, arXiv preprint arXiv:2404.19756, 2024.

[2] C. de Boor, *A Practical Guide to Splines*, Springer-Verlag, 1978.

[3] Tianrui Ji, Yuntian Hou, Di Zhang *A COMPREHENSIVE SURVEY ON KOLMOGOROV ARNOLD NETWORKS (KAN)*, arXiv preprint arXiv:2407.11075, 2024.

A Appendix

A.1 B-splines and Cox-de Boor Recursion

B-splines are a class of piecewise polynomial functions commonly used for approximation and interpolation tasks. A B-spline of order k is defined over a non-decreasing sequence of knots $\{t_0, t_1, \ldots, t_{n+k}\}$, which partition the input domain.
The B-spline basis functions are computed using the Cox–de Boor recursion formula (2):

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } t_i \leq x < t_{i+1}, \\ 0 & \text{otherwise.} \end{cases}$$
$$B_{i,k}(x) = \frac{x-t_i}{t_{i+k}-t_i} B_{i,k-1}(x) + \frac{t_{i+k+1}-x}{t_{i+k+1}-t_{i+1}} B_{i+1,k-1}(x)$$

These recursive definitions allow for efficient evaluation of splines of any order.

A.2 Optimization with L-BFGS

KANs are typically trained using the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) optimizer, a quasi-Newton method suited for high-dimensional smooth optimization problems. Unlike first-order optimizers such as SGD or Adam, L-BFGS uses gradient information from multiple previous steps to approximate the inverse Hessian matrix. Given a loss function $\mathcal{L}(\theta)$ L-BFGS approximates the update direction p_k by solving:

$$p_k = H_k \nabla \mathcal{L}(\theta_k),$$

where H_k is the hessian approximation built from stored vectors $s_k = \theta_{k+1} - \theta_k$ and $y_k = \nabla \mathcal{L}_{k+1} - \nabla \mathcal{L}_k$. The optimization update is then:

$$\theta_{k+1} = \theta_k + \alpha_k p_k,$$

with α_k chosen via line search. In practice, the "limited version" only stores a small number of past updates to reduce memory usage.
L-BFGS is particularly effective in KANs due to the smoothness of B-spline activation functions and the relatively small parameter count compared to traditional deep networks. It enables fast convergence without requiring mini-batch training or heavy regularization.

A.3 Functions we implemented KANs on

1. $f_1(x) = J_0(20x)$, where J_0 is the zeroth-order Bessel function of the first kind,
2. $f_2(x, y) = \exp(\sin(\pi x) + y^2)$,
3. $f_3(x, y) = xy$, Figure 5,8
4. $f_4(x_1, x_2, \ldots, x_{100}) = exp(\frac{1}{100}(\Sigma_{i=1}^{100} sin^2(\frac{\pi x_i}{2})))$,
5. $f_5(x_1, x_2, x_3, x_4) = \exp(\frac{1}{2}(sin(\pi(x_1^2 + x_2^2)) + sin(\pi(x_3^2 + x_4^2))))$,
6. MNIST image classification dataset.
7. On a few selected examples from the Feynman dataset(Table 1)
8. $f_6(x, y) = \exp(J_0(20x) + y^2)$

9. $f_7(x, y) = \sqrt{(1 + x^2 + y^2)}$

10. $f_8(x, y) = \exp(\sin(\pi x) + y^2)$

11. $f_9(x, y) = exp(x + y^2)$

12. $f_{10}(x_1, x_2, x_3, x_4, x_5) = exp(\frac{1}{5}(\Sigma_{i=1}^{5} sin^2(\frac{\pi x_i}{2})))$(Figure 6)

13. $f_{11}(x, y) = x(\frac{1}{y} - 1)$(Figure 7)

A.4 Catastrophic Forgetting

Catastrophic forgetting is a serious problem in current machine learning systems. When a human masters a task and then switches to a new one, they typically retain the ability to perform the original task. Unfortunately, neural networks do not exhibit this property. When trained sequentially on multiple tasks, neural networks tend to overwrite previously learned knowledge.

A key distinction between artificial neural networks and biological brains is the presence of functional locality in the latter. The human brain undergoes structural changes in localized regions relevant to the new task, leaving unrelated areas untouched. In contrast, most artificial neural networks—including MLPs—lack this locality, which is a likely cause of catastrophic forgetting.

KANs, however, offer a potential solution due to their use of spline parameterizations. Spline bases are inherently local, so during training, only a few coefficients near the current data points are updated, preserving distant coefficients that may store important previous information. On the other hand, MLPs rely on global activation functions like ReLU, Tanh, or SiLU, causing even small updates to propagate across the network and overwrite unrelated regions.

To demonstrate this, we used a simple 1D regression task involving five Gaussian peaks, where each peak's data is introduced sequentially. As illustrated in the figures above, KANs remodel only the region relevant to the current phase, while MLPs tend to overwrite previously learned peaks. This preliminary result suggests that KANs may be more resilient to catastrophic forgetting.

While this toy example shows the promise of leveraging locality in KANs, it remains to be seen whether the benefits carry over to high-dimensional tasks. In future work, we aim to study how KANs can integrate with state-of-the-art continual learning strategies.

A.5 Additional Graphs and Plots

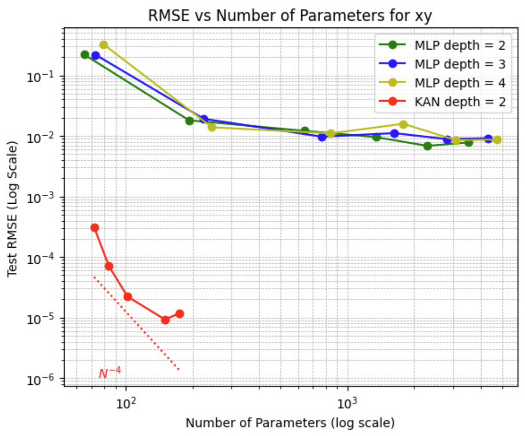


Figure 5: Verifying scaling bounds for $f_3(x, y)$

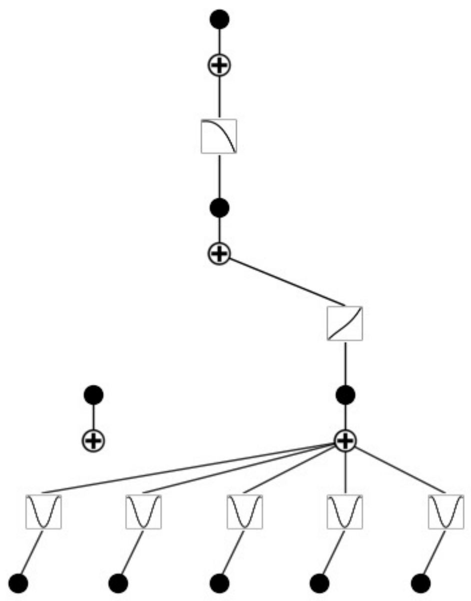


Figure 6: Visualization of $f_{10}(x_1, x_2, x_3, x_4, x_5)$ composition in KANs

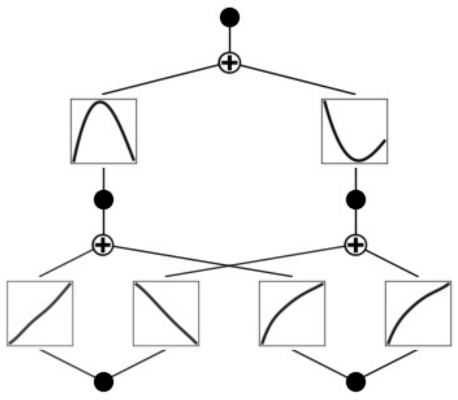


Figure 7: Visualization of $f_{11}(x, y)$ composition in KANs

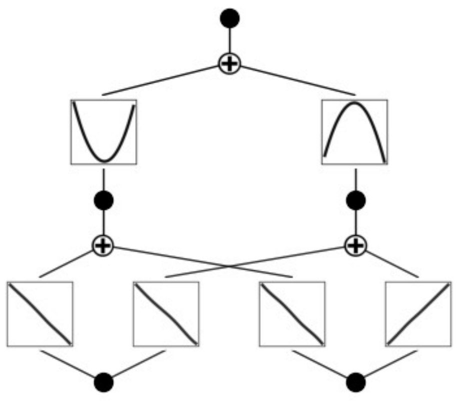


Figure 8: Visualization of $f_3(x, y)$ composition in KANs

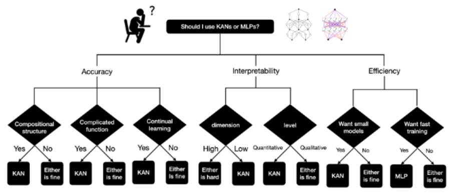


Figure 9: KANs v/s MLPs (1) (Figure 6.1)

B Contributions

B.1 JS Vishnu Teja

Read and understood the research and survey papers of KAN, implemented the neural scaling laws and experiments under Section 3.2 i.e., interpretability of KANs, Feynman datasets, catastrophic forgetting and MNIST dataset.

B.2 L Sai Harsh

Read and understood the research and survey papers of KAN, implemented the neural scaling laws and experiments under Section 3.2 i.e., interpretability of KANs, Feynman datasets, catastrophic forgetting and MNIST dataset.

B.3 K Bhargav Sashank

Read and understood the research and survey papers of KAN, implemented the neural scaling laws and experiments under Section 3.2 i.e., interpretability of KANs, Feynman datasets, catastrophic forgetting and MNIST dataset.

B.4 Kunjan Manoj Kumar S

Read and understood the research and survey papers of KAN, implemented the neural scaling laws and experiments under Section 3.2 i.e., interpretability of KANs, Feynman datasets, catastrophic forgetting and MNIST dataset.