Understanding the Limitations of B-Spline KANs: Convergence Dynamics and Computational Efficiency

Avik Pal[*]
MIT CSAIL
avikpal@mit.edu

Dipankar Das
Intel Labs
das.dipankar@intel.com

Abstract

Kolmogorov-Arnold Networks (KANs) have recently emerged as a potential alternative to multi-layer perceptrons (MLPs), leveraging the Kolmogorov Representation Theorem to introduce learnable activation functions on each edge rather than fixed activations at the nodes. While KANs have demonstrated promise in small-scale problems by achieving similar or better performance with fewer parameters, our empirical investigations reveal significant limitations when scaling to real-world tasks. Specifically, KANs suffer from increased computational costs and reduced performance, rendering them unsuitable for deep learning applications. Our study explores these limitations through extensive testing across diverse tasks, including computer vision and scientific machine learning, and provides a detailed comparison with MLPs.

1 Introduction

Kolmogorov-Arnold Networks (KANs) have recently gained attention as a novel architecture that leverages the Kolmogorov Representation Theorem [1], offering a shift from traditional multi-layer perceptrons (MLPs). By enabling learnable activation functions on edges rather than fixed node activations, KANs introduce the potential for improved parameter efficiency, as shown in small-scale studies [2, 3, 4]. However, whether KANs can scale effectively to complex, real-world tasks remains uncertain. Our comprehensive empirical analysis across computer vision and scientific machine learning tasks reveals significant limitations in computational costs and convergence dynamics, and overall practicality. We demonstrate that KANs consistently under-perform compared to MLPs in larger-scale applications, particularly through eigenvalue spectrum analysis and parameter efficiency measurements.

Although previous studies have raised concerns about KANs as a potential replacement for MLPs, the circumstances for successful deployment remain unclear. For example, [5] found KANs sub-optimal for computer vision, while [6] noted prohibitive computational costs. When FLOPs were controlled, MLPs generally outperformed KANs across tasks, excluding symbolic formula representation [7]. Additionally, [8] reported performance degradation in B-Spline KANs for differential equations and operator learning, and even in high-energy physics, KANs offered limited advantages [9]. Our work systematically evaluates KANs across multiple performance metrics, including parameter efficiency, memory consumption, and convergence behavior.

Our contributions are as follows:

1. **Parameter Efficiency:** We demonstrate that the parameter efficiency of KANs is limited to toy problems, with MLPs outperforming KANs on more complex tasks like computer vision [Section 4.2] and more structured problems like Neural ODEs [Section 4.1]. Unlike previous work, we use state-of-the-art models as our baseline.

[*]Work done during an Internship at Intel Labs

Workshop on Scientific Methods for Understanding Deep Learning, NeurIPS 2024.

2. **Scientific Machine Learning (SciML) Tasks:** We show that KANs are inadequate for common scientific machine learning tasks, such as learning dynamical systems with Neural ODEs, due to their explosive parameter growth [Section 3.1]. MLPs offer a more practical solution, fitting the data more accurately [Section 4.1].

3. **Convergence Analysis:** Our Hessian eigenvalue spectrum analysis reveals that KANs are prone to converge to sharper minima more frequently than MLPs, particularly in vision tasks [Section 4.2], potentially compromising their generalization performance.

2 Background

2.1 Kolmogorov Arnold Networks

Kolmogorov-Arnold Networks (KANs) [10] represent a specialized neural network architecture that utilizes B-splines to create adaptive activation functions. Based on the Kolmogorov Representation Theorem [1], which posits that any continuous multivariate function can be represented as a superposition of univariate functions, KANs are structured to exploit this theorem by placing activation functions on the edges instead of at the nodes, as seen in traditional MLPs. This approach, as proposed in [10], allows KANs to model complex functions with greater parameter efficiency, making them especially attractive in scenarios where memory and computational efficiency are critical.

2.2 Hessian Eigenvalue Spectral Distribution

Large-batch training methods in deep learning tend to exhibit poorer generalization compared to their small-batch counterparts [11]. This phenomenon is primarily linked to the Hessian matrix $\mathcal{H}$ characteristics. Specifically, large-batch methods are prone to converging to sharp minimizers, which are identified by a significant number of large positive eigenvalues of $\mathcal{H}$. Conversely, small-batch methods typically converge to flat minimizers, marked by a preponderance of small eigenvalues of $\mathcal{H}$.

Our analysis reveals that KANs exacerbate the issue of sharp minima, as indicated by a substantial increase in positive eigenvalues of the Hessian matrix, even when trained with small batch sizes. This sharpness strongly correlates with poor generalization performance in our experiments. Through detailed experimental evaluation, we demonstrate that KANs consistently under-perform in test-time accuracy across all computer vision tasks. This suggests that their tendency to converge to sharp minima negatively impacts their ability to generalize, especially in more complex and high-dimensional tasks.

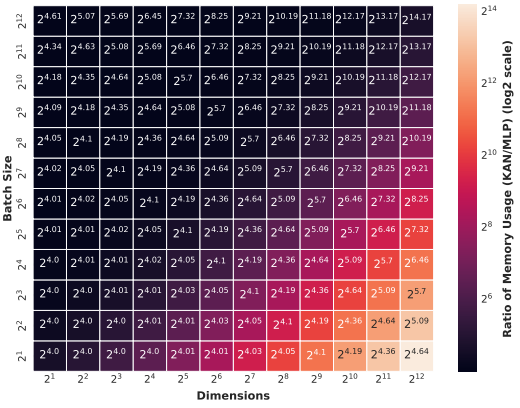


Figure 1: Heatmap representation of the additional memory requirement of KANs [Equation (1)] on varying the input/output dimensions and the batch size.

3 Analysing the Computational Requirements for KANs

KANs exhibit significantly higher computational demands compared to MLPs when learning a mapping from $\mathbb{R}^I$ to $\mathbb{R}^O$ with comparable network width. In this section, we conduct a thorough analysis of the computational requirements for KANs across two dimensions: parameter count, and memory usage[2]. In our analysis, we denote σ as an arbitrary activation function applied subsequent to the fully-connected (FC) layer. For the KAN layer, we employ the `SiLU` as the base activation function, with the grid size and spline order represented by G and S, respectively.

[2]Our analysis is based on the efficient-KAN implementaion [12].

3.1 Parameter Count

The number of parameters for a fully connected layer is $O \times (I + 1)$. For a KAN layer the total number of learnable parameters, which includes the control points, shortcut weights, and B-spline weights, is $I \times O \times (2 + G + S)$. Hence, a standard KAN formulation with $G = 5$ and $S = 3$ has approximately 10 times more trainable parameters than an equivalent FC layer.

3.2 Memory Usage

Forward Pass: An FC layer requires the allocation of a single $O \times B$ matrix to store the output. In contrast, evaluating a KAN layer necessitates the allocation of $I \times B + O \times B + I \times (G + S) \times B$ in total. Consequently, KANs demand $I \times (1 + G + S) \times B$ additional elements compared to an FC layer.

Reverse Pass: During training, additional memory must be allocated to store intermediates for gradient computation. An FC layer requires at most an additional $O \times B$ matrix for caching these intermediates. Utilizing De Boor's Algorithm [13] for computing B-splines, we can recursively calculate the gradients for splines of order S using the spline evaluations for order $S - 1$. This necessitates an extra tensor of size $I \times (G+S-1) \times B$ for gradient computation in KANs. Therefore, the total extra memory allocation required by a single KAN layer, in comparison to an FC layer, is as follows:

$$\underbrace{I \times (1 + G + S) \times B}_{\text{Forward Pass}} + \underbrace{I \times (G + S - 1) \times B - O \times B}_{\text{Reverse Pass}} + \underbrace{I \times O \times (2 + G + S) - I \times (O + 1)}_{\text{Parameter Count}} \quad (1)$$

$$= I\,[2(G + S)B + (G + S + 1)O - 1] - OB \quad (2)$$

4 Experiments

We evaluate the performance of KANs versus MLPs across three key domains, selecting problem setups aligned with the original KAN paper's claims on parameter efficiency [10]:

- Learning dynamical systems with neural ODEs [14, 15]: SciML problems are characterized by low-parameter configurations with strong inductive biases, making this an ideal test case for KANs, which are designed to perform well in low-parameter regimes.
- Computer vision with mixer models and transformers: MLPs are heavily used in these models and often present a performance bottleneck, replacing them with KANs could provide parameter efficiency gains with potential direct performance improvements.
- Operator Learning with Fourier Neural Operators[16].

Our computer vision and operator learning task experiments are conducted using PyTorch [17], while we use Lux.jl [18, 19] for learning dynamical systems.

4.1 Neural Ordinary Differential Equations

Neural ODEs [15] utilize neural networks to model the rate of change in a dynamical system, which is then numerically solved using ODE solvers. The model parameters are optimized through adjoint methods or discrete sensitivities [20, 21].

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \underbrace{f(u, \theta, t)}_{\text{neural network}} \quad (3)$$

This approach excels at learning from temporal or sequential data by embedding structural information inherent in the dynamics of the system [14]. In our experiments, we evaluate the effectiveness of KAN-based and MLP-based neural ODEs across various tasks, including the

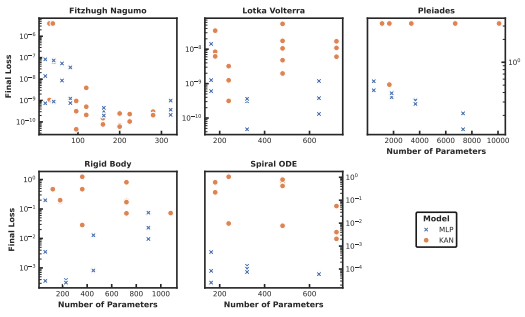
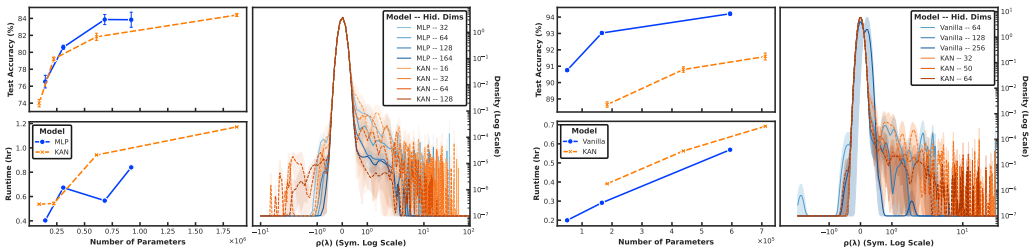


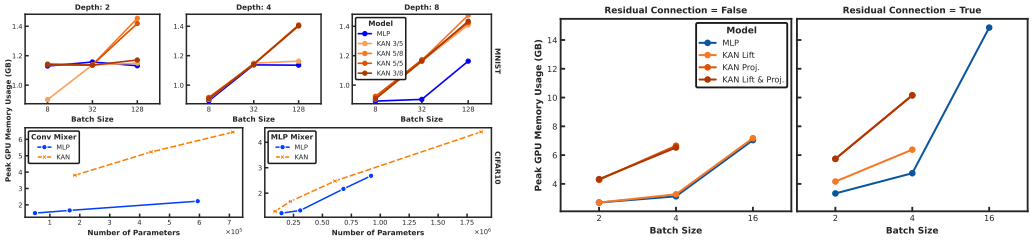
Figure 2: **Comparison of Learning Dynamical Systems using Neural ODEs with MLPs and KANs**: MLPs converge to a lower final loss compared to KANs in all problems (except Fitzhugh Nagumo where the difference in final loss is in the order of 10^{-10}).



(a) MLP-Mixer with KAN and MLP Base

(b) Conv-Mixer with KAN and Conv Base

Figure 3: KAN variants converge to sharp minima (large number of positive eigenvalues) which MLP variants converge to smooth minima. Additionally KANs use significantly large number of parameters compared to MLP while attaining a lower final accuracy.



(a) Image Classification Tasks

(b) Operator Learning Tasks

Figure 4: **KAN-based models have significantly higher memory usage with poor scaling**: While both KAN-based and MLP-based models demonstrate a linear growth in memory usage, the memory usage for the former grows at a much faster rate. *Missing points in the plot demonstrate that the models ran out of memory on a 16GB V100 GPU.*

Lotka-Volterra [22], Pleiades Problem [23], FitzHugh-Nagumo [24], Rigid Body Problem, and Spiral ODE. Our results [Figure 2] demonstrate that, while MLP-based models consistently converge to lower final losses, KANs struggle to fit dynamical systems like the Pleiades and Spiral ODE problems.

4.2 Computer Vision

We evaluate the effect of introducing KANs in MLP-Mixer [25], Conv-Mixer [26], and DeiT [27] for CIFAR-10 and CIFAR-100[3]. In MLP-Mixer, we replace the MLPs with Spline KANs, similarly for Conv-Mixer we augment the convolution layers with a KAN layer[4]. For DeiT, we use the formulation in [29] of replacing the MLPs with B-Spline and RSWAF KANs [30]. Figure 3 shows that KAN variants converge to sharp minima (as evidenced by a large number of positive eigenvalues). Additionally, the original claim of KANs requiring lower parameters [10], doesn't hold up in our experiments, and KANs require significantly larger number of parameters compared to an equivalent MLP, and even then fail to match the accuracy of MLPs. While this issue of sharper minima doesn't seem to occur for larger models like DeiT, comparing the accuracies of KANs and MLPs we observe that MLPs outperform B-Spline KANs by $10-12\%$ and RSWAF KANs by $8-10\%$ while using fewer parameters.

Base	Variant	Number of Params	Top 1 Acc (%)	Top 5 Acc (%)
`MLP`	`Tiny`	5.5M	83.93	**96.85**
$\texttt{KAN}_{\texttt{Spline}}$	`Tiny`	14.3M	73.03	93.23
$\texttt{KAN}_{\texttt{RSWAF}}$	`Tiny`	12.6M	72.92	93.13
`MLP`	`Small`	21.7M	86.49	96.16
$\texttt{KAN}_{\texttt{Spline}}$	`Small`	39.4M	73.88	93.49
$\texttt{KAN}_{\texttt{RSWAF}}$	`Small`	35.8M	78.69	94.75
`MLP`	`Base`	85.8M	**86.89**	96.08

Table 1: CIFAR-100 classification using DeiT models: MLPs outperform KANs by a margin of $8-12\%$ while requiring significantly less parameters.

[3]For the sake of brevity, we don't report the MNIST results in this manuscript, however, we have similar observations to CIFAR-10 when using MLP-Mixer.

[4]We don't use the formulation in [28] to avoid an explosive parameter growth due to grouped convolutions.

4.3 Fourier Neural Operators

Fourier Neural Operators (FNOs) leverage the Fourier transform to learn mappings between infinite-dimensional function spaces to solve partial differential equations by efficiently encoding spatial-temporal correlations in the frequency domain [16]. We use the FNO++ formulation proposed in [31].

We evaluate KANs as an alternative to the lift and projection MLP networks in FNOs. Figure 5 show KANs performing slightly worse compared to MLPs without residual connections. However, in the presence of residual connections, KANs do outperform MLPs. However, this comes at the cost of extremely high memory requirements [Figure 4b] making training these models infeasible.

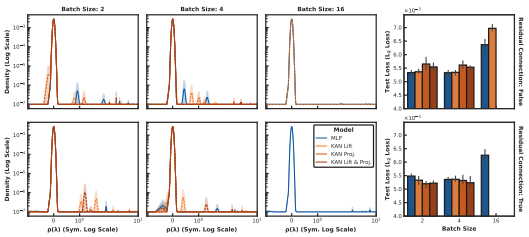


Figure 5: MLPs outperform KANs in learning solutions of darcy flow using weight-tied fourier neural operators without residual connections. The presence of residual connections provides a minor improvement to KAN performance.

5 Conclusion

Our comprehensive analysis highlights significant performance limitations of KANs when scaling to complex, real-world tasks. Despite their theoretical appeal and parameter efficiency in toy problems, KANs fail to outperform MLPs in practical applications, including computer vision and SciML tasks. Our eigenvalue spectral analysis shows that KANs are more likely to converge to sharp minima, negatively impacting their generalization performance. Moreover, their higher computational cost and parameter inefficiency render them impractical for large-scale deep learning applications Thus, at least for the time being, MLPs remain a more effective and efficient choice for a broader range of applications.

5.1 Shortcomings

We note that our studies have the following limitations. We additionally highlight the reason why we don't cover them in our experiments:

1. Progressive Refinement of the Grid: [10] progressively refine the grid to improve the fitting of the model. However, as noted previously the parameter growth w.r.t. a finer grid prohibitively increases the parameter count and memory usage making the comparison to a smaller MLP unfair.
2. Higher Order Optimization: We don't consider optimizers like L-BFGS in our experiments (except for Neural ODEs), since these tend to not scale for larger problems that were the primary focus of our study.

References

[1] Vladimir M. Tikhomirov. On the representation of continuous functions of several variables as superpositions of continuous functions of one variable and addition. 1991.

[2] Diab W Abueidda, Panos Pantidis, and Mostafa E Mobasher. Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems. *arXiv preprint arXiv:2405.19143*, 2024.

[3] Minjong Cheon. Demonstrating the efficacy of kolmogorov-arnold networks in vision tasks. *arXiv preprint arXiv:2406.14916*, 2024.

[4] Amanda A Howard, Bruno Jacob, Sarah H Murphy, Alexander Heinlein, and Panos Stinis. Finite basis kolmogorov-arnold networks: domain decomposition for data-driven and physics-informed problems. *arXiv preprint arXiv:2406.19662*, 2024.

[5] Basim Azam and Naveed Akhtar. Suitability of kans for computer vision: A preliminary investigation. *arXiv preprint arXiv:2406.09087*, 2024.

[6] Haoran Shen, Chen Zeng, Jiahui Wang, and Qiao Wang. Reduced effectiveness of kolmogorov-arnold networks on functions with noise. *arXiv preprint arXiv:2407.14882*, 2024.

[7] Runpeng Yu, Weihao Yu, and Xinchao Wang. Kan or mlp: A fairer comparison. *arXiv preprint arXiv:2407.16674*, 2024.

[8] Khemraj Shukla, Juan Diego Toscano, Zhicheng Wang, Zongren Zou, and George Em Karniadakis. A comprehensive and fair comparison between mlp and kan representations for differential equations and operator networks. *arXiv preprint arXiv:2406.02917*, 2024.

[9] Johannes Erdmann, Florian Mausolf, and Jan Lukas Späh. Kan we improve on hep classification tasks? kolmogorov-arnold networks applied to an lhc physics example. *arXiv preprint arXiv:2408.02743*, 2024.

[10] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks, 2024.

[11] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[12] Blealtan. An efficient implementation of kolmogorov-arnold network. `https://github.com/Blealtan/efficient-kan`, 2024.

[13] C de Boor. Subroutine package for calculating with b-splines, 1971.

[14] Christopher Rackauckas, Yingbo Ma, Julius Martensen, Collin Warner, Kirill Zubov, Rohit Supekar, Dominic Skinner, Ali Ramadhan, and Alan Edelman. Universal differential equations for scientific machine learning. *arXiv preprint arXiv:2001.04385*, 2020.

[15] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[16] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations, 2021.

[17] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[18] Avik Pal. Lux: Explicit Parameterization of Deep Neural Networks in Julia, April 2023. If you use this software, please cite it as below.

[19] Avik Pal. On Efficient Training & Inference of Neural Differential Equations, 2023.

[20] Patrick Kidger. On neural differential equations. *arXiv preprint arXiv:2202.02435*, 2022.

[21] Facundo Sapienza, Jordi Bolibar, Frank Schäfer, Brian Groenke, Avik Pal, Victor Boussange, Patrick Heimbach, Giles Hooker, Fernando Pérez, Per-Olof Persson, et al. Differentiable programming for differential equations: A review. *arXiv preprint arXiv:2406.09699*, 2024.

[22] Peter J Wangersky. Lotka-volterra population models. *Annual Review of Ecology and Systematics*, 9:189–218, 1978.

[23] George H Herbig. Spectral classification of faint members of the hyades and pleiades and the dating problem in galactic clusters. *Astrophysical Journal, vol. 135, p. 736*, 135:736, 1962.

[24] Eugene M Izhikevich and Richard FitzHugh. Fitzhugh-nagumo model. *Scholarpedia*, 1(9):1349, 2006.

[25] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[26] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022.

[27] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. arxiv 2020. *arXiv preprint arXiv:2012.12877*, 2(3), 2020.

[28] Ivan Drokin. Kolmogorov-arnold convolutions: Design principles and empirical studies. *arXiv preprint arXiv:2407.01092*, 2024.

[29] 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.

[30] Athanasios Delis. Fasterkan. `https://github.com/AthanasiosDelis/faster-kan/`, 2024.

[31] Tanya Marwah, Ashwini Pokle, J. Zico Kolter, Zachary C. Lipton, Jianfeng Lu, and Andrej Risteski. Deep equilibrium based neural operators for steady-state pdes, 2023.