A Reproduction and Critique of Kolmogorov-Arnold Networks

Pau Mateo
paumb@kth.se

Pol Resina
polrm@kth.se

André Lindgren
andrel4@kth.se

Abstract

This project replicates Kolmogorov-Arnold Networks (KANs), confirming their superior parameter efficiency and adherence to N^{-4} scaling laws in low-complexity tasks. Critically, our reproduction found KANs highly susceptible to overfitting when increasing capacity, displaying U-shaped error curves. In special function approximation, MLPs were significantly more robust, achieving lower test RMSE in 10 out of 15 benchmarks. Finally, KANs successfully demonstrated locality by avoiding catastrophic forgetting in a 1D problem, but this property failed to generalize to a 2D classification task.

1 Introduction

The Multi-Layer Perceptron (MLP) has long served as the foundational architecture of deep learning. Since the popularization of the backpropagation algorithm by Rumelhart et al. (2), the standard design of fixed activation functions on neurons combined with learnable linear weights has underpinned advancements from computer vision to large language models. While MLPs are theoretically capable of approximating any continuous function, a property guaranteed by the Universal Approximation Theorem, they often struggle with interpretability and can be inefficient in parameter usage for high-dimensional scientific tasks.

Recently, Liu et al. (1) proposed a fundamental shift in this design by introducing Kolmogorov-Arnold Networks (KANs). This architecture draws inspiration from the Kolmogorov-Arnold representation theorem, originally established by Kolmogorov (3), which posits that multivariate continuous functions can be represented as a finite composition of continuous functions of a single variable and addition.

While MLPs place fixed non-linearities on nodes, KANs place learnable activation functions on edges, eliminating linear weight matrices entirely. Instead, every parameter is replaced by a univariate B-spline. The authors demonstrate that this architectural change allows KANs to outperform MLPs in terms of both accuracy and interpretability on small-scale scientific tasks. Furthermore, KANs reportedly exhibit faster neural scaling laws (RMSE $\propto N^{-4}$) compared to the slower convergence of MLPs.

In this work, we attempt to reproduce the core results presented by Liu et al. (1), and additionally report results from new continual learning experiments on non-symbolic datasets. Beyond performance evaluation, we emphasize reproducibility by examining how reliably the reported results can be replicated. We discuss challenges encountered during reimplementation, including ambiguities and missing methodological details, and assess their impact on reproducibility. Code for all experiments is available at `https://github.com/paumateo/Deep-Learning-Advanced-KAN-Project`.

Project report for DD2412 or FDD3412

2 Related Work

The original KAN paper (Kolmogorov–Arnold Networks) introduces a neural architecture motivated by the Kolmogorov–Arnold representation theorem, replacing linear weights with learnable univariate spline functions to improve parameter efficiency and interpretability compared to standard multilayer perceptrons (**?**). The work builds upon earlier studies on Kolmogorov–Arnold representations in neural networks (e.g., Kolmogorov–Arnold Representations and Neural Networks) (3), which mainly focused on theoretical constructions or shallow models, and extends them with modern optimization and empirical evaluation.

Following the introduction of KAN, several works have explored its applicability in other domains. For tabular learning, TabKAN (7) and TabKANet: Enhancing Tabular Data Modeling with KAN-based Numerical Embeddings (6) use KAN modules as numerical feature encoders, with TabKANet integrating KAN-based embeddings into Transformer architectures. In graph learning, GraphKAN (8) applies KAN components within graph neural networks for feature transformation and aggregation. Together with other recent extensions, such as KAGNNs (9), these works suggest that KANs can serve as flexible architectural building blocks, particularly as embedding or feature transformation modules in larger models across diverse data modalities.

3 Methods

Kolmogorov–Arnold Networks

Kolmogorov–Arnold Networks (KANs) are neural architectures inspired by the Kolmogorov–Arnold representation theorem, which states that any multivariate continuous function can be expressed as a finite sum of univariate functions composed with additions. Unlike standard multilayer perceptrons (MLPs), which rely on linear weights followed by nonlinear activations, KANs replace each linear weight with a learnable univariate function defined on each edge of the network.

Grid extension

Grid extension in KANs increases the resolution of the B-spline grids, allowing finer approximations of target functions by fitting a new fine-grained spline to a previously learned coarse representation. By making the grid arbitrarily fine, the model's expressive capacity improves without retraining larger networks from scratch. (1)

Sparsification and pruning

To encourage interpretability and reduce model complexity, KANs employ sparsification techniques during training. An L1 and entropy regularization term is applied to the spline parameters, promoting sparsity in the learned univariate functions. As a result, many edge functions become nearly constant or inactive, effectively identifying irrelevant connections.

After training, pruning removes edges whose learned functions contribute negligibly to the output, typically based on small weights or low functional variance. This reduces model complexity while preserving predictive performance, leading to sparse, interpretable, and parameter-efficient networks.

4 Data

We utilize synthetic datasets to precisely benchmark the neural scaling laws of the models. Synthetic data provides a controlled ground truth, allowing us to evaluate function approximation without the noise inherent in real-world data.

4.1 Data for Section 3.1: Toy datasets

To reproduce the results from Section 3.1 of Liu et al. (1), we generated five distinct "toy" datasets. These functions were chosen to represent various challenges in scientific machine learning, including high dimensionality, oscillatory behavior, and variable interaction. Following the experimental setup in the original paper, we sampled N=1000 training points and $N = 1000$ test points uniformly from the domain $[-1, 1]^d$.

- **Case 1 (1D Oscillatory):** $f(x) = J_0(20x)$, where J_0 is the Bessel function of the first kind. This function tests the model's ability to capture high-frequency oscillations.
- **Case 2 (2D Asymmetric):** $f(x, y) = \exp(\sin(\pi x) + y^2)$. This serves as a benchmark for non-linear interactions between variables.
- **Case 3 (2D Product):** $f(x, y) = xy$. This dataset tests the ability to model multiplication, a task where KANs are theoretically expected to outperform MLPs due to their ability to learn logarithmic and exponential transformations.
- **Case 4 (100D High-Dimensional):** $f(\mathbf{x}) = \exp\left(\frac{1}{100}\sum_{i=1}^{100}\sin^2\left(\frac{\pi x_i}{2}\right)\right)$. This high-dimensional function is used to investigate the model's resistance to the "curse of dimensionality."
- **Case 5 (4D Interaction):** $f(\mathbf{x}) = \exp\left(0.5\left(\sin(\pi(x_1^2 + x_2^2)) + \sin(\pi(x_3^2 + x_4^2))\right)\right)$. This simulates a physical interaction with a clear compositional structure.

4.2 Data for Section 3.2: Special functions

To reproduce the findings regarding special function approximation, we generated synthetic datasets for 15 distinct special functions common in mathematics and physics, as specified in Table 1 of Liu et al. (1). These functions, including Jacobian elliptic functions, Bessel functions ($J_\nu, Y_\nu, I_\nu, K_\nu$), Associated Legendre polynomials, and spherical harmonics, were chosen to test the model's ability to capture high-frequency oscillations and singularities (1). Ground truth values were generated using the `scipy.special` API (10). Consistent with the original experimental setup, we uniformly sampled $N = 1000$ training points and $N = 1000$ test points from the respective domains of each function.

4.3 Data for Section 3.3: Feynman datasets

To reproduce the results of Section 3.3 of Liu et al. (1) we used a subset of the feynman datasets used in the original article (see Table 2). These Feynman datasets are sourced from Feynman's textbooks (4) (5). Trying to use an experimental setup as close as the one used in the original KAN paper, we used as well $N = 1000$ training points and $N = 1000$ test points, uniformly sampled from the functions, although in the original paper (1) they did not specify the number of train and test points used with the feynman datasets. Later in Section 6 we show how this could lead to different results. We used the sampling ranges we found in the implementation of Feynman datasets of the pykan package or in the github repository https://github.com/KindXiaoming/pykan (1).

5 Experiments and findings

In this work, we focus on reproducing the fundamental experimental results presented by Liu et al. (1). To do so, we implement a comparative framework between the proposed Kolmogorov-Arnold Networks (KANs) and the traditional Multi-Layer Perceptron (MLP) architecture. The goal is to evaluate their respective parameter efficiencies under identical optimization conditions.

5.1 Reproduction of Section 3.1: Toy Function Approximation

Following Section 2 of the original paper (1), we utilize KANs with learnable B-spline activations on edges. We adopt the *grid extension* strategy to progressively refine the network, increasing the grid size G according to the schedule $G \in \{5, 20, 50, 100, 200, 500\}$ with cubic splines ($k = 3$). We employ the *LBFGS* optimizer (200 steps per grid level) to ensure the high-precision convergence required for scientific tasks.

For the baseline, we implement standard MLPs with `ReLU` activations. To ensure a fair comparison based on parameter efficiency, we perform a hyperparameter sweep over depths $D \in \{2, \ldots, 5\}$and widths $W \in \{10, \ldots, 1000\}$, excluding configurations that exceed 200,000 parameters. Consistent with the KAN experiments, all MLP baselines are trained using LBFGS to minimize Mean Squared Error (MSE) (1).

The findings validate the core claims of the original paper regarding KAN efficiency.

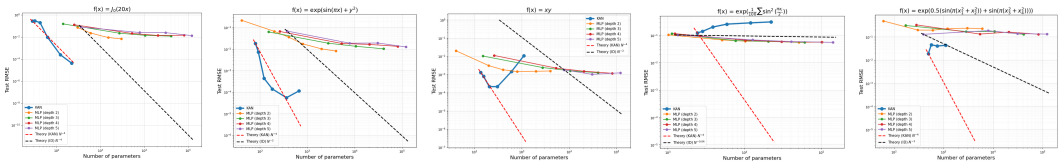

Figure 1: Neural scaling laws for the five toy datasets. The plots show Test RMSE vs. Number of Parameters. The blue line represents KAN, while other colors represent MLPs of varying depths. The dashed red line indicates the theoretical N^{-4} scaling.

The experimental results confirm the efficiency of KANs, particularly in low-parameter regimes where they consistently outperform MLPs. As shown in Figure 1, KANs exhibit a steeper scaling law, with the $f(x) = J_0(20x)$ case specifically showing a remarkable alignment with the theoretical N^{-4} bound. This validates that KANs can achieve superior accuracy with significantly fewer parameters.

However, a notable deviation from the original paper's findings is observed as model capacity increases. While the reference results report a monotonic reduction in Test RMSE, the blue trendlines in this reproduction frequently exhibit a distinct "U-shaped" trajectory or sharp uptick in error after reaching a minimum (most visible in the $exp(sin(\pi x) + y^2)$ and 4D function cases). This behavior indicates that the KAN models in this setup are suffering from overfitting in the high-parameter regime. In contrast, the original authors' results show continued improvement, suggesting their implementation likely employed more robust regularization or optimization strategies to preserve generalization as complexity grew.

5.2 Reproduction of Section 3.2: Special Functions

In this section, we reproduce the experiments on special function approximation presented in Section 3.2 of Liu et al. (1). The benchmark consists of a diverse collection of special functions implemented via the `scipy.special` API (10), including elliptic integrals, Bessel functions, associated Legendre functions, and spherical harmonics.

For each function, we evaluate:

- A **minimal KAN configuration**, defined as the smallest network (in terms of width and depth) capable of achieving a test RMSE below 10^{-2}.
- The **best-performing KAN configuration** found during the architecture search.
- A **best MLP baseline** trained under the same optimization protocol.

Table 1 summarizes performance. Detailed scaling plots are provided in Appendix C; despite visual discrepancies with the original paper, we base our analysis on the table's quantitative metrics.

Name	scipy.special API	Minimal KAN shape	Minimal KAN test RMSE	Best KAN shape	Best KAN test RMSE	MLP test RMSE
Jacobian elliptic functions	ellipj(x, y)	[2,2,1]	1.20×10^{-2}	[2,3,2,1,1,1]	5.33×10^{-2}	$\mathbf{3.86 \times 10^{-3}}$
Incomplete elliptic integral (1st kind)	ellipkinc(x, y)	[2,1,1]	1.88×10^{-2}	[2,2,1,1,1]	7.25×10^{-2}	$\mathbf{7.12 \times 10^{-3}}$
Incomplete elliptic integral (2nd kind)	ellipeinc(x, y)	[2,1,1]	1.66×10^{-2}	[2,2,1,1]	1.96×10^{-2}	$\mathbf{1.34 \times 10^{-3}}$
Bessel function (1st kind)	jv(x, y)	[2,2,1]	2.12×10^{-2}	[2,3,1,1]	9.07×10^{-2}	$\mathbf{3.83 \times 10^{-3}}$
Bessel function (2nd kind)	yv(x, y)	[2,3,1]	5.32×10^{4}	[2,2,2,1]	8.72×10^{4}	$\mathbf{5.24 \times 10^{4}}$
Modified Bessel (2nd kind)	kv(x, y)	[2,1,1]	7.22×10^{4}	[2,2,1]	8.27×10^{4}	$\mathbf{1.54 \times 10^{4}}$
Modified Bessel (1st kind)	iv(x, y)	[2,4,3,2,1,1]	2.83×10^{-1}	[2,4,3,2,1,1]	1.93×10^{-1}	$\mathbf{1.38 \times 10^{-2}}$
Assoc. Legendre $(m = 0)$	lpmv$(0, x, y)$	[2,2,1]	$\mathbf{7.69 \times 10^{-2}}$	[2,2,1]	9.33×10^{-2}	8.93×10^{-2}
Assoc. Legendre $(m = 1)$	lpmv$(1, x, y)$	[2,4,1]	$\mathbf{3.72 \times 10^{-1}}$	[2,4,1]	4.48×10^{-1}	5.42×10^{-1}
Assoc. Legendre $(m = 2)$	lpmv$(2, x, y)$	[2,2,1]	$\mathbf{6.24}$	[2,3,2,1]	3.36×10^{1}	3.15×10^{1}
Spherical harmonics $(m = 0, n = 1)$	sph_harm$(0, 1, x, y)$	[2,1,1]	1.44×10^{-2}	[2,1,1]	1.47×10^{-2}	$\mathbf{1.36 \times 10^{-3}}$
Spherical harmonics $(m = 1, n = 1)$	sph_harm$(1, 1, x, y)$	[2,2,1]	9.96×10^{-3}	[2,3,2,1]	8.51×10^{-3}	$\mathbf{4.81 \times 10^{-3}}$
Spherical harmonics $(m = 0, n = 2)$	sph_harm$(0, 2, x, y)$	[2,1,1]	3.14×10^{-3}	[2,1,1]	$\mathbf{2.40 \times 10^{-3}}$	9.69×10^{-3}
Spherical harmonics $(m = 1, n = 2)$	sph_harm$(1, 2, x, y)$	[2,2,1]	$\mathbf{1.09 \times 10^{-2}}$	[2,2,1,1]	7.20×10^{-2}	1.32×10^{-2}
Spherical harmonics $(m = 2, n = 2)$	sph_harm$(2, 2, x, y)$	[2,2,1]	$\mathbf{8.26 \times 10^{-3}}$	[2,2,3,2,1]	1.70×10^{-1}	2.08×10^{-2}

Table 1: Performance comparison on special function approximation tasks.

The data indicates that the MLP baseline is currently the more robust approximator in our setup, achieving the lowest test RMSE in 10 out of 15 functions. While KANs perform competitively on Associated Legendre polynomials and some spherical harmonics, both architectures struggle significantly with numerical stability on functions containing singularities (e.g., Y, K), resulting in extremely high error magnitudes.

These findings contrast sharply with the original paper, where KANs were reported as strictly superior to MLPs across all tasks, often with error rates orders of magnitude lower (10^{-5} vs. our 10^{-2}). Furthermore, the original work successfully approximated the unbounded functions that caused divergence in our experiments, suggesting their implementation likely utilized specific domain handling or optimization strategies that were necessary to achieve such high precision.

5.3 Reproduction of Section 3.3: Feynman datasets

In this section we aim to reproduce the experiments and results presented on section 3.3 Feynamn datasets of Liu et al. (1). The goal of this experiments is to evaluate KANs and compare them with MLPs with datasets where the human-constructed KANs might not be optimal. In this regime, it's specially interesting to compare human-constructed KANs and auto-discovered KANs via pruning (techniques described in Section 2.5.1 of Liu et al. (1)).

For each of the Feynman datasets we used, we compared four kinds of neural networks:

(1) Human-constructed KANs. Oracle KANs manually derived from Kolmogorov–Arnold representations of the ground-truth symbolic expressions.

(2) KAN without pruning. Fixing the KAN shape to width 5 and depths swept over {2,3,4}.

(3) KAN with pruning. Using the same shapes as in (2), but using sparsification and pruning after fitting the network to obtain a smaller KAN. The sparsification rate is swept with $\lambda = 10^{-2}$ and $\lambda = 10^{-3}$.

(4) MLPs with "same" shapes used in (2). That is, width fixed to 5 and depths swept over {2,3,4,5,6}. Activations are swept from {Tanh, ReLU, SiLU}.

All the KANs described are trained with LBFGS, and initialized with grid $G = 3$, increasing every 200 steps to cover $G = \{3, 5, 10, 20, 50, 100, 200\}$. Each parameter combination is run with three different seeds, and for a given model and dataset we consider only the best result over all tries. We used the seeds {42, 171, 3} for both training and data generation, except for the experiments with the pruned KANs, due to its high computing time. With most models we simply considered the test RMSE, but with pruned KANs, the smallest shape that achieves $< 10^{-1}$ RMSE and the shape of the model with best performance (lowest test RMSE) are reported as well.

Feynman Equation	Human-constructed KAN shape	Pruned KAN shape (smallest shape that achieves RMSE $< 10^{-1}$)	Pruned KAN shape (lowest loss)	Human-constructed KAN loss (lowest test RMSE)	Pruned KAN loss (lowest test RMSE)	Unpruned KAN loss (lowest test RMSE)	MLP loss (lowest test RMSE)
I.6.20	[2,2,1,1]	[2,2,1]	[2,5,5,1]	3.94×10^{-4}	9.07×10^{-3}	$\mathbf{1.47 \times 10^{-5}}$	3.75×10^{-4}
I.6.20b	[3,2,2,1,1]	[3, 2, 1]	[3, 2, 1]	5.42×10^{-2}	2.27×10^{-2}	$\mathbf{3.26 \times 10^{-4}}$	5.72×10^{-4}
I.9.18	[6,4,2,1,1]	—	—	1.12×10^{-2}	—	$\mathbf{8.13 \times 10^{-4}}$	1.21×10^{-3}
I.15.3x	[2,2,1,1]	[2,2,1]	[2,3,1]	1.46×10^{-1}	7.33×10^{-2}	5.68×10^{-3}	$\mathbf{4.19 \times 10^{-3}}$
I.16.6	[2,2,2,2,2,1]	[2,2,1]	[2,2,1]	6.83×10^{-2}	2.69×10^{-2}	$\mathbf{2.22 \times 10^{-3}}$	3.47×10^{-3}
I.18.4	[2,2,2,1,1]	[2,2,1]	[2, 3, 1]	1.27×10^{-1}	6.01×10^{-2}	$\mathbf{8.87 \times 10^{-4}}$	1.64×10^{-3}
I.27.6	[2,2,1,1]	—	—	1.74×10^{-4}	—	$\mathbf{2.05 \times 10^{-5}}$	5.49×10^{-4}
III.17.37	[3,3,3,2,2,1]	[3,2,1]	[3,2,1,1]	5.75×10^{-3}	2.01×10^{-2}	$\mathbf{5.96 \times 10^{-4}}$	3.75×10^{-3}
I.26.2	[2,2,2,1,1]	[2,2,1]	[2,2,1]	3.11×10^{-3}	8.57×10^{-3}	$\mathbf{5.62 \times 10^{-4}}$	2.30×10^{-3}

Table 2: Feynman dataset results. Some datasets have missing results for the Pruned KANs due to errors in the pykan library when fitting the models with those specific datasets.

Table 2 summarizes the results obtained in our experiments on functions from the Feynman dataset. While the original KAN paper (1) reports comparable average performance between MLPs and KANs on this benchmark, our results show a more nuanced behavior. In several of the evaluated functions, unpruned KANs achieved significantly lower RMSE than MLPs, indicating a clearer dominance of KANs in our experimental setting.

The original paper argues that the Feynman dataset is relatively simple and therefore does not fully highlight the advantages of KANs over MLPs, as KANs are expected to excel in functions with strong periodic or oscillatory structure (1). While this reasoning remains plausible, our results suggest that KANs can still outperform MLPs even on such relatively simple symbolic regression tasks, depending on training and model configuration.

Consistent with the findings reported in (1), we observed that KAN architectures discovered automatically through sparsification and pruning are often smaller than manually designed, human-constructed KAN shapes. These compact architectures were still able to achieve good performance, typically

reaching RMSE values below 10^{-1}, highlighting KANs' potential for learning minimal yet effective representations.

However, when comparing numerical results in more detail, we observe two notable discrepancies with respect to the original paper (1). First, unpruned KANs in our experiments consistently outperform MLPs by a larger margin than reported in the original work. Second, pruned KANs achieved substantially worse performance across all tested datasets, despite yielding network shapes (Table 2, columns 2 and 3) that were visually similar to those reported in (1). After extensive experimentation with different training and pruning configurations, we hypothesize that these discrepancies stem from missing or underspecified implementation details in the original paper, which we discuss further in Section 6.

5.4 Reproduction of Section 3.5: Continual Learning

This section verifies KAN ability to mitigate catastrophic forgetting using the provided toy example from the paper. The goal was to show that a KAN network only updates its model around the region it is currently being trained on, in sharp contrast to an MLP, which exhibits significant catastrophic forgetting in this scenario. Our reproduction, utilizing the authors' provided code, successfully validated the original paper's findings. As shown in Figure 10, the KAN model only adjusted parameters in the trained local region, whereas the MLP failed to retain data from prior training stages.

5.5 Extension of Section 3.5: Continual Learning

We further investigated the capacity of KANs to mitigate catastrophic forgetting beyond the simple, 1-dimensional example presented by the original authors. The goal was to better understand the conditions under which KANs exhibit this property.

A simple 2-class, 2-dimensional classification task was constructed using the `make_moons` dataset from `sklearn.datasets`. Two baseline classifiers were trained: a [2, 3, 2] MLP and a [2, 2] KAN with grid $G = 50$.

The large grid size was chosen based on prior observations during reproduction, which indicated that a high G was crucial for establishing the localized spline-fitting behavior necessary for region locality. This choice, however, made the KAN overfit, making it unsuitable for drawing conclusions about its parameter efficiency in this context and suggesting a possible heavy trade-off between parameter efficiency and continual learning capability.

Two sequential training experiments were conducted:

In the first experiment, the classifiers were trained sequentially, one class at a time. This experiment showed that neither the MLP nor the KAN was able to learn a meaningful decision boundary, as can be seen in 12.

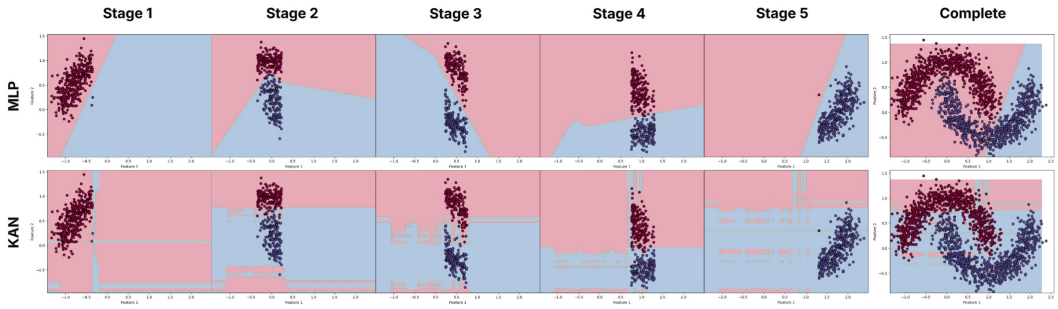

Figure 2: Sequential training across 5 stages for MLP and KAN. Each stage shows the training data for that stage and the decision boundary modeled up to that point in the training. The rightmost plots show the complete training set including the final models decision boundaries.

In the second experiment, the models were trained sequentially in five distinct stages across a feature vector of the training data. The results, visualized in figure 2, demonstrate that both the MLP and

the KAN exhibited severe catastrophic forgetting in this scenario as well. The decision boundary consistently shifted to accommodate the incoming data from later stages but failed to preserve the boundary learned from the earlier stages.

The observed failure of KANs to retain past knowledge was initially unexpected, as this 2D task was intended as a minimal complexity step beyond the authors' 1D demonstration toward higher-dimensional datasets. One possible explanation of this limitation is that in multi-dimensional feature spaces, sequential training along one input dimension still leads to overwritten spline functions that are directly linked to one of the other feature dimensions. So the problem is that the data can not be read sequentially because there is no way of reading it sequentially across more than 1 dimension. Therefor, it could be that the continual learning property of KANs are not useful for real-world problems, but this would need to be further investigated.

6 Challenges

We primarily utilized the official `pykan` Python library and GitHub repository provided by the authors for our implementation. This resource greatly streamlined the initial setup, and we encountered no significant challenges when reproducing the results for the Toy Datasets, Special Functions, and the initial Continual Learning experiments. The provided examples and library functions were of great help to replicate the reported behaviors in these settings without major modifications.

However, the reproduction of the Feynman dataset experiments proved to be the most challenging aspect of this project. The primary hurdle was the high computational cost associated with training deep KANs. In particular, experiments involving network depths of 5 and 6 were infeasible within our resource constraints, as single training runs often exceeded 2 hours, rendering comprehensive parameter and seed sweeps impractical.

Furthermore, several critical methodological details were not specified in the original paper, including the exact number of training samples, random seeds, learning rates, and feature sampling ranges. While seemingly minor, our additional experiments (Appendix C) indicate that these hyperparameter choices can substantially affect performance. Due to these ambiguities, we relied on the default settings of the `pykan` implementation, which may differ from the specific tuning used to generate the reported results.

Finally, pruned KANs consistently underperformed relative to the results reported in the original paper. Consequently, we were forced to relax the RMSE threshold from 10^{-2} to 10^{-1} to obtain stable results. We speculate that this discrepancy may stem from differences in the pruning procedure: in our implementation, pruning was applied only after completing all grid expansions (with 200 optimization steps per grid), based on our interpretation of the text. It is possible that the original work performed pruning incrementally between grid refinements, though this detail was not explicitly documented.

7 Conclusion

We successfully replicated the core findings of Liu et al. (1), confirming that KANs achieve superior parameter efficiency and adhere to N^{-4} scaling laws on low-dimensional tasks. However, our experiments suggest that KANs are more susceptible to overfitting in high-parameter regimes than originally discussed. We also observed significant numerical instability when approximating functions with singularities (e.g., Bessel Y_ν, K_ν), a domain where MLPs proved more robust. Additionally, while unpruned KANs performed well on Feynman datasets, the pruning pipeline was difficult to reproduce. Finally, our novel extension to a 2D continual learning task demonstrated that the "locality" advantage of splines does not generalize beyond 1D, as KANs exhibited catastrophic forgetting similar to MLPs. Future work should focus on improving stability and regularization for deep KAN architectures.

8 Ethical consideration

KANs are a step towards more interpretable ML models. The original paper described a workflow where humans insert their expert knowledge into the model by fixing certain functions, and where

they learn from interpreting the model graph and it's corresponding symbolic function. This workflow bridges the gap between raw "black box" machine learning and manual scientific discovery, and can enable powerful machine learning to be used in scientific discovery while still keeping the humans able to have accountability over the discovered formulas.

9 Limitations

The original paper omits important reproducibility details, such as random seeds and specific pruning schedules, which we believe led to significant performance discrepancies in our experiments. We also found KANs to be computationally expensive, making deep architectures infeasible to train. Also, our reproduction revealed that KANs are highly susceptible to overfitting (displaying U-shaped error curves) and are numerically unstable on functions with singularities, where MLPs proved more robust. Finally, the reported resistance to catastrophic forgetting is limited, as it failed to generalize to our 2D extension when using a non-symbolic classification dataset.

10 Self Assessment

We believe this project deserves an A. We successfully reproduced the core experiments (Toy datasets, Special Functions and Feynman datasets) and went beyond the scope by designing a novel 2D continual learning extension with non-symbolic datasets, demonstrating that KANs' "locality" benefits do not generalize to multidimensional data. Crucially, we provided a rigorous critical analysis of reproducibility, uncovering unreported issues like overfitting and missing implementation details that significantly impact the reliability of the original paper's claims.

References

[1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold Networks," *arXiv preprint arXiv:2404.19756*, 2024.

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

[3] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Doklady Akademii Nauk SSSR*, vol. 114, pp. 953–956, 1957.

[4] S.-M. Udrescu and M. Tegmark, "AI Feynman: A physics-inspired method for symbolic regression," *Science Advances*, vol. 6, no. 16, eaay2631, 2020.

[5] S.-M. Udrescu, A. Tan, J. Feng, O. Neto, T. Wu, and M. Tegmark, "AI Feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4860–4871, 2020.

[6] W. Gao, Z. Gong, Z. Deng, F. Rong, C. Chen, and L. Ma, "TabKANet: Tabular Data Modeling with Kolmogorov–Arnold Network and Transformer," arXiv preprint arXiv:2409.08806, 2024. :contentReference[oaicite:1]index=1

[7] A. Eslamian, A. Afzal Aghaei, and Q. Cheng, "TabKAN: Advancing Tabular Data Analysis using Kolmogorov–Arnold Network," arXiv preprint arXiv:2504.06559, 2025. :contentReference[oaicite:2]index=2

[8] F. Zhang and X. Zhang, "GraphKAN: Enhancing Feature Extraction with Graph Kolmogorov–Arnold Networks," arXiv preprint arXiv:2406.13597, 2024. :contentReference[oaicite:3]index=3

[9] R. Bresson, G. Nikolentzos, G. Panagopoulos, M. Chatzianastasis, J. Pang, and M. Vazirgiannis, "KAGNNs: Kolmogorov–Arnold Networks meet Graph Learning," *Transactions on Machine Learning Research*, 2025. :contentReference[oaicite:4]index=4

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, et al., "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

Appendix

A Kolmogorov-Arnold Ketworks

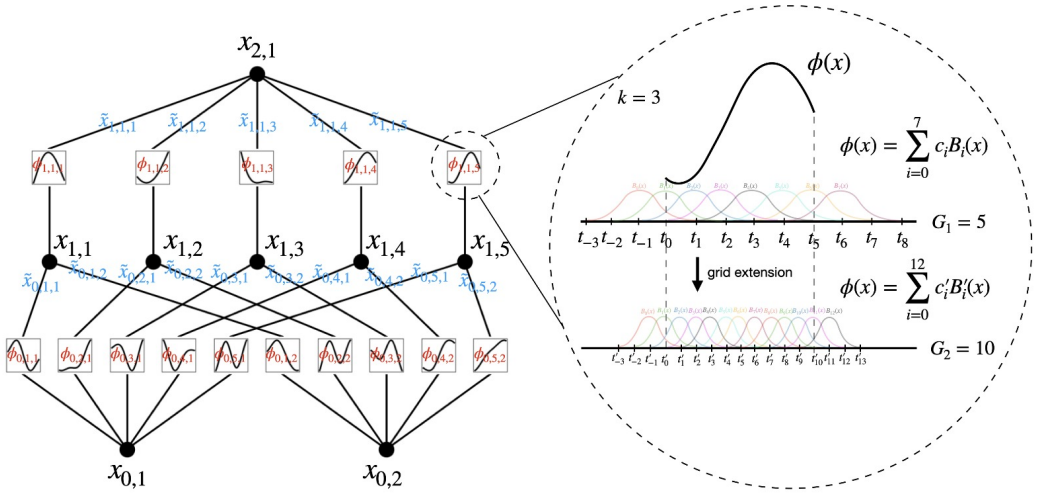

Figure 3: Left: Notations of activations that flow through the network. Right: an activation function is parameterized as a B-spline, which allows switching between coarse-grained and fine-grained grids. Reproduced from Liu et al. (1).

B Special functions

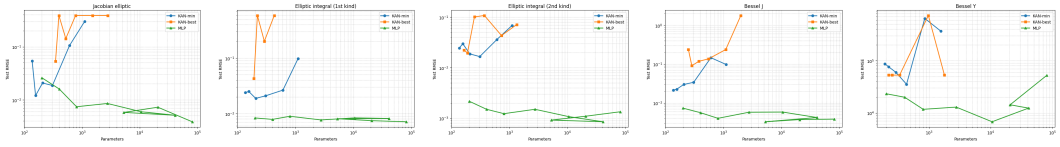

Figure 4: Performance comparison (Test RMSE vs. Parameters) for Jacobian elliptic functions, incomplete elliptic integrals, and standard Bessel functions (J_ν, Y_ν).

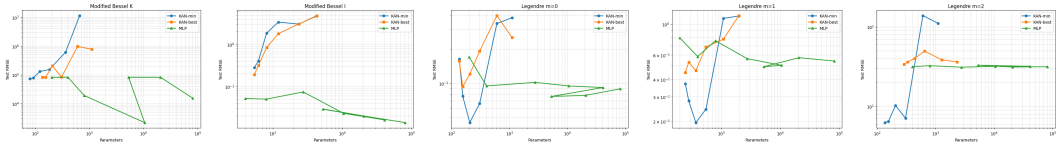

Figure 5: Performance comparison (Test RMSE vs. Parameters) for modified Bessel functions (K_ν, I_ν) and associated Legendre polynomials (P_l^m).

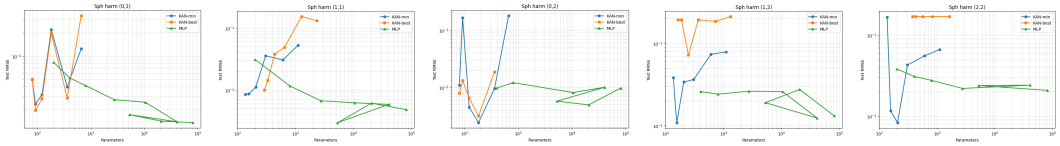

Figure 6: Performance comparison (Test RMSE vs. Parameters) for spherical harmonics (Y_l^m) of varying degrees and orders.

C Feynman datasets

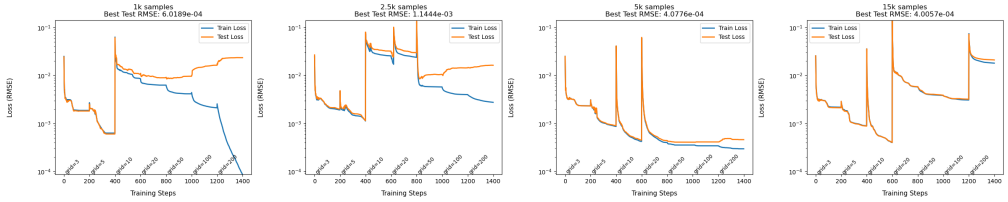

Figure 7: Train and Test losses (RMSE) for the Feynman dataset I.6.20 and a KAN [2,2,1], with different training samples. Test samples are set to 2k. Used seed=171.

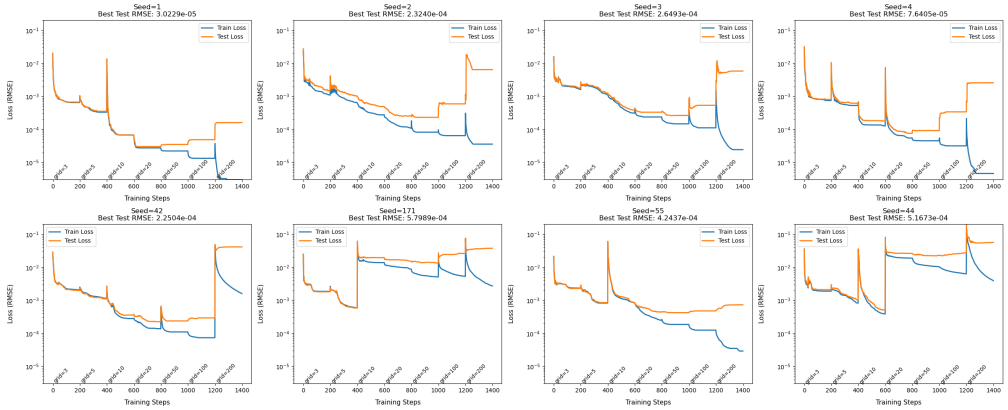

Figure 8: Train and Test losses (RMSE) for the Feynman dataset I.6.20 and a KAN [2,2,1], with different random seeds. Train and Test samples are set to 1k.

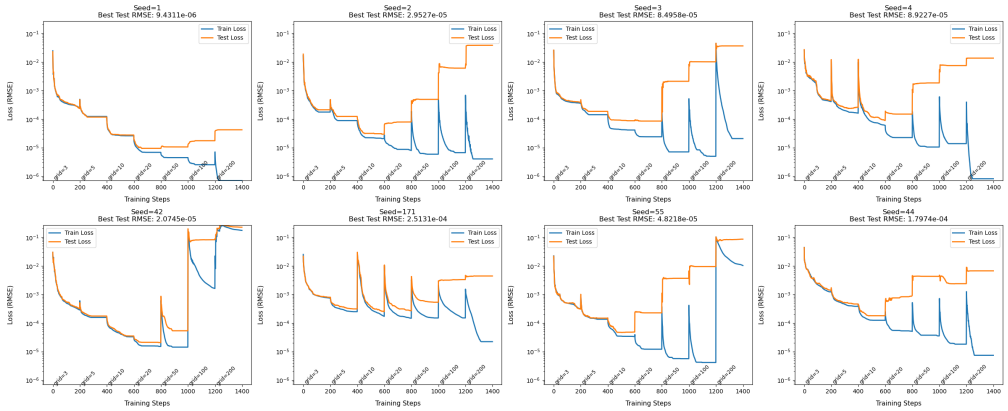

Figure 9: Same as Figure 8 but with KAN [2,2,2,1].

D Continual learning

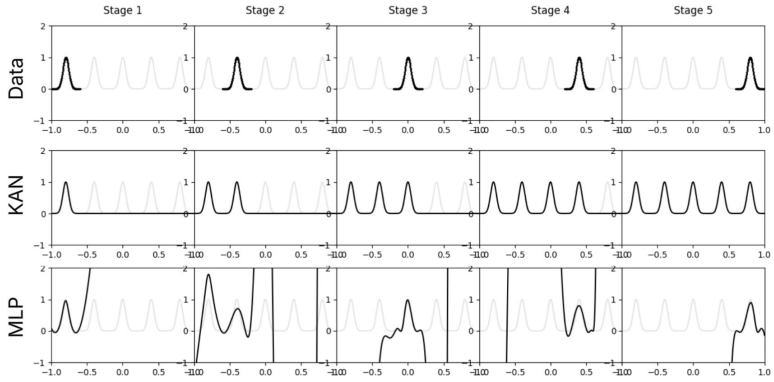

Figure 10: Continual learning problem. The networks are trained sequentially using the data from each stage. The findings that KANs can avoid catastrophic forgetting is consistent with the original paper.

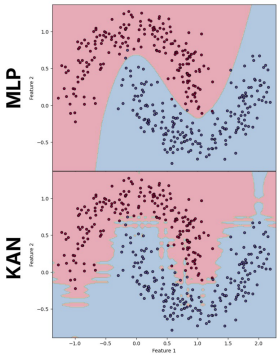

Figure 11: MLP and KAN classifiers with decision boundary. Trained on moons dataset.

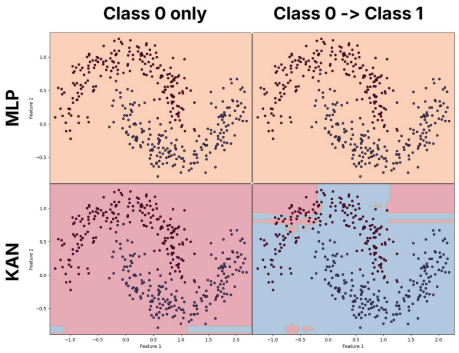

Figure 12: MLP and KAN classifiers with decision boundary. Trained sequentially with first only class 0 labels and then trained further with class 1 labels. No meaningful decision boundary was learned.