Kolmogorov-Arnold Networks Still Catastrophically Forget but Differently from MLP

Anton Lee[1], **Heitor Murilo Gomes**[1], **Yaqian Zhang**[2], **W. Bastiaan Kleijn**[1]

[1]School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand
[2]University of Waikato, Hamilton 3240, New Zealand
anton.lee@vuw.ac.nz, heitor.gomes@vuw.ac.nz, yaqian.zhang@waikato.ac.nz, bastiaan.kleijn@vuw.ac.nz

Abstract

Catastrophic forgetting is when a neural network loses previously learnt information after learning a new task sequentially. Avoiding catastrophic forgetting could reduce the resources necessary to update neural networks. Recently, Kolmogorov–Arnold Networks (KAN) gained the community's attention as preliminary experiments suggest KAN avoid catastrophic forgetting. KAN replace neural network edges with learnable B-splines and sum incoming edges in nodes. Proponents of KAN argue they avoid forgetting, are more accurate, are interpretable, and use fewer parameters. Our work investigates the claims that KAN avoid catastrophic forgetting, finding that they fail to do so on more complex datasets containing features that overlap between tasks. We give a simple explanation as to why and how KAN catastrophically forget. Motivated by evidence suggesting KAN are superior for symbolic regression, we augment KAN in the same ways as multilayer perceptron (MLP) to perform continual learning tasks, making special accommodations to support KAN. Our experiments found that unmodified KAN often forget more than MLP, but KAN can be better than MLP when combined with continual learning strategies. We aim to highlight some of the current shortcomings and strengths associated with KAN for continual learning.

Code — https://github.com/tachyonicClock/AAAI25-clkan

Introduction

Recently, Kolmogorov-Arnold Networks (KAN) (Liu et al. 2024) attracted attention within the deep learning research community, leading to hundreds of related works within a few months of its initial release. KAN replace edges in multilayer perceptron (MLP) with learnable 1D curves. Liu et al. (2024)'s simple continual learning experiment contributed to the excitement, suggesting KAN are more resilient to catastrophic forgetting—the longstanding problem in artificial neural networks where learning new information causes the forgetting of previously learnt information (McCloskey and Cohen 1989; French 1999; Kudithipudi et al. 2022). Catastrophic forgetting makes updating neural networks inefficient since a portion of each batch contains old samples to prevent forgetting rather than entirely new samples. If KAN's ability to overcome catastrophic forgetting goes beyond this initial simple experiment, it could lead to more efficient and more cheaply updated models. Alongside this demonstration, Liu et al. (2024) claim that KAN are more precise, interpretable, and have lower parameter counts than their MLP counterparts on symbolic regression datasets inspired by physics. This paper explores KAN for Continual Learning, discovering that KAN remain susceptible to catastrophic forgetting in most circumstances. Despite this, we modify KAN with a novel parameter isolation method and pre-existing regularisation strategies to take advantage of their other benefits. With these modifications, we discover that KAN give modest improvements on some regression problems over MLP, and without the modifications, KAN are more forgetful than MLP.

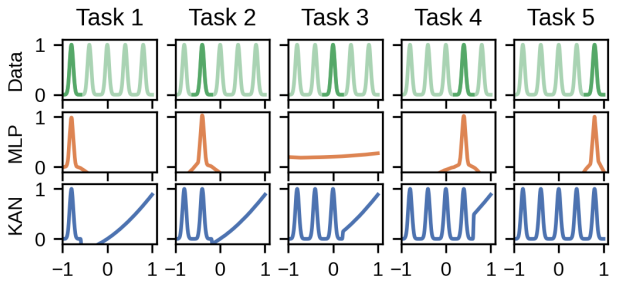

Figure 1: This figure replicates a key experiment of Liu et al. (2024). The experiment demonstrates that KAN avoid catastrophic forgetting in this specific scenario. However, we show that this advantage does not generalize to other scenario.

Figure 1 contrasts a KAN's and an MLP's performance on a simple continual learning regression problem. In it, we observe that KAN exhibit no catastrophic forgetting, unlike the MLP; this simple experiment inspired this paper. Liu et al. (2024) argues that KAN avoid catastrophic forgetting by leveraging the locality of splines: "...since spline bases are local, a sample will only affect nearby spline coefficients, leaving faraway coefficients intact (which is desirable since faraway regions may have already stored information that we want to preserve)." In other words, each coefficient in the spline is responsible for a particular interval. The first coefficient, for example, might only contribute when the input is

between $[-1.0, -0.9]$. Since backpropagation only touches coefficients that contribute to the output given the inputs, KAN avoids erroneous modifications on this dataset.

We found KAN's resistance to forgetting breaks down on more complex problems. Instead, we propose combining KAN with parameter-isolation-based continual learning using a novel procedure we name WiseKAN. We playfully derive the name from the adage that wisdom is knowing what you can and cannot change. The saying approximately describes how our parameter isolation methods work. Our method best suits continual learning regression problems because KAN can outperform MLP in those tasks (Liu et al. 2024).

We devised specialised accommodations to support isolating parameters in KAN. In contrast, we more easily combined KAN with model agnostic classic regularisation-based continual learners. As a baseline we construct WiseMLP, which is equivalent to WiseKAN but has had its KAN features removed. We found that the KAN variants often outperformed their MLP equivalent.

Most continual learning research focuses on classification (He and Sick 2021). However, catastrophic forgetting is equally problematic in regression tasks, where data distributions shift over time. Continual learning could be crucial, for example, in future Smart Grids or power forecasting applications when these models need to adapt quickly without forgetting recurring patterns (He and Sick 2021; He 2021).

We make the following contributions:

1. Devise a specialised parameter isolation procedure for KAN called **WiseKAN**. We show that WiseKAN usually outperforms KAN, WiseMLP and two classic regularisation continual learning algorithms.
2. Evaluate and critique KAN on broader task incremental regression problems and combine KAN with pre-existing continual learning strategies. Reporting results showing that combining KAN with pre-existing continual learning strategies (EWC and SI (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017)) outperform their MLP equivalents.
3. Explain and provide supporting evidence as to why "local plasticity" is insufficient for continual learning and why unmodified KAN are more forgetful than MLP.

Kolmogorov-Arnold Networks (KAN)

To ground our explanation in the familiar we contrast KAN to linear layers. The activation of individual output neuron j in some layer can be written $f_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$,

$$f_j(\mathbf{x}) = \sigma(\sum_i^n w_{j,i} x_i + b_j),$$

where: $x_i \in \mathbf{x}$ is a scalar from the layers input vector; $w_{j,i} \in \mathbf{W}$ is a scalar from a weight matrix; $b_j \in \mathbf{b}$ is a scalar from a bias vector; and σ is a non-linear activation function. We write this slightly differently from the usual vector valued function, $f(\mathbf{x}) = \mathbf{W}\mathbf{x} + \mathbf{b}$, for the sake of expressing KAN clearly, but it should be unmistakably a linear layer.

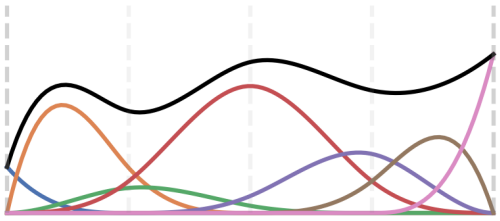

Figure 2: Cubic B-spline decomposed into 7 scaled basis functions. Vertical lines indicate knot locations defining the spline grid. ($p = 3$, $K = 7$, $grid = 5$)

In contrast to a linear layer, a KAN is:

$$f_j(\mathbf{x}) = \sum_i^n \text{spline}(x_i, \boldsymbol{c}_{j,i}),$$

where $\boldsymbol{c}_{j,i}$ is a vector of B-spline coefficients that define a continuous differentiable piece-wise polynomial connecting an input neuron i to an output neuron j. Figure 3, visualises this in a minimal network containing a single spline. KAN are parametrised by a vector per-edge rather than a scalar.

A B-spline is a linear combination of K coefficients and basis functions $B_{k,p} : \mathbb{R} \to \mathbb{R}$ of degree p,

$$\text{spline}(x_i, \boldsymbol{c}_{j,i}) = \sum_k^K c_{j,i,k} B_{k,p}(x_i),$$

where p is a hyper-parameter controlling smoothness, or degree of the polynomial. Figure 2 shows how multiple scaled B-spline basis functions combine to form a curve. Other efforts have also parametrised neural networks with B-spline (Bohra et al. 2020; Aziznejad et al. 2020).

Finally, in practice, Liu et al. (2024) uses residual connections and a scale parameter to make optimisation easier:

$$f_j(\mathbf{x}) = \sum_i^n w_{j,i}(\text{spline}(x_i, \boldsymbol{c}_{j,i}) + \text{silu}(x_i)) \qquad (1)$$

where $w_{j,i}$ is a scale parameter.

Related Work

Continual Learning

Continual learning algorithms are often grouped into families of strategies based on their main mechanism of overcoming catastrophic forgetting (Delange et al. 2021; Parisi et al. 2019; Wang et al. 2023). To give a brief overview: **architecture** methods add or isolate parameters (Mallya and Lazebnik 2018; Wortsman et al. 2020; Serrà et al. 2018; Kim et al. 2022a), **regularisation** methods modify the loss function (Kirkpatrick et al. 2017; Zenke, Poole, and Ganguli 2017; Li and Hoiem 2016), **optimisation** methods modify the optimisation program more deeply like meta-learning or constrained optimisation (Lopez-Paz and Ranzato 2017; Riemer et al. 2019), **replay** methods use stored or generated data, (Buzzega et al. 2020; Rebuffi et al. 2017; van de Ven,

Siegelmann, and Tolias 2020), and **representation** methods use pre-training or self-supervised learning (Zhou et al. 2024; Rebuffi et al. 2017).

WiseKAN is a fixed network size architecture method that isolates parameters for particular tasks. It is unique from related methods because we designed it specifically for KAN and employ a novel incremental pruning procedure.

Our paper compares WiseKAN against two classic regularisation methods, and a comparable architecture method. We do not compare against replay, representation, or optimisation because they usually introduce a replay buffer, pretrained model, or meta-learning phase that makes comparisons hard.

EWC and SI are regularisation techniques that preserve a model's knowledge of previous tasks by penalising significant deviations from past parameter values. Both EWC and SI introduce a regularisation term to the loss function:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{L}_{CE}(\boldsymbol{\theta}) + \lambda \sum_{i}^{N} \Omega_i (\theta_i - \theta_i^*)^2,$$

where $\theta_i \in \boldsymbol{\theta}$ are current parameters, $\theta_i^* \in \boldsymbol{\theta}^*$ are corresponding parameters saved from before the current task, $\Omega_i \in \boldsymbol{\Omega}$ quantifies each parameter's importance, and λ controls regularisation strength. $\mathcal{L}_{CE}$ is the usual unregularised loss function, such as cross entropy in classification. Weight regularisation methods differ in how to calculate this importance measure. Elastic weight consolidation (EWC) calculates $\boldsymbol{\Omega}$ as the diagonal of the fisher information matrix over the previous task (Kirkpatrick et al. 2017). Synaptic intelligence (SI) (Zenke, Poole, and Ganguli 2017) calculates importance from the sum of gradients during training—what they call a path integral. The idea is that weights that changes lots during training are important. Regardless of mechanism the idea is to keep important weights the same by anchoring them to the old model, while allowing unimportant weights to change freely.

KAN

Recent efforts have attempted to apply the benefits of KAN to broader problem settings with muddled results. One of the most promising research direction has been in applying KAN for time series forecasting where they have been shown to outperform MLP (Vaca-Rubio et al. 2024; Genet and Inzirillo 2024). Poeta et al. (2024) found that KAN are comparable or better than MLP on real world datasets experimenting with nine UCI classification datasets. Yu, Yu, and Wang (2024) is less optimistic concluding that: "Under the same number of parameters or FLOPs, we find KAN outperforms MLP only in symbolic formula representing, but remains inferior to MLP on other tasks of machine learning, computer vision, NLP, and audio processing." Shen et al. (2024) experiments "demonstrates that incorporating noise into the training data drastically diminishes the performance of Kolmogorov-Arnold Networks (KAN)." The trend of this research is that KAN are not a simple drop in upgrade for MLP.

Can KAN Learn Continually?

Liu et al. (2024) demonstration of KAN's ability to avoid catastrophic forgetting on simple 1D functions sparked this investigation (Figure 1). We, however, provide further analysis and discuss some technicalities missing from their discussion. Furthermore, our investigation found that KAN still suffers from catastrophic forgetting.

Spline Input Overlap and Catastrophic Forgetting. The continual learning experiments presented by Liu et al. (2024) use a network with zero hidden layers containing a singular high-resolution spline connecting the input to the output. Figure 3 shows such a network. This KAN avoids forgetting in Figure 1 because of "local plasticity", the trait that coefficients within a spline are only responsible for an interval of the inputs. Optimisation leaves coefficients that are not responsible for that interval unchanged. Liu et al. (2024) note, this is an "extremely simple example". Our evidence shows that KAN fail to generalise beyond it.

KAN's ability to continually learn does not generalise because of something we will call "**spline input overlap**." It occurs when the inputs to a spline from different tasks fall within an overlapping region of the spline's input. Figure 4 gives a trivial example of catastrophic forgetting occurring because of a small region of spline input overlap between tasks. Spline input overlap causes the optimiser to erroneously modify coefficients that are significant to a previous task.

The trouble with "spline input overlap" is that it is often unavoidable and even desired because it allows for generalisation and reuse. For example, a network that learns low-level features, such as lines in images, can only reuse those features if their spline inputs overlap with future tasks. When the network requires a compositional distributed representation, catastrophic forgetting occurs again because in these circumstances, spline inputs overlap.

Spline Input Ranges and Normalisation. KAN splines are non-zero only for specific input ranges, but inputs often exceed these boundaries. Liu et al. (2024) address this by dynamically adjusting splines based on recent samples. Unfortunately, this is unsuitable for continual learning as it can destroy performance on past tasks. Their imperfect solution is to restrict inputs to set ranges.

WiseKAN tries three alternatives: layer normalisation, task-specific batch normalisation, and keeping the grid constant. Task-specific batch normalisation is regular batch normalisation but maintains different statistics for each task. We found that the choice of normalisation depends on the dataset. We ablate the choice of normalisation for WiseKAN in the technical appendix.

Scale Weights KAN. KAN include a weight scale applied to the entire spline for optimisation efficiency. Specifically the parameter $w_{j,i}$ from Equation 1, causes catastrophic forgetting because changes to it change the scale of the spline globally, impacting already learned spline coefficients. WiseKAN freezes and prunes this parameter as part of the spline.

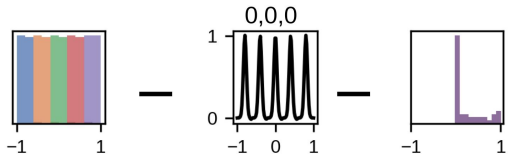

Figure 3: KAN containing a single spline used to produce *Figure 1*. KAN nodes are plotted as a histogram of activations and edges show the literal spline.

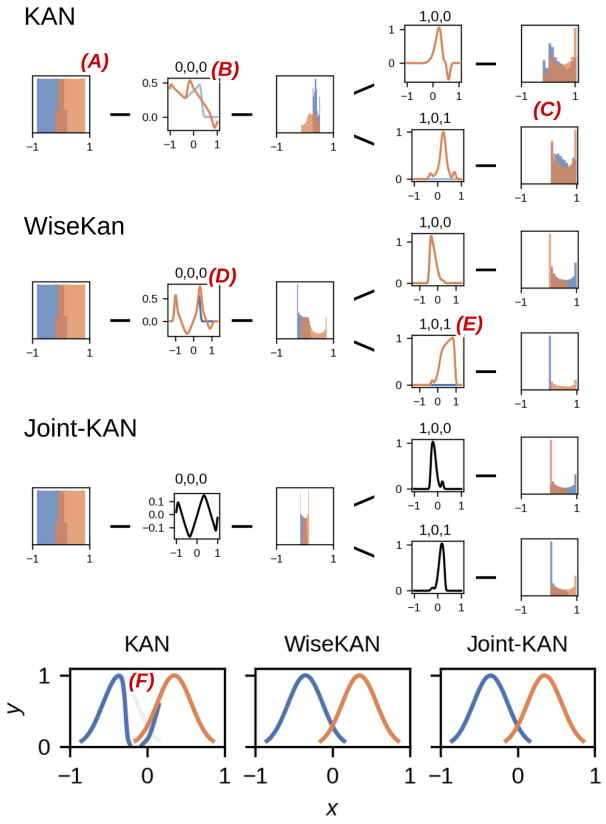

Figure 4: **(A):** The dataset contains two tasks (blue and orange) slightly overlapping in their input. **(B):** After training on each task, the graph plots the spline in its respective colour. The graph shows the spline learnt for the blue task was changed for the orange task, causing the forgetting in (F). **(C):** Each task has a dedicated output such that the model is not expected to infer the task from the data. **(D):** WiseKAN avoids inference by isolating parameters for particular tasks. Here only the zeroed coefficients are modified in the subsequent tasks. Since WiseKAN tracks which parameters are assigned to particular task both the orange and blue spline can be enabled by specifying the task id, using our "visibility rule". **(E):** WiseKAN pruned this entire node after the first task. **(F):** KAN exhibits catastrophic forgetting when spline inputs overlap. WiseKAN avoids forgetting by isolating important parameters. Joint-KAN avoids forgetting by *not* continually learning fitting both tasks simultaneously.

WiseKAN Algorithm

KAN, while believed to be more accurate and interpretable than MLP for certain regression tasks (Liu et al. 2024), are not inherently resistant to catastrophic forgetting. To benefit from the other benefit we propose WiseKAN a novel parameter isolation strategy tailored for KAN.

The technical appendix contains algorithm pseudo code, while this section summarises how WiseKAN functions.

Stopping Forgetting

WiseKAN dedicates parameters to specific tasks to avoid catastrophic forgetting. An integer allocation mask vector $\boldsymbol{m}$ records the task each of the N parameters belongs to. Using the mask we apply the following two rules:

1. **Visibility Rule:** Parameters allocated to a past task are visible to subsequent tasks during both training and evaluation. Formally we can compute a parameter visibility binary mask $\mathbf{v}(\boldsymbol{m}, t) = \langle v(m_1, t), ..., v(m_N, t) \rangle$ using the function:

$$v(m_i, t) = \begin{cases} 1 & m_i \leq t \\ 0 & \text{otherwise.} \end{cases}$$

   Element-wise multiplication uses the mask to enact the visibility rule: $\theta^* = \theta \odot \mathbf{v}(\boldsymbol{m}, t)$ where t is the current or historic task identity. The visibility rule temporarily zeros invisible parameters. Each task reuses old parameters and representation since they are visible despite being frozen.

2. **Freezing Rule:** WiseKAN freezes parameters allocated to earlier tasks to avoid forgetting, making them immutable. We can compute the mask of frozen parameters $\boldsymbol{r}(\boldsymbol{m}, t) = \langle r(m_1, t), ..., r(m_N, t) \rangle$ using the function:

$$r(m_i, t) = \begin{cases} 1 & m_i \neq t \\ 0 & \text{otherwise.} \end{cases}$$

   In the backwards pass the gradient vector $\nabla \mathcal{L}_\theta$ is modified so only nonfrozen parameters are changed $\nabla \mathcal{L}^*_\theta = (1 - \boldsymbol{r}(\boldsymbol{m}, t)) \odot \nabla \mathcal{L}_\theta$, where t is the current task.

Initially $\boldsymbol{m}$ is filled with ones so that all parameters belong to the first task. WiseKAN performs pruning to update $\boldsymbol{m}$ so that parameters are available for future tasks.

Pruning KAN

WiseKAN prunes parameters by assigning them to the next task $m_i \leftarrow t + 1$, where t is the current task index. Parameters eligible for pruning are those assigned to the current task $m_i = t$. The current sparsity level is the number assigned to the current task $m_i = t$ divided by the number assigned to either the current task $m_i = t$ or the next task $m_i = t + 1$. By assigning parameter to future tasks sparsity and capacity for future tasks increases.

Unlike linear layers where any weight is roughly equivalent to any other weight, in KAN there is structure and hierarchy because coefficients belong to splines. There are two levels where sparsity and pruning is desirable: entire splines and coefficients within splines. This necessitates a novel pruning procedure for KAN.

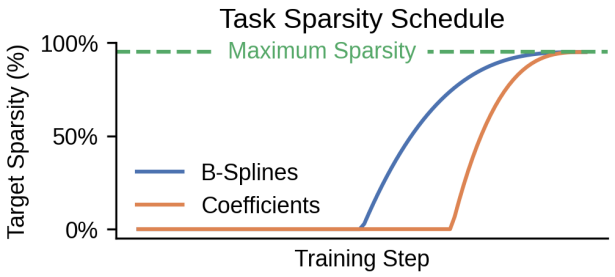

Figure 5: Zhu and Gupta (2017) automated gradual pruning algorithm modified for KAN. The target and maximum sparsity's are rarely achieved because WiseKAN will stop pruning early to avoid over-pruning when performance degrades on a validation set.

WiseKAN employs a hierarchical iterative pruning procedure that treats both splines and coefficients within splines as prunable. WiseKAN initial coarsely prunes at the spline level before pruning at finer levels. The motivation is to anneal pruning, initially removing many redundant connections aggressively, and then slowing the pruning rate to refine the model. This is achieved by scheduling two of Zhu and Gupta (2017) sparsity functions, designed to anneal pruning, to separately control the rate of pruning for splines and coefficients. Their function increases sparsity from zero to a final sparsity value s_{final} over S pruning steps:

$$s_i = \begin{cases} s_{\text{final}} - s_{\text{final}}(1 - \frac{1}{S}(i - i_{\text{start}}))^3 & i > i_{\text{start}} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where i is the current training step and i_{start} is the configured first step where pruning should occur. The pruning iteration is reset with each task. We use each epoch since the start of the task as the pruning step. Figure 5 depicts how we schedule pruning B-spline and coefficients.

WiseKAN prunes spline with the smallest mean absolute magnitude of the spline's coefficients. The individual coefficients within the splines are also pruned based on their absolute magnitude. Weight magnitude pruning is simple yet effective because it essentially snaps weights near zero to zero (Han et al. 2015; Gale, Elsen, and Hooker 2019). One interaction between these two separate pruning procedures is that pruning a spline will prune its component coefficients.

WiseKAN balances pruning to optimise current and future performance. Excessive pruning creates capacity for new tasks but harms current performance. To address this trade-off, we employ early stopping. When the current tasks validation performance drops by a configured threshold (like 5-10%), we roll back the model and halt pruning the splines. Afterwards WiseKAN prunes only the coefficients within splines until performance drops again, triggering another rollback and halting pruning for the remainder of training. WiseKAN progresses from coarse to fine pruning, gradually reducing the amount of pruning from entire splines to just coefficients.

WiseMLP incorporates all of WiseKAN features but removes KAN-specific accommodations. It is still different from pre-existing parameter-isolation methods, such as PackNet (Mallya and Lazebnik 2018), because it incorporates WiseKAN's adaptive incremental pruning procedure outlined above.

Results

We use random search to explore hyper-parameters optimising for R2 (coefficient of determination) and parameter count. Our R2 value is an average over the course of tasks, it is consistent with Díaz-Rodríguez et al. (2018) "average accuracy" metric but adapted for R2. Figure 6 shows the Pareto front of these results. Details on the search space are in the technical appendix.

Table 2, 3, 4 contain average results of five runs of the best scoring hyper-parameters regardless of parameter count (N). Concrete values are reported in the technical appendix. We mark statically significant (two-sided unequal variance t-test $p < 0.05$) cases where a method outperforms its counterpart KAN or MLP with a "*". (PackNet does not have a counterpart KAN, since it is incompatible with KAN).

The non-continual upper baselines strategies Joint-MLP and Joint-KAN train on all tasks at once.

We implemented the hyper-parameter search with *Optuna* (Akiba et al. 2019) and *Hydra* (Yadan 2019). Our KAN used a modified version of *EfficientKAN* (Blealtan and Dash 2024). The technical appendix includes other details of the hardware and software used.

Datasets

We focus on multi-headed, or task-incremental (**TI**), scenarios where the evaluation phase provides the task associated with each data point (Van De Ven, Tuytelaars, and Tolias 2022). Knowing the task simplifies the problem by eliminating the need for the model to infer both the task and the target simultaneously. By focusing on task incremental learning, we focus on catastrophic forgetting in isolation. Typically, implementations achieve this by swapping the neural network's final layer (hence multi-headed) for each new task or masking the output layer.

TI Feynman re-purpose the Feynman Symbolic Regression Database (Udrescu and Tegmark 2020) used to benchmark KAN in original paper (Liu et al. 2024) for continual learning. We turn it into a continual learning problem splitting each equation into a task. TI Feynman is used because we know KAN perform well on symbolic regression. Since the inputs cover similar domains we chose to give each task its own set of inputs, zeroing the other unused inputs.

TI Europe Wind Farm (Gensler 2016; He and Sick 2021) dataset contains hourly wind power generation and day-ahead wind speed forecasts for 45 European wind farms. The goal is to predict power output from forecasted weather. For continual learning, we split the dataset into eight tasks, each equating to a quarter of a year. TI Europe Wind Farm has been used before in the continual learning regression niche, and there is a strong argument why continual learning is useful for this application (He and Sick 2021; He 2021).

TI River Radar (Nick Lim 2023) dataset's objective is to predict rain gauges and the height of a river at two locations.

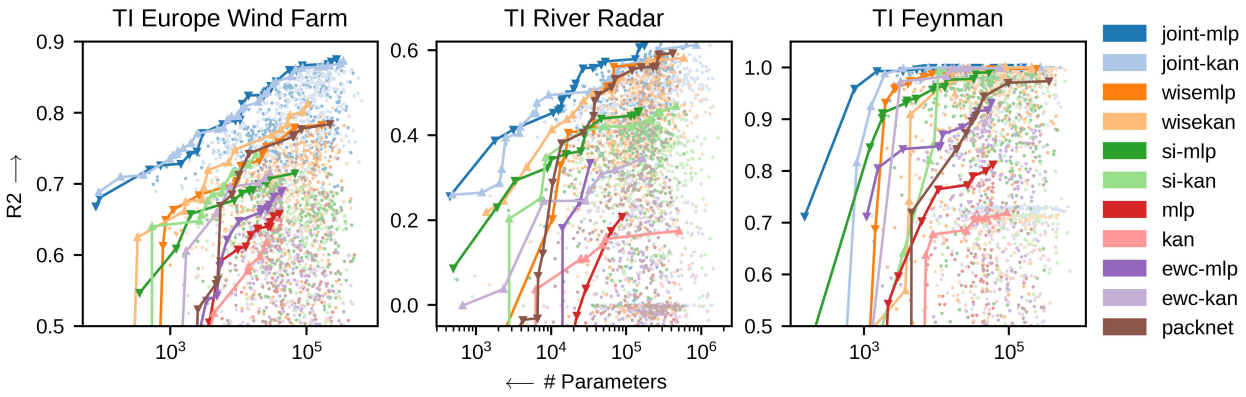

Figure 6: Pareto front that minimises parameter count and maximises R2 score. Non-Pareto optimal runs are plotted as small dots. Pareto optimal runs, forming the Pareto front, dominate non-Pareto optimal runs by outperforming them in all objectives. Random search ran each algorithm 400 times.

The learner is given rain radar reflectance measured across a grid at three altitudes. The dataset is split into seven tasks representing roughly four month chunks. We used TI River Radar because it is a higher dimensional computer vision like regression problem with similar seasonality to TI Europe Wind Farm. It is potentially helpful for hydroelectricity and "smart grids", where adaptability without forgetting is desirable.

Metrics

Continual learning metrics consider performance over each task throughout training. This can be represented in a matrix $R_{i,j}$ where j is the performance of the model on task j after training on task i. We use Díaz-Rodríguez et al. (2018) variants of "average accuracy" and "backwards transfer" that we adapt for regression into $R2$ and $R2_{bwt}$ respectively by using R2 instead of classification accuracy.

Scenario Average ($R2$) measures the average performance on past tasks over each task.

$$SA = \frac{\sum_{i \geq j}^{T} R_{i,j}}{\frac{T(T+1)}{2}}$$

Backward transfer ($R2_{bwt}$) measures the influence learning a task has on previous tasks, where negative values indicate catastrophic forgetting.

$$BWT = \frac{\sum_{i=2}^{T} \sum_{j=1}^{i-1} (R_{i,j} - R_{j,j})}{\frac{T(T-1)}{2}}$$

In addition to these variants of standard metrics, we report the model's ability to adapt to each task regardless of its ability to avoid catastrophic forgetting:

Plasticity ($R2_{on}$) evaluates the model's performance on each new task post-training:

$$ON = \frac{\sum_{i}^{T} R_{i,i}}{T}$$

Discussion

WiseKAN, WiseMLP and Packnet are all parameter isolation methods that are immune to forgetting in our setting. That is their $R2_{bwt}$ is always zero since they all isolate parameters using the "freezing rule". We found that WiseKAN usually beats WiseMLP, and WiseMLP usually beats PackNet. This implies that WiseKAN is capable of taking advantage of the expressive KAN without catastrophic forgetting.

A surprising finding is that the KAN variants of classic weight regularisation strategies outperform their MLP counterparts. This is primarily true at higher parameter counts. In contrast, regular KAN under-perform in comparison to MLP.

In our hyper-parameter search (Figure 6) R2 is often negative and occasionally negative because of catastrophic forgetting. Negative R2 indicates that the model performs worse than a naive model that predicts the mean. It occurs because in naive cases there is no mechanism that keeps these past outputs in a reasonable range.

In the tables we observe that KAN and MLP sometimes have lower $R2_{on}$ than continual learning strategies. This is because to achieve good scenario average R2 lower learning rates are chosen for these models leading to reduced plasticity.

Why Can KAN Be More Forgetful?

In Figure 6 we observe that the KAN pareto front is often more forgetful than MLP. Likewise Table 2, 3, 4 reflect this finding. Yu, Yu, and Wang (2024) also reports that KAN are more forgetful than MLP on computer vision tasks. We hypothesise this to be related to the width of the network and the corresponding "gradient sparsity" by building on observations from MLP (Mirzadeh et al. 2021).

Figure 7 illustrates the gradient distribution of KAN and MLP from the final task of the EuropeWind dataset. Similar to MLP (Mirzadeh et al. 2021), we observe that the sparsity of gradients in KAN is influenced by network width: wider KAN have sparser gradients and suffer less from catas-

	Tasks	$\|\mathbf{x}\|$	$\|\mathbf{y}\|$	$\|\mathbf{y}^*\|$	Count
TI Feynman	34	83	34	1	3.4M
TI Europe Wind Farm	8	78	8	1	45k
TI River Radar	7	442	28	4	86k

Table 1: Summary of datasets. Where: $|\mathbf{x}|$ is the number of input features; $|\mathbf{y}|$ is the number of output features; and $|\mathbf{y}^*|$ is the number of output features for a specific task.

Method (N)	R2 ↑	$R2_{bwt}$ ↑	$R2_{on}$ ↑
joint-mlp (155k)	$0.9998 \pm .0001$	NA	NA
joint-kan (274k)	$\mathbf{0.9999 \pm .0001}$	NA	NA
wisemlp (308k)	$0.9958 \pm .0016$	$\mathbf{0.000 \pm .000}$	$0.996 \pm .001$
wisekan* (177k)	$\mathbf{0.9994 \pm .0002}$	$\mathbf{-0.000 \pm .000}$	$\mathbf{0.999 \pm .000}$
si-mlp (53k)	$0.9896 \pm .0012$	$-0.001 \pm .001$	$0.992 \pm .000$
si-kan* (71k)	$0.9992 \pm .0002$	$\mathbf{0.000 \pm .000}$	$\mathbf{0.999 \pm .000}$
ewc-mlp (58k)	$0.9065 \pm .0164$	$-0.080 \pm .017$	$0.984 \pm .001$
ewc-kan* (77k)	$0.9983 \pm .0009$	$-0.001 \pm .001$	$\mathbf{0.999 \pm .000}$
mlp* (60k)	$0.7893 \pm .0122$	$-0.120 \pm .013$	$0.901 \pm .001$
kan (94k)	$0.7046 \pm .0158$	$-0.213 \pm .017$	$0.910 \pm .001$
packnet (364k)	$0.9327 \pm .0347$	$\mathbf{0.000 \pm .000}$	$0.886 \pm .040$

Table 2: TI Feynman (Udrescu and Tegmark 2020). Where N is the parameter count.

Method (N)	R2 ↑	$R2_{bwt}$ ↑	$R2_{on}$ ↑
joint-mlp (276k)	$0.869 \pm .004$	NA	NA
joint-kan (348k)	$\mathbf{0.870 \pm .003}$	NA	NA
wisemlp (278k)	$0.781 \pm .005$	$\mathbf{0.000 \pm .000}$	$0.782 \pm .003$
wisekan* (289k)	$\mathbf{0.794 \pm .006}$	$\mathbf{0.000 \pm .000}$	$\mathbf{0.785 \pm .011}$
si-mlp (68k)	$0.699 \pm .002$	$-0.001 \pm .002$	$0.698 \pm .002$
si-kan* (18k)	$0.733 \pm .002$	$\mathbf{-0.000 \pm .000}$	$0.736 \pm .001$
ewc-mlp (44k)	$0.680 \pm .005$	$-0.030 \pm .007$	$0.694 \pm .004$
ewc-kan* (29k)	$0.709 \pm .008$	$-0.063 \pm .005$	$0.755 \pm .009$
mlp* (40k)	$0.651 \pm .014$	$-0.067 \pm .017$	$0.715 \pm .002$
kan (26k)	$0.629 \pm .012$	$-0.134 \pm .019$	$0.742 \pm .004$
packnet (255k)	$0.782 \pm .004$	$\mathbf{0.000 \pm .000}$	$0.769 \pm .005$

Table 3: TI Europe Wind Farm (Gensler 2016).

Method (N)	R2 ↑	$R2_{bwt}$ ↑	$R2_{on}$ ↑
joint-mlp (442k)	$\mathbf{0.623 \pm .018}$	NA	NA
joint-kan (869k)	$0.619 \pm .010$	NA	NA
wisemlp (525k)	$0.582 \pm .017$	$\mathbf{0.000 \pm .000}$	$0.543 \pm .018$
wisekan (710k)	$0.539 \pm .059$	$\mathbf{0.000 \pm .000}$	$0.487 \pm .059$
si-mlp (149k)	$0.446 \pm .012$	$\mathbf{-0.000 \pm .003}$	$0.421 \pm .008$
si-kan* (461k)	$0.464 \pm .009$	$\mathbf{0.000 \pm .001}$	$0.422 \pm .012$
ewc-mlp (33k)	$0.217 \pm .069$	$-0.262 \pm .097$	$0.380 \pm .006$
ewc-kan (166k)	$0.289 \pm .060$	$-0.211 \pm .070$	$0.412 \pm .009$
mlp (89k)	$0.109 \pm .062$	$-0.400 \pm .085$	$0.413 \pm .002$
kan (508k)	$0.159 \pm .019$	$-0.283 \pm .024$	$0.378 \pm .001$
packnet (523k)	$\mathbf{0.587 \pm .004}$	$\mathbf{0.000 \pm .000}$	$\mathbf{0.546 \pm .004}$

Table 4: TI River Radar (Nick Lim 2023).

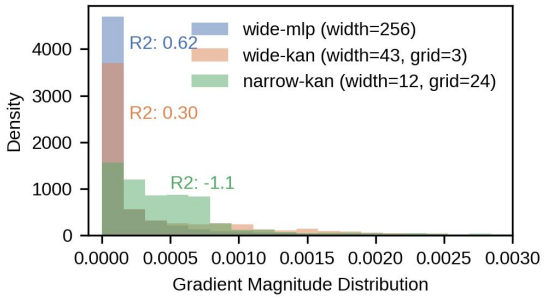

Figure 7: A histogram of the absolute magnitude of the gradient vector on the Europe Wind Dataset. Sparser gradient updates contain more zero or near zero updates and fewer larger changes, skewing the distribution further. All networks contain one hidden layer and 22k parameters.

trophic forgetting. However, when comparing KAN to MLP with the same number of parameters, the KAN architecture necessitates a trade-off between spline resolution and network width. Holding the parameter count constant, the width of KAN layers is inversely proportional to the spline resolution (grid). KAN's narrower structure may explain its higher forgetting rate compared to MLP, despite local plasticity.

We observe that KAN outperforms MLP when combined with methods to mitigate catastrophic forgetting. The improvement is likely because, in these instances, KAN benefits from the expressiveness of its learnable splines and possibly "local plasticity", while forgetting is mitigated by another mechanism.

Limitations

Going beyond a task incremental setting will require inferring the task ID which is left as future work. Inferring tasks for computer vision has been explored in prior works (Kim et al. 2022b; Kim, Liu, and Ke 2022; Aljundi, Chakravarty, and Tuytelaars 2017).

We limit the scope by focusing on classic methods and our novel KAN-specific approach, leaving broader comparisons for future work. Furthermore, WiseKAN explicitly favours avoiding forgetting over learning and will stop learning when its finite capacity is exhausted. This limitation may be overcome by supporting graceful forgetting (Golkar, Kagan, and Cho 2019), or adding more parameters (Rusu et al. 2016).

Conclusion

KAN catastrophically forget differently from MLP. While they can outperform MLP when paired with continual learning techniques, they also suffer more catastrophic forgetting when used by themselves due to their narrower width and less sparse gradients. However, we are sceptical about whether the modest performance gains justify KAN's added complexity. KAN remain exciting because they may popularise the study of more diverse neural network architectures, which could lead to novel approaches for mitigating catastrophic forgetting.

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Aljundi, R.; Chakravarty, P.; and Tuytelaars, T. 2017. Expert Gate: Lifelong Learning with a Network of Experts. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7120–7129.

Aziznejad, S.; Gupta, H.; Campos, J.; and Unser, M. 2020. Deep neural networks with trainable activations and controlled lipschitz constant. *IEEE Transactions on Signal Processing*, 68: 4688–4699.

Blealtan; and Dash, A. 2024. An Efficient Implementation of Kolmogorov-Arnold Network. https://github.com/Blealtan/efficient-kan. Accessed: 2025-01-14.

Bohra, P.; Campos, J.; Gupta, H.; Aziznejad, S.; and Unser, M. 2020. Learning Activation Functions in Deep (Spline) Neural Networks. *IEEE Open Journal of Signal Processing*, 1: 295–309.

Buzzega, P.; Boschini, M.; Porrello, A.; Abati, D.; and CALDERARA, S. 2020. Dark Experience for General Continual Learning: a Strong, Simple Baseline. In *Advances in Neural Information Processing Systems*, volume 33, 15920–15930. Curran Associates, Inc.

Delange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; and Tuytelaars, T. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

Díaz-Rodríguez, N.; Lomonaco, V.; Filliat, D.; and Maltoni, D. 2018. Don't forget, there is more than forgetting: new metrics for Continual Learning.

French, R. M. 1999. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4): 8.

Gale, T.; Elsen, E.; and Hooker, S. 2019. The State of Sparsity in Deep Neural Networks. *arXiv:1902.09574 [cs, stat]*.

Genet, R.; and Inzirillo, H. 2024. TKAN: Temporal Kolmogorov-Arnold Networks. *ArXiv*, abs/2405.07344.

Gensler, A. 2016. Europe Wind Farm Dataset. https://www.uni-kassel.de/eecs/ies/downloads. Accessed: 2025-01-14.

Golkar, S.; Kagan, M.; and Cho, K. 2019. Continual learning via neural pruning. *CoRR*, abs/1903.04476.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

He, Y. 2021. Adaptive Explainable Continual Learning Framework for Regression Problems with Focus on Power Forecasts. *ArXiv*, abs/2108.10781.

He, Y.; and Sick, B. 2021. CLeaR: An Adaptive Continual Learning Framework for Regression Tasks. *AI Perspectives*, 3(1): 2.

Kim, G.; Esmaeilpour, S.; Xiao, C.; and Liu, B. 2022a. Continual learning based on OOD detection and task masking. In *IEEE/CVF conference on computer vision and pattern recognition workshops, CVPR workshops 2022, new orleans, LA, USA, june 19-20, 2022*, 3855–3865. IEEE.

Kim, G.; Liu, B.; and Ke, Z. 2022. A Multi-Head Model for Continual Learning via Out-of-Distribution Replay. In *Proceedings of The 1st Conference on Lifelong Learning Agents*, 548–563. PMLR.

Kim, G.; Xiao, C.; Konishi, T.; Ke, Z.; and Liu, B. 2022b. A theoretical study on solving continual learning. In Oh, A. H.; Agarwal, A.; Belgrave, D.; and Cho, K., eds., *Advances in neural information processing systems*.

Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A. A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; Hassabis, D.; Clopath, C.; Kumaran, D.; and Hadsell, R. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13): 3521–3526.

Kudithipudi, D.; Aguilar-Simon, M.; Babb, J.; Bazhenov, M.; Blackiston, D.; Bongard, J.; Brna, A. P.; Chakravarthi Raja, S.; Cheney, N.; Clune, J.; Daram, A.; Fusi, S.; Helfer, P.; Kay, L.; Ketz, N.; Kira, Z.; Kolouri, S.; Krichmar, J. L.; Kriegman, S.; Levin, M.; Madireddy, S.; Manicka, S.; Marjaninejad, A.; McNaughton, B.; Miikkulainen, R.; Navratilova, Z.; Pandit, T.; Parker, A.; Pilly, P. K.; Risi, S.; Sejnowski, T. J.; Soltoggio, A.; Soures, N.; Tolias, A. S.; Urbina-Meléndez, D.; Valero-Cuevas, F. J.; van de Ven, G. M.; Vogelstein, J. T.; Wang, F.; Weiss, R.; Yanguas-Gil, A.; Zou, X.; and Siegelmann, H. 2022. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3): 196–210.

Li, Z.; and Hoiem, D. 2016. Learning without forgetting. *CoRR*, abs/1606.09282.

Liu, Z.; Wang, Y.; Vaidya, S.; Ruehle, F.; Halverson, J.; Soljačić, M.; Hou, T. Y.; and Tegmark, M. 2024. KAN: Kolmogorov-Arnold Networks.

Lopez-Paz, D.; and Ranzato, M. A. 2017. Gradient Episodic Memory for Continual Learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Mallya, A.; and Lazebnik, S. 2018. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7765–7773. Salt Lake City, UT: IEEE. ISBN 978-1-5386-6420-9.

McCloskey, M.; and Cohen, N. J. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*, volume 24, 109–165. Elsevier. ISBN 978-0-12-543324-2.

Mirzadeh, S. I.; Chaudhry, A.; Hu, H.; Pascanu, R.; Gorur, D.; and Farajtabar, M. 2021. Wide Neural Networks Forget Less Catastrophically. *arXiv*.

Nick Lim, S. D., Phil Mourot. 2023. River radar and sensor data. https://datasets.cms.waikato.ac.nz/taiao/river_radar_2015_2018/. Accessed: 2025-01-14.

Parisi, G. I.; Kemker, R.; Part, J. L.; Kanan, C.; and Wermter,

S. 2019. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113: 54–71.

Poeta, E.; Giobergia, F.; Pastor, E.; Cerquitelli, T.; and Baralis, E. 2024. A Benchmarking Study of Kolmogorov-Arnold Networks on Tabular Data.

Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; and Lampert, C. H. 2017. iCaRL: Incremental Classifier and Representation Learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Riemer, M.; Cases, I.; Ajemian, R.; Liu, M.; Rish, I.; Tu, Y.; and Tesauro, G. 2019. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *ICLR*.

Rusu, A. A.; Rabinowitz, N. C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; and Hadsell, R. 2016. Progressive Neural Networks. *arXiv:1606.04671 [cs]*.

Serrà, J.; Suris, D.; Miron, M.; and Karatzoglou, A. 2018. Overcoming catastrophic forgetting with hard attention to the task. In Dy, J. G.; and Krause, A., eds., *Proceedings of the 35th international conference on machine learning, ICML 2018, stockholmsmässan, stockholm, sweden, july 10-15, 2018*, volume 80 of *Proceedings of machine learning research*, 4555–4564. PMLR.

Shen, H.; Zeng, C.; Wang, J.; and Wang, Q. 2024. Reduced Effectiveness of Kolmogorov-Arnold Networks on Functions with Noise.

Udrescu, S.-M.; and Tegmark, M. 2020. AI Feynman: a Physics-Inspired Method for Symbolic Regression.

Vaca-Rubio, C. J.; Blanco, L.; Pereira, R.; and Caus, M. 2024. Kolmogorov-Arnold Networks (KANs) for Time Series Analysis. *ArXiv*, abs/2405.08790.

van de Ven, G. M.; Siegelmann, H. T.; and Tolias, A. S. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, 11(1): 4069.

Van De Ven, G. M.; Tuytelaars, T.; and Tolias, A. S. 2022. Three types of incremental learning. *Nature Machine Intelligence*, 4(12): 1185–1197.

Wang, L.; Zhang, X.; Su, H.; and Zhu, J. 2023. A Comprehensive Survey of Continual Learning: Theory, Method and Application.

Wortsman, M.; Ramanujan, V.; Liu, R.; Kembhavi, A.; Rastegari, M.; Yosinski, J.; and Farhadi, A. 2020. Supermasks in superposition. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.-F.; and Lin, H.-T., eds., *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, NeurIPS 2020, december 6-12, 2020, virtual*.

Yadan, O. 2019. Hydra - A framework for elegantly configuring complex applications. https://github.com/facebookresearch/hydra. Accessed: 2024.

Yu, R.; Yu, W.; and Wang, X. 2024. KAN or MLP: A Fairer Comparison.

Zenke, F.; Poole, B.; and Ganguli, S. 2017. Continual Learning Through Synaptic Intelligence. In *International Conference on Machine Learning*, 3987–3995.

Zhou, D.-W.; Sun, H.-L.; Ning, J.; Ye, H.-J.; and Zhan, D.-C. 2024. Continual learning with pre-trained models: A survey. In *Proceedings of the thirty-third international joint conference on artificial intelligence, IJCAI 2024, jeju, south korea, august 3-9, 2024*, 8363–8371. ijcai.org.

Zhu, M.; and Gupta, S. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression.

Acknowledgements

The author(s) wish to acknowledge the use of New Zealand eScience Infrastructure (NeSI) high performance computing facilities, consulting support and/or training services as part of this research. New Zealand's national facilities are provided by NeSI and funded jointly by NeSI's collaborator institutions and through the Ministry of Business, Innovation & Employment's Research Infrastructure programme. https://www.nesi.org.nz.