Overcoming Catastrophic Forgetting in Continual Learning using Kolmogorov-Arnold Networks

Harris Hadjiantonis

Master of Science
School of Informatics
University of Edinburgh
2024

Abstract

Traditional Deep Neural Networks have been optimised to perform well in scenarios where they are given a single task to complete and have access to the entirety of the dataset describing the task. However, in continual learning scenarios, where new tasks arrive sequentially, these models often suffer from catastrophic forgetting (a phenomenon where models lose knowledge obtained in previous tasks as they try to adapt to new ones). To address this issue, many strategies have been proposed in the literature such as regularisation, rehearsal and structural adaptation of the models, however no perfect solution has been able to solve catastrophic forgetting. The novel neural structure of Kolmogorov Arnold Networks (KANs) recently proposed by Liu et al. [1] has shown promising potential for avoiding catastrophic forgetting. In this study, we provide the first complete and rigorous evaluation of KANs performance on classic continual learning benchmarks for a range of scenarios of increasing difficulty. We empirically show that replacing MLPs with KANs reduces the amount of parameters required (by a constant factor) for completing the same task and improves the performance of the model for simple benchmarks. However the current implementation experiences significant forgetting as the datasets become more irregular and complex, thus requiring the assistance of rehearsal and regularisation strategies to mitigate the issue.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Harris Hadjiantonis*)

Table of Contents

Chapter 1

Introduction

1.1 Motivation

The traditional machine learning paradigm often relies on the assumption that the data which we attempt to model are produced by a fixed distribution. However, the majority of real-world systems evolve over time thus causing conventionally trained models to rapidly lose their relevance and accuracy since this evolution shifts the original data distribution. To address this issue, a common practice is to frequently fine-tune and re-train the models using the updated datasets, a procedure that is both time-consuming and computationally expensive, especially when considering the size and complexity of the deep networks architectures employed in modern systems.

A better approach that allows models to maintain high levels of autonomy and efficiency without the need for extensive re-training is Lifelong or Continual Learning (CL) [2]. Systems trained in a CL manner are able to take a continuous stream of data generated by a particular distribution, extract useful information necessary for modeling these data and still be able to model shifted distributions throughout their lifetime. This approach is very suitable in domains characterized by non-stationary environments, where the data distribution evolves faster than the time required to re-train the model using the entire accumulated dataset. A great example of such domain is the financial market where stock valuations constantly fluctuate based on a multitude of economic, political, and other social events. Since constant refinement is necessary for a stock prediction system to capture these evolving dynamics [3] [4], yet costly to the organisation relying on the model's prediction for their daily trades, models trained in a CL manner could efficiently identify these shifts and adjust their predictions accordingly.

Despite its potential, Continual Learning faces a major challenge known as catastrophic forgetting [5]. This phenomenon occurs when the model attempts to learn new information, and in the process of improving its predictions for the current task, it loses the previously acquired knowledge. Without the ability to retain past information, the system is focused on improving its present state and becomes oblivious to the previous model of the world. Therefore overcoming catastrophic forgetting is of paramount importance if we want continual learning systems that are able of long-term adaptation.

1.2 Scope and Contributions

The aim of this study was to explore and rigorously examine the effectiveness of Kolmogorov-Arnold Networks (KANs), a novel architecture introduced by Liu et al. [1], for addressing the challenge of catastrophic forgetting in continual learning. Liu suggested that KANs could integrate new information without erasing previously learned knowledge because of the difference in the way the computed loss is backprobagated through the network and the increased control this architecture provides. However these conclusions were drawn primarily from limited experiments on simple, toy-like datasets, leaving substantial questions about the broader applicability of KANs in more complex scenarios. To examine the suitability of this proposal, in this study we conducted a comprehensive and rigorous evaluation of KANs, comparing their performance against state-of-the-art continual learning architectures across a range of datasets and incremental learning tasks of increasing difficulty.

The preliminary results we obtained when testing KANs' abilities to battle catastrophic forgetting on simple, and artificially generated datasets suggested that this architecture could be a promising solution to the problem of catastrophic forgetting. By leveraging their unique architecture which allows for localized learning through grid-based partitioning and spline-based function approximation, KANs were able to maintain high accuracy across sequential tasks with minimal forgetting compared to the traditional architectures, which often exhibited significant performance degradation as new tasks were introduced.

Despite the KANs' ability to retain information on synthetic datasets, when tested on larger and more complex vision benchmarks they didn't show any significant resistance to forgetting. To overcome this issue, we utilised the existing Convolutional Neural Network architecture (CNN) and replaced the standard Multilayer-Perceptron (MLP) that was respondible for the function approximation with the KAN counterpart. This

integration produced an improved model that combines the feature extraction abilities of CNNs with the information retention ability of KANs.

After combining the modified architecture with rehearsal and regularisation strategies, and testing its performance on the Split-MNIST dataset, we observed a 14% improvement in accuracy of the model and 33% reduction in forgetting however for the more complex benchmark of class-inremental-CIFAR-100 there was little to no improvement the accuracy or forgetting of the network but 3 times less parameters were required. We attribute the limitations of this architecture to the naive integration of the two models that is unable to efficiently remember a larger set of features of more irregular images and conclude this study with recommendations for future directions that may offer promising solutions to these challenges.

The remainder of this paper is organized as follows: Chapter 2 conducts a review into related literature on continual learning and catastrophic forgetting, providing background on the key concepts utilized in this research. Chapter 3 introduces the Kolmogorov-Arnold Network in greater detail and explains the core principles underlying our approach. Chapter 4 outlines the methodology we used to test and evaluate the performance of KANs. Chapter 5 describes the experiments conducted in depth and presents the results we obtained. Chapter 6 provides a comparative analysis between architectures and argues about the significance of the obtained results, and Chapter 7 concludes with a discussion of our findings, along with suggestions for future research.

1.3 Terminology

The following terms are used throughout the paper:

- **Task:** Refers to a unique objective the model is asked to perform (e.g. distinguish dogs from cats, recognise handwritten digits, etc).
- **Batch:** A collection of data given all-together to the model to handle.
- **Experience:** Refers to the current batch of data associated with a specific task. Different experiences might contain data from the same or different tasks but every experience contains data from a single unique task.
- **Stream:** There are three streams of data: 1) training, 2) validation, 3) testing. Data in the training and validation sets are used to fine-tune the hyperparameters of the model and the testing stream is used to evaluate the model.

Chapter 2

Background

2.1 Origins of Continual Learning

The concept of Continual Learning (CL) was firstly introduced by Grossberg [6] in an attempt to understand how the brain learns to continuously adapt to various environments with high efficiency and develop an understanding of the cognitive coding mechanism of humans. In his research, Grossberg concluded that that the brain is capable of continually rewiring its neural connections through a self-organising system that prolongs and amplifies signals in order to ensure that critical information is retained while new information is integrated. The main question his research was trying to answer was ”how the brain is able to form stable memories while still maintaining adaptive responses to new tasks” which was later formalised as the stability-plasticity dilemma [7].

Following the ideas introduced by Grossberg’s study and combining it with insights from biological reality, Feldman et al [8] expanded the idea of connectionist models - which had previously inspired the creation of artificial neural networks - and managed to describe the procedure of learning new tasks incrementally using a ”networks of units”. However, as pointed out later by Ratcliff et al. [9], early connectionist models [10] were significantly challenged by catastrophic forgetting [11] since they would rapidly forget previously learned information and they would face difficulty in discriminating between new and already studied items. Ratcliff’s evaluation set the standard for testing the resilience of models to catastrophic forgetting by suggesting a systematically variation of the complexity of tasks and the frequency of each task’s presentation. When tested early connectionist models, this evaluation analysis highlighted their significant limitations and prompted the development of new strategies to mitigate forgetting.

2.2 Battling Catastrophic Forgetting

Since the early 1990's many papers in the literature have been experimenting with a range of different approaches for designing and training models that can battle catastrophic forgetting which can be roughly categorised into three groups as follows:

1. **Regularisation-based methods:** Methods that penalise changes of the learnt parameters to retain knowledge acquired previously
2. **Rehearsal-based methods:** Methods that use a buffer of experiences which they use to retrain the model using examples from previous tasks
3. **Architectural-based methods:** Methods that dynamically alter the structure of the model's architecture to accommodate for different objectives

2.2.1 Regularisation-based methods

The main cause of catastrophic forgetting is the quick adjustment of a model's parameter (i.e. its weights) so that it performs better to new objectives. As a result, the learnt values of the previously important parameters become obsolete and overridden by new values. To discourage this quick adaptation, regularisation-based methods like Elastic Weight Consolidation (EWC) [12] apply a penalisation fee to the change of the important parameters that are considered critical for solving earlier tasks in an attempt to reduce forgetting. Specifically, EWC uses the Fisher Information Matrix [13] for determining the importance of each parameter on the previous task and applies a penalty of proportional magnitude if the model decides to change that parameter. The core idea of this method is that parameters with higher importance (as indicated by the Fisher Information Matrix) should not change much when learning new tasks, because they are critical for the success of that task and changing them leads to forgetting.

Another regularisation-based approach developed around the same time is Learning Without Forgetting (LWF) [14]. This method allows the model to train on the first task and then use the output of the learnt representations to penalise any deviations to these distributions caused by any parameter adaptation while transitioning to new tasks. This loss of knowledge (distillation loss) is calculated using a measure of difference between distributions - the Kullback-Leibler (KL) divergence [15]. Then the total loss for a parameter change is multiplied by a temperature-like factor that controls the trade-off between the learning of the new task and the retention of knowledge from the old task

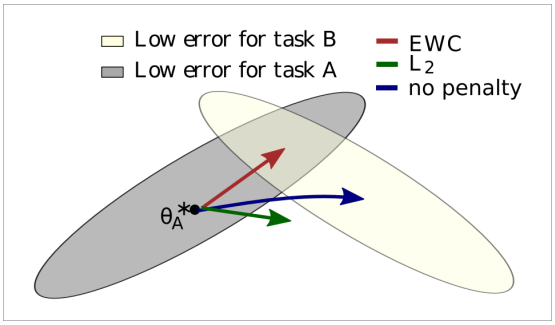


Figure 2.1: Figure from [12] showing the shift of the parameters of a network with EWC (red arrow) and without any regularisation (blue arrow) indicating how the intended shift in the parameter space should lead to the intersection of parameter space that are shared by both tasks (under the fundamental assumption that incremental tasks share representations embedded in the parameters)

and added to the task loss to compute the final LWF-loss which eventually reduces forgetting. However regularisation methods struggle with scalability as the number of tasks increases and therefore these methods do not guarantee the existence of a right regularisation balance.

2.2.2 Rehearsal-based methods

Rehearsal-based methods take a different approach than trying to preserve the stability of critical parameters by retrain the model on a combination of new an previously learnt data. The simplest form of rehearsal is explicit rehearsal where a dedicated buffer space is allocated in memory to store a subset of past experiences to be used again later to remind the model how it used to behave towards them in the past. However the major drawback with this approach is the increasing memory requirement which makes this method impractical for many real-life applications. A more efficient alternative to explicit is sample-based rehearsal. Methods such as Incremental Classifier and Representation Learning (iCaRL) [16] maintain a fixed size buffer which is constantly updated with new representative samples from each previous task - an idea originally proposed by the experience-replay paper [17].

Since the efficiency of rehearsal methods depends on the selection of the most representative samples, a heavy reliance on the chosen heuristic approach (e.g. herding [16]) is not ideal and could rather hinder the performance of the model with each replay. A more reliable alternative is the generative replay method which includes proposals such as deep generative replay [18] which trains a generative model to be able to

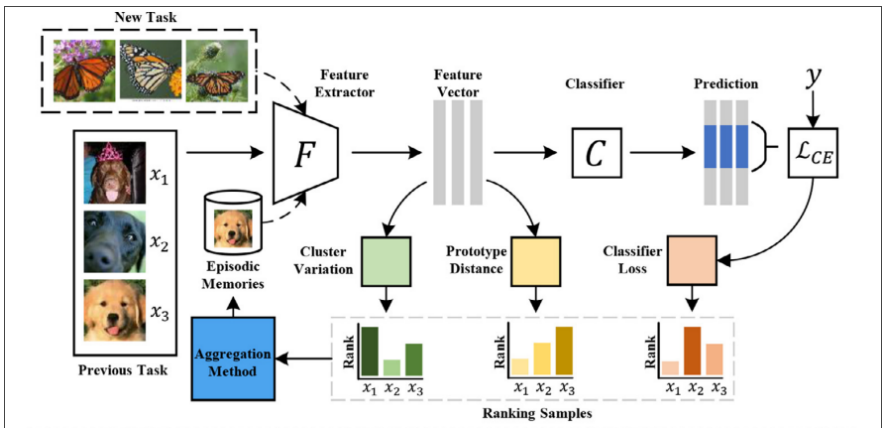


Figure 2.2: Heuristic evaluation (herding) for sample selection proposed by [19]. (Explanation: The figure shows the pipeline for handling samples from new tasks entering the network. Steps: 1) Features extracted from the new dataset and combined with the features saved in the episodic memory 2) The corresponding feature distributions encoding the different classes are updated and aggregated using a selected aggregation method 3) Classifier makes the prediction 4) Loss is backprobagated through the network.)

produce synthetic data that mimic the distribution of the original training dataset. This method significantly reduces the memory usage however the quality of the produced samples depends on the how well the generative has been train which is a significant challenge for certain datasets.

2.2.3 Structural-based methods

Following Grossberg's work [6] one of the first attempts to battle catastrophic forgetting was a structural-based approach that was mimicking brain's rewiring process. Grossberg and Carpenter collaborated in [20] and [21] and formalised Adaptive Resonance Theory in an investigation of how structural changes detected in the brain could inspire autonomous re-arranging of artificial neural networks' structure to allow for task adaptation. These studies primarily focused on brain's ability to perform object identification and recognition because of the better interpretability of visual results and this trend still appears in most of continual learning evaluation.

In recent years, working proposals like Progressive Neural Networks (PNNs) [22] managed to extend the network architecture by adding new sub-networks for each task while keeping the parameters of previous sub-networks fixed and thus preserve knowledge within them. The major drawback of this approach is the linear growth of the

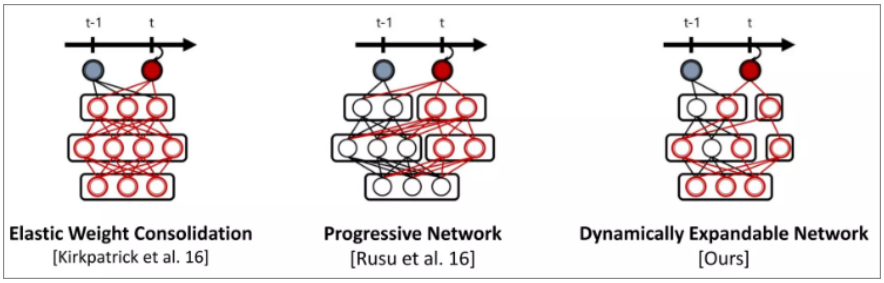


Figure 2.3: Comparison of EWC, Progressive Networks and Dynamically Expandable Networks as presented in [23]. (Explanation: The blue and black nodes along with the black edges represent the original network at time $t-1$ and the red nodes with the red edges represents the updated network at time t)

parameters with each new task. At a later proposal, Dynamic Expansion Networks [23] suggested selectively expanding the network by adding neurons or layers when needed while pruning unnecessary components in order to reduce excessive memory usage. Yet, despite the benefits of these methods, they usually require extensive computational resources and are not practical for real-time or resource-constrained environments.

2.3 Stability-Plasticity dilemma and the solution of KAN

The main challenge faced by almost all of the above solutions is finding the right balance of knowledge retention and adaptation to new tasks which is described as the stability-plasticity dilemma. Kolmogorov-Arnold Networks theoretically can avoid dealing with this dilemma because of the alternative parameter representation as a set of continuous univariate activation functions (see B-splines 3.3). This architectural choice removes the need of making a trade-off between stability and plasticity when choosing which memories to store/disregard or which components to update because the grid structure of these functions (if large enough) can be partition to physically split the predictions in different sub-ranges. This idea is further discussed in 6.1.

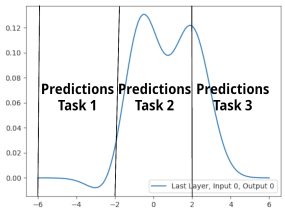


Figure 2.4: Physical prediction partition for an activation function

Chapter 3

Kolmogorov-Arnold Networks

3.1 Kolmogorov-Arnold Theorem

The fundamental idea behind KANs is the Kolmogorov-Arnold Representation Theorem which states:

Theorem 1. *Every multivariate continuous function $f : [0,1]^n \to \mathbb{R}$ can be written as a finite composition of continuous functions of a single variable and the binary operation of addition. [24]*

$$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right) \tag{3.1}$$

where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$

This theorem provides an alternative paradigm to the widely used Universal Approximation Theorem used by conventional MLPs (see Appendix A.1). Instead of providing an approximation of the function however, the above theorem provides a theoretical guarantee that any problem that can be modeled by a continuous, multivariate function can therefore be exactly determined by summing univariate functions. Thus the original task of modeling the any multivariate function reduces to the problem of finding the set of appropriate univariate functions that can be sum together to produce it.

The main issue with this theorem is that there are no guarantees that the univariate functions are smooth everywhere in their domain, which would make automatic differentiation impossible and therefore make any optimisation algorithm unable to improve and learn such a function. However Liu et al [1] empirically showed that in practice these functions are smooth and continuous. The breakthrough in the implementation of KANs occurred when the architecture was expanded beyond the initial two-layered

structure (used in KAN theorem 1) to arbitrary widths and depths, to produce a deeper KAN of univariate functions connected with the addition operator as shown in figure 3.1.

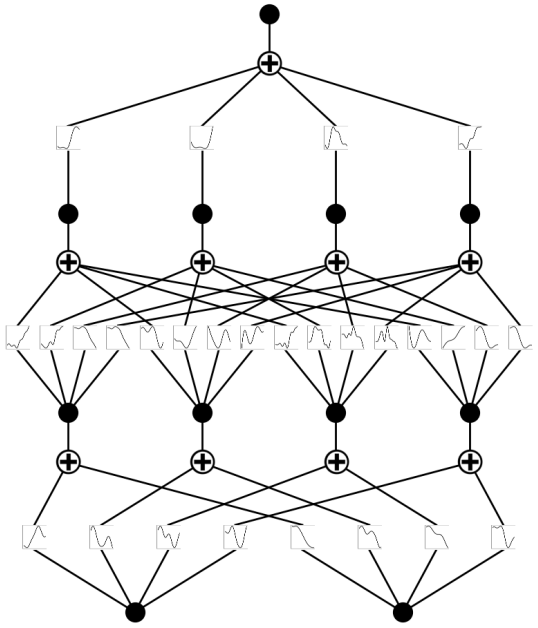


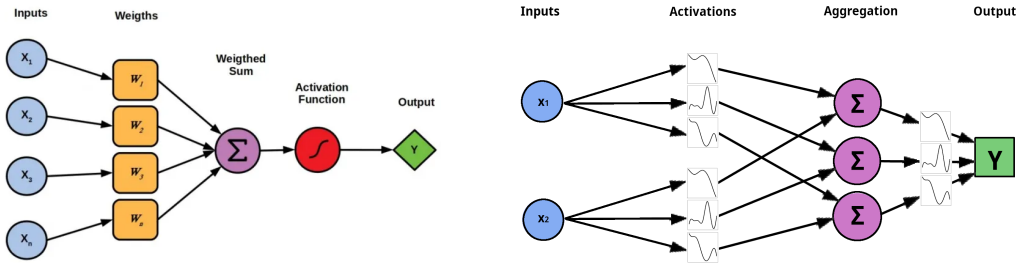
Figure 3.1: Example of a trained Kolmogorov-Arnold Network of a 2-layered architecture

3.2 Comparison to MLPs

A traditional Multi-Layered Perceptron can approximate any non-linear function by learning a set of fixed parameters (the weights of the network) that encode the importance of a connection in a complex network of nodes. Mimicking the biological neural spiking activity, each node in the network represents a neuron and performs a few very simple operations: a) collect the signals coming into the neuron through its connections with other neurons b) boost the signal based on the strength/importance of the connection c) determine if the sum of the incoming signals is above a threshold and if so inform all of the connected neurons that are connected to you. This procedure can be simulated by an artificial neural network as follows: Each neuron collects inputs from connected neurons, weights them, and passes the result through an activation function to also allow for non-linear functions to be learnt. If the activation exceeds a threshold, the neuron ”fires” and sends its output to the next layer.

A Kolmogorov-Arnold Network follows a slightly different approach by replacing the fixed weights of traditional neural networks with learnable activation functions. Instead of adjusting weights for determining the importance of connections between

neurons, this architecture learns a much more complex activation function that directly transform the inputs, allowing it to approximate complex functions without relying on weighted connections. This idea only works because of theorem 1 where the activation functions correspond to the univariate functions mentioned above and the connections between the nodes of the network are used to indicate which pair of processed input is to be summed together to obtain the output given to the next layer as input. Figure 3.2 shows a simplified version of the data propagation in each architecture.



(a) MLP Forward Pass (b) KAN Forward Pass

Figure 3.2: Shows the difference in the forward propagation of data in MLPs and KANs

After each network has finished propagating the information in the forward direction as shown in figure 3.2, it outputs the result of the last layer as its final prediction for the given data sample. This prediction is then compared with the actual value the network should have produced with the help of an appropriate loss function (e.g. mean squared error for regression tasks and categorical cross-entropy for classification tasks) and the error of the network is computed. This error is finally fed back to the network in order to adjust its parameters and improve its predictions. This adjustment in the case of MLPs is done by increasing or decreasing the value of each weight in the direction that reduces the loss (as indicated by the derivative of the loss function). On the other hand, KANs directly adjust the equation of each activation function by changing the position of its control points which removes the need of adjusting volatile weight values.

This difference between the two networks makes KANs an interesting architecture to study for continual learning scenarios since there is an extra degree of freedom (i.e. directly controlling the output of each activation function) which can be exploited if an appropriate family of equations is chosen. The idea here is to choose a family of equations that allows the function to be adjusted locally in the specific subdomain from which the error has occurred and thus leaving the rest of the function unperturbed. In fact, the original KAN paper [1] already suggests the use of B-Spline activation functions which possesses this locality property and can reduce catastrophic forgetting.

3.3 B-Splines

A Basis-Spline (B-Spline) function is a piecewise-defined polynomial function of degree d that is uniquely determined by a sequence of n control points ($\{P_0,\dots,P_n\}$) and $n+d+1$ points called knots ($\{t_0,\dots,t_{n+d+1}\}$) that partition the domain into $n+d$ distinct intervals. Formally a B-Spline is defined as follows:

Definition 1. *Let curve $S:[a,b]\to\mathbb{R}$ be defined in the closed interval $[a,b]$ covered by $n+d$ ordered disjoint sub-intervals as such:*

$$\begin{aligned}[a,b] &= [t_0,t_1)\cup[t_1,t2)\cup\ldots\cup[t_{n+d-1},t_{n+d})\cup[t_{n+d}]\\ a=t_0\le t_1\le\ldots\le t_{n+d-1}&\le t_{n+d}=b\end{aligned} \tag{3.2}$$

Then the curve is a B-Spline function if it can be defined as a linear combination of n basis functions $N_{i,d}(t)$ defined $\forall t\in[a,b]$, given a set of control points P_i as such:

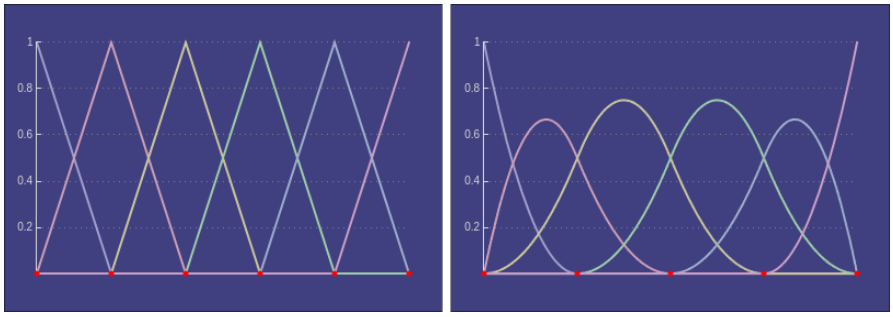
$$S(t)=\sum_{i=0}^{n-1}N_{i,d}(t)\cdot P_i \tag{3.3}$$

where $N_{i,d}(t)$:= polynomial of degree d defined recursively using Cox-de Boor formula:

$$N_{i,0}(t)=\begin{cases}1 & \text{if } t_i\le t<t_{i+1},\\ 0 & \text{otherwise},\end{cases} \tag{3.4}$$

$$N_{i,d}(t)=\frac{t-t_i}{t_{i+d}-t_i}N_{i,d-1}(t)+\frac{t_{i+d+1}-t}{t_{i+d+1}-t_{i+1}}N_{i+1,d-1}(t)$$

and $\{P_0,\dots,P_n\}$ the set of control points that influence the shape of the curve



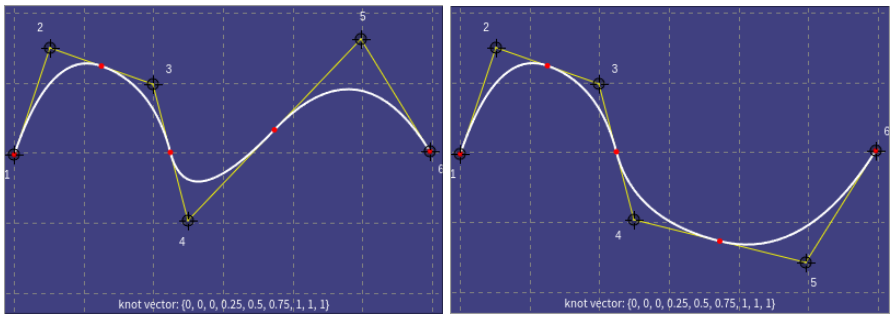
(a) Basis functions $N_{i,1}(t)$ ($d=1$) (b) Basis functions $N_{i,2}(t)$ ($d=2$)

Figure 3.3: Displays the basis functions for degree $d=1$ and $d=2$ defined over interval $[a,b]$. The knots are shown as red dots and are chosen to be equidistant

B-Splines are continuous and differentiable everywhere in their domain $[a,b]$ which are the two necessary properties required for any function to serve as an activation function. This is crucial because the backpropagation algorithm needs to be able to compute the derivative of the activation function and make the necessary adjustments while training the network.

In addition to these properties, B-Splines have the locality property which allows the learning algorithm to adjust the activation only in the interval around the area of the error. As shown in figure 3.4a the initial B-spline curve is made of 6 control points $(P_0,\ldots,P_5)$ which are linearly combined with the 6 basis functions from 3.3b. Then on figure 3.4b, the 5-th control point (P_4) is adjusted by being pulled downwards. This change is shown to have affected the shape of the curve only within the area from knot t_2 to knot t_4 (red dots) and has left the rest of the curve unperturbed.

This is the key property that has motivated this study; assuming that it is possible to roughly guess the number of task the model will experience (say e experiences) then by partitioning the domain of each activation function of the KAN network in at least $e \times d$ intervals would allow the network to separate its domain to memorise its behaviour and store it in that sub-interval (”experience dedicated grid”). Thankfully, KANs require significantly less parameters to express a function compared to traditional MLPs and therefore this solution will not require an excessive amount of extra memory to store the different behaviours of the network. In this way, different behaviours remain physically separated in different girds and the locality property of B-Splines guarantees that any adjustments in one grid will not affect the other, thus catastrophic forgetting should be significantly reduced.



(a) Initial B-Spline with 6 control points (b) Adjusted B-Spline

Figure 3.4: Shows the effect of adjusting the 5-th control point (black dot with the number 5) and exemplifies the locality property of B-splines

3.4 Improved KAN implementations

The original KAN paper [1] focused on the ability of Kolmogorov-Arnold Networks to accurate model symbolic functions (e.g. sine, cosine, exponent) and the combination of these functions into more complex formulas (e.g. $\sqrt{1+x^2-2xcos(x)-3e^2}$). However later studies attempted to test the performance of KANs on real world data. Azam et al [25] in a proprietary study showed that KANs can achieve comparable performance to MLPs in the vision domain with a significantly less amount of parameters required but it was only tested on small datasets because of some notable shortcomings of the original implementation. Since then, many improvements have been suggested which allow for computation speedup and GPU utilisation thus enabling faster and easier scaling. Because continual learning evaluation is customarily evaluated on vision datasets, the emphasis of this study was to find an efficient implementation of KANs that could handle large vision datasets.

While a few implementations have been developed that allow the architecture to be scaled and improved, these usually attempt to replace the recursive Cox-de Boor computation of splines (see 3.4) which removes the locality property observed in splines. Implementations such as FastKAN [26] which utilises an approximation theorem that enables radial basis functions to approximate any B-Spline efficiently, allow the network to handle large datasets and therefore scale better but do not provide a useful implementation of the architecture that can be used in continual learning settings. To overcome this issue, we decided to use a modified PyTorch implementation of the original paper as introduced by [27] since (after much experimentation) it demonstrated the best performance compared to the other implementations in terms of reduced forgetting.

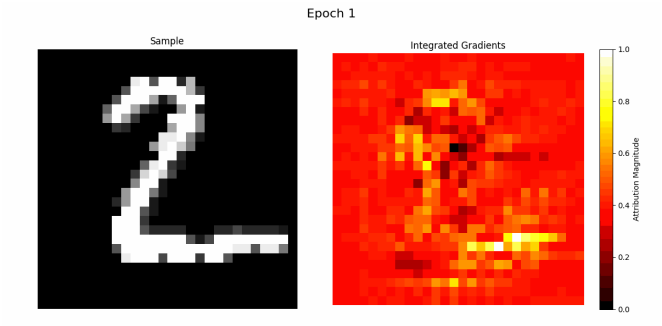


Figure 3.5: Image of TorchKAN implementation performing gradient analysis on MNIST dataset [27]

Chapter 4

Methodology

To properly assess the real advantage of KANs and measure their potential to battle catastrophic forgetting we need to perform a series of tests against a number of different learning scenarios and benchmarks of increasing complexity. This chapters provides a detailed description of the scenarios used for setting up the environment of each experiment, describes the benchmark choices and defines the metrics used to measure the difference between the two architectures.

4.1 Incremental Learning

The main difference between traditional batch learning and continual learning is the way data are introduced to the model. Unlike traditional batch learning where all data are given to the model in batches, data in continual learning are provided as they become available, requiring the model to update its parameters incrementally. There are three ways in which data might be introduced to the model: a) when the domain is explored incrementally (domain incremental) b) when new classes appear over time (class incremental) c) when the task shifts and there is a new objective that the model needs to achieve (task incremental)

1. **Domain Incremental:** In this training scenario there is a fixed number of tasks and classes conceptually but the underlying distribution of the input data (the different domains) shifts. In the example shown on figure 4.1a, the model is originally trained to distinguish between sketches of dogs and cats then the input domain changes to pictures of dogs and cats and finally the last domain contains synthetic 3d models of dogs and cats. At every stage there are two classes (cats and dogs) and the same task (distinguish between the two).

(a) Domain shift example (b) Class shift example (c) Task shift example

Figure 4.1: This figure provides examples for the three different incremental learning scenarios used for comparing the performance of KANs against MLPs

2. **Class Incremental:** This incremental strategy describes the scenario where new classes are introduced over time. An example of this scenario is shown in figure 4.1b where the model is initially trained to recognise images of just cats then images of cats and dogs and finally images of cats, dogs and birds. All images are pictures of these animals in nature thus the domain is the same and the task hasn't changed (to distinguish between the different animals)

3. **Task Incremental:** The last scenario involves incrementally changing the objective the model needs to complete while keeping the domain and the classes the same. A building blocks game is a great example where this scenario can be applied: given a fixed set of blocks, at every increment the model is required to assemble the building blocks in different shapes at every new task as shown in figure 4.1c.

4.2 Benchmarks

There is a broad range of benchmarks used for assessing the capabilities of CL models and therefore there is no consensus as of to which ones yield the most reliable and realistic results. Yet, it is common practice to use computer vision datasets for benchmarking because of the higher level of interpretability of the results without having to sacrifice tasks complexity. This study provides a mixture of custom toy datasets used as a proof of concept and classic vision datasets of increasing complexity and size (from MNIST to CIFAR100).

However the above datasets still don't provide an accurate representation of the

true performance of the model because of the artificial and temporal evolution of the data increments (i.e. the way in which successive data are associated) and the limited amount of samples available per increment. These datasets - while providing a very useful indication of the performance of the model - they still introduce abrupt changes which are undesirable for testing the true ability of the model to battle catastrophic forgetting in the real world. The CLEAR benchmark [28] was develop with the focus to provide a rich, labeled, high-quality data of real-world images that allows for the most realistic evaluation of the model while establishing a streaming protocol for deploying any CL model.

In this study we decided to test our models only against the MNIST and CIFAR100 benchmarks which are still suitable for providing a fair comparison between the different networks since we will be using the same environment settings for all experiments. Still, the study was heavily influenced by the training and evaluation protocols introduced by CLEAR and [29] which allowed for a robust and unbiased comparison to be performed.

4.3 Metrics and Evaluation

By simply comparing the accuracies of the best-performing models for each architecture we are disregarding any other important advantages a model can offer such as reduced parameter count and less forgetting. There is a plethora of important factors that can be consider for a fair comparison between architectures, however in our study we decided to focused on: a) the ability of each model to learn and adapt to new tasks and b) the ability each model retain information from the past. To quantify the performance on each of the above factors, we have chosen the following metrics for evaluation:

1. **Average Accuracy:** Measures the average accuracy on the validation data after the model has been continually trained up to T tasks

$$AA_T = \frac{1}{T}\sum_{i=0}^{T} a_{T,i} \tag{4.1}$$

   where $a_{T,i}$ is the accuracy of the model after learning all tasks up to task T for the test dataset of task i

2. **Learning Accuracy:** Measures the accuracy of the model's performance on the current task's test dataset immediately after training on it and averaging over all

experiences (good indication of the plasticity of the model)

$$LA = \frac{1}{T}\sum_{i=0}^{T} a_{i,i} \tag{4.2}$$

where $a_{i,i}$ is the accuracy of the model on the dataset of the current task i that it has just been trained on.

3. **Average Forgetting:** Measures the extent to which the model forgets previous tasks as it learns new ones by comparing the best performance achieved on each task to the performance of the current task

$$F_T = \frac{1}{T-1}\sum_{i=1}^{T} -1\left(\max_{t\leq T-1} a_{t,i} - a_{T,i}\right) \tag{4.3}$$

These results are presented in the form of heatmaps and graphs in the experiment section thus providing empirical quantifiable data that can be used for a fair comparison of architectures.

4.4 Environment Setup

To conduct the experiments and evaluate the performance of the models under different continual learning scenarios, the Avalanche library was heavily utilised [30]. This PyTorch-based framework was designed specifically for continual learning research and has allowed for the experimentation process to be streamlined. Figure 4.2 shows the training and evaluation life-cycles for every experience $(e_1, \ldots, e_n)$ and how they interact with the different loggers and the benchmark modules provided by the library.

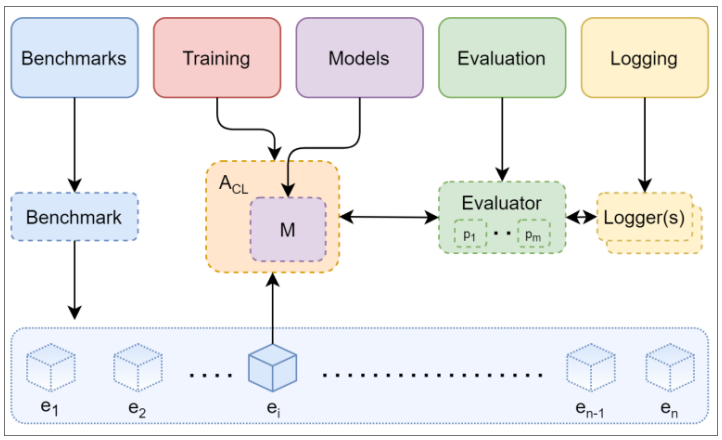


Figure 4.2: Avalanche modules and pipelines [30]

Chapter 5

Experiments and Results

5.1 Empirical evaluation of KANs performance on analytic functions

5.1.1 Mixture of Gaussians - 1D

The original hypothesis of this study (KANs can battle catastrophic forgetting) is based on the observation made by Liu et al [1] that when KANs are trained in a continual learning manner on an 1-dimensional function (a mixture of 5 Gaussian distributions) they are able to overcome catastrophic forgetting. To validate this observation, we attempted to replicate the paper and test the validity of the claim on a slightly more complex function. Let function $f(x) : [-1,1] \to \mathbb{R}$ be the mixture of 5 Gaussian distributions with equidistant centers $\{\mu_1 = -0.8, \ldots, \mu_5 = 0.8\}$ and standard deviations $\{\sigma_1 = \sigma, \ldots, \sigma_5 = \frac{\sigma}{5}\}$, where each individual function is defined as follows:

$$G(x;\mu,\sigma) = e^{-\frac{(x-\mu)^2}{2\sigma^3}} \tag{5.1}$$

then by summing the contribution of each distribution together, we obtain the following equation for the target function:

$$f(x) = \sum_{i=1}^{5} \exp\left(-\frac{(x-\mu_i)^2}{2\cdot\left(\frac{\sigma}{i}\right)^2}\right) \tag{5.2}$$

The above equation uses index i to identify each of the five Gaussian functions and to also control the width and minimum height of each peak in the mixture. This modification is made to ensure that each constituent distribution differs from the rest of the distributions in more than one ways (i.e. shifted mean and standard deviation) resulting in the function shown in 5.1.

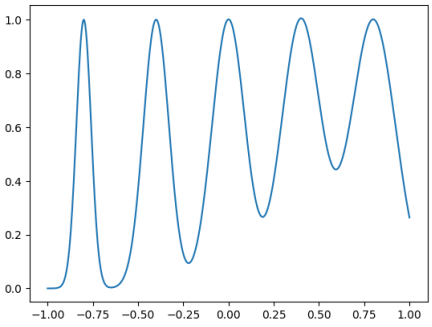


Figure 5.1: Mixture of 5 Gaussian distributions with equidistant centers and increasing variance as defined by equation 5.2

By partitioning the domain in 5 disjointed sub-sets and uniformly sampling from each of them, we obtain 5 experiences which are then split into training and testing streams with ratio 80:20. Each network accepts a single experience and attempts to learn the distribution that produced those data in that sub-domain. Then every network is asked to make predictions for inputs from the entire domain by sampling 1000 random samples from equation 5.2. After producing the predictions, a new experience of data is presented to the model and the same procedure is repeated again. The results of this process are shown in figure 5.2.

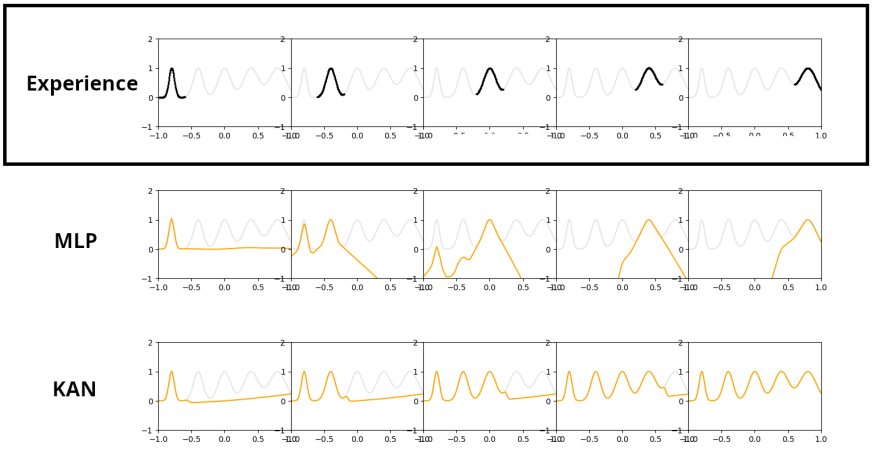


Figure 5.2: Performance difference between MLP and KAN architectures for simple univariate mixture of Gaussian function

Figure 1 clearly displays the difficulty experienced by MLPs to remember the distributions of previous experiences as they attempted to adapt to new ones. On the contrary, KANs experienced little to no forgetting for the above scenario and required a significantly less amount of parameters to achieve this result (as shown in table 5.1).

Model	Architecture	Grids/Order	Parameters	Optimiser	Learning Rate	Epochs
MLP	[1,256,1]	N/A	512	Adam	0.001	100
KAN	[1,1]	50/2	150	LBFGS	0.001	40

Table 5.1: Parameters used for each network (1D function)

Figure 5.3 shows the trained KAN architecture and the learnt spline function that closely follows the shape of the target function. This experiment validates the claim of the authors of [1] however it provides an unfair advantage to KANs since it directly exploits the locality property of the univariate spline function to model a smooth, univariate function such as the target function. By allowing KAN's splines to have 50 control points, and all polynomials to be of degree 3, we have essentially assigned every 10 control points to handle one experience and fix the shape of the spline while leaving the rest of control points unperturbed. This is the indented behaviour we wanted to achieve however this trick might not be possible for when modeling more complex functions that take more than one element as input.

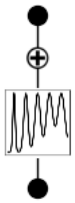


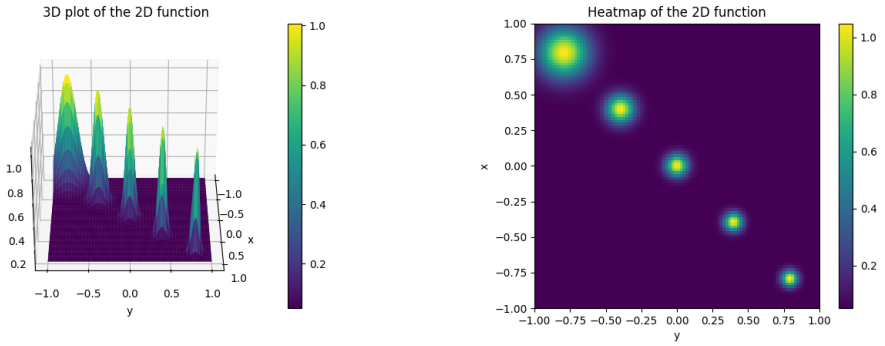
Figure 5.3: Plot of trained KAN for 1D function - domain incremental scenario

5.1.2 Mixture of Gaussians - 2D

To debunk the representation argument proposed by critics of the KAN architecture (i.e. that the locality of splines is benefitial only when handling univariate analytical functions), we device a slightly more complicated experiment that involves two variables. We follow a similar procedure as with our previous experiment but we extend the mixture of Gaussians to a 2D grid where the center of each distribution is located along the main diagonal of the input space (i.e. where $y = x$). Again we chose function $f(x,y)$ to be restricted in the domain $[-1,1] \times [-1,1]$ and to map the output to any real value in $\mathbf{R}$ with equation:

$$f(x,y) = \sum_{i=0}^{4} \left[\exp\left(-\left(\frac{(x-\mu_{x,i})^2}{2\sigma_{x,i}^2} + \frac{(y-\mu_{y,i})^2}{2\sigma_{y,i}^2} \right) \right) + 0.2 \right] \tag{5.3}$$

where $\mu_{x,i}$ and $\mu_{y,i}$ are the centers of the i-th Gaussian distribution, $\sigma_{x,i}$ and σ_{y_i} are the standard deviations, which change with each i.We have added an arbitrary 0.2 for better visualisation (to distinguish gaps from sampling and actual zero values)



(a) Three-dimensional representation of the 2D Mixture of Gaussians

(b) Heatmap of 2D Mixture of Gaussian

Figure 5.4: Representation of the target function $f(x)$ as defined by equation 5.3

Figure 5.4a displays the 3D plot of the two-dimensional function $f(x,y)$ as defined by equation 5.3 while figure 5.4b displays the values of the target function as a heatmap. In order to test the performance of each model in a domain incremental scenario, we need to partition the domain into distinct experiences, each containing data from a different subdomain of the input space. We chose to do that by splitting the input space along the y-axis thus creating five horizontal stripes from which we uniformly sample to create the training and testing data-streams with a ratio of 80:20 as shown in figure 5.5. Each stripe is then presented as a new experience to both models in an incremental manner for them to be trained on the sample data and learn the equation of th distribution that produced them. The results of this procedure are visually displayed in figures 5.7 and 5.6 respectively whereas the parameters of each model are shown in table 5.2.

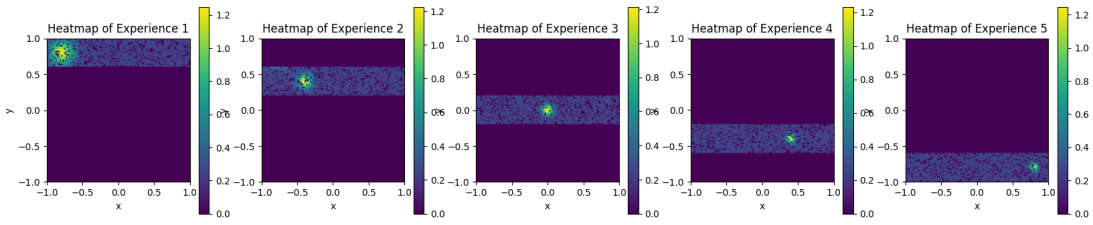


Figure 5.5: Samples given in incremental order for training

Model	Architecture	Grids/Order	Parameters	Optimiser	Learning Rate	Epochs
MLP	[64,512,32,1]	N/A	49184	Adam	0.001	100
KAN	[2,5,11,1]	40/2	6080	LBFGS	0.001	40

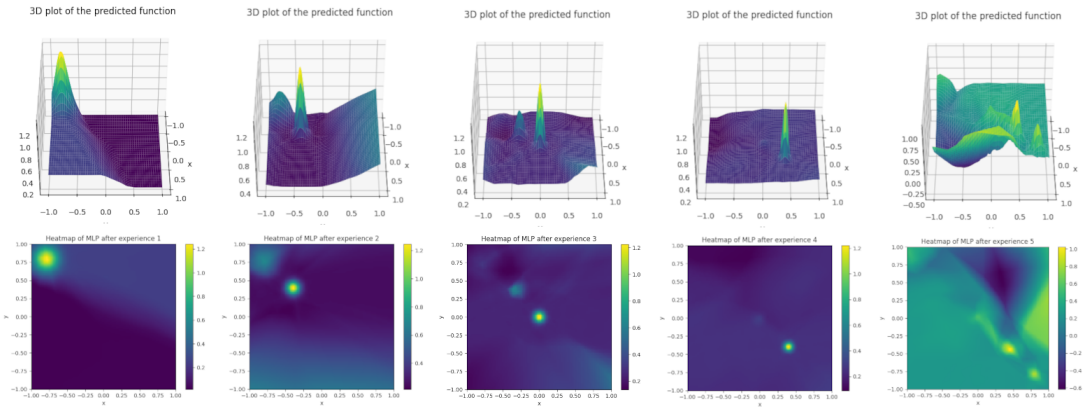


Figure 5.6: Predictions of MLP on Incremental Dataset

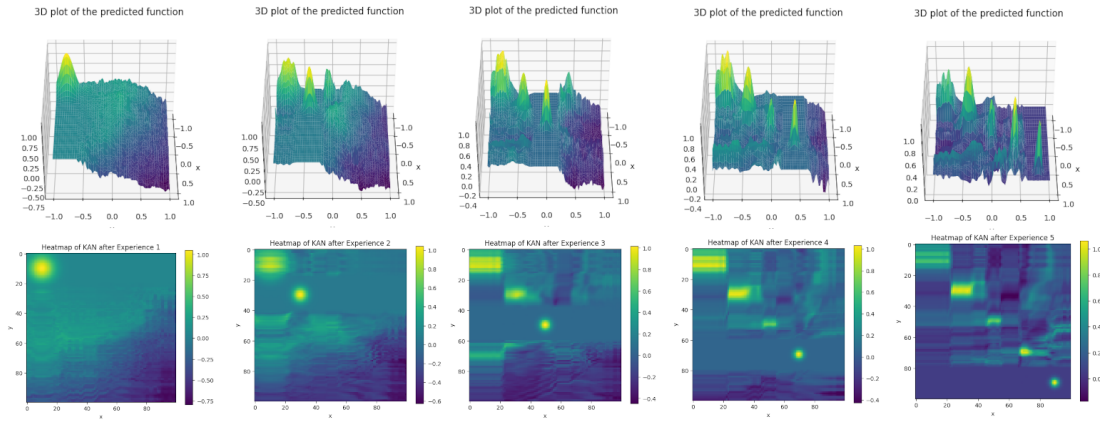


Figure 5.7: Predictions of KAN on Incremental Dataset

In this experiment, again, it is evident that KANs demonstrate a much stronger ability to retain and approximate the position and variance of each Gaussian peak with minimal forgetting. In contrast, MLPs tend to forget the existence of a peak for experiences before the penultimate.

It is important at this point to notice the rougher grid prediction produced by KANs compared to the smoother predictions of the MLP architecture. While increasing the amount of parameters in the architecture allows the network to learn more complex latent representations and thus produces smoother predictions, attempting to do so in the case of KANs makes them more prone to forgetting. This observation could indicate that the KANs perform lookup operations from a partitioned grid rather than developing

a smooth, continuous understanding of the data. This behaviour could be acceptable for a small artificial dataset if it results in less forgetting, however it defies to purpose of a developing a generic model, able to capture more abstract and generalizable features that are present in larger and more complex datasets.

5.2 Robust Benchmarking on Vision Datasets

To overcome the issue of feature representation learning experienced by KANs with small architectures, we replace the vanila KAN architecture with a mixture of Convolutional Neural Networks [31] that are able to extract the features of complicated dataset and pass them to KANs to perform the classification or regression task as depicted in figure 5.8:

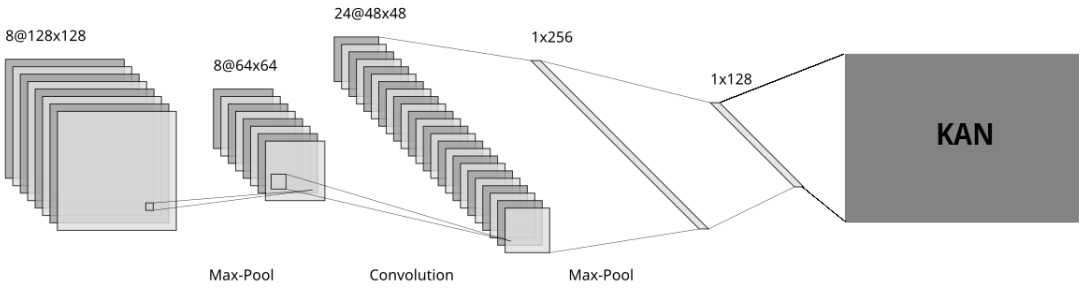


Figure 5.8: Convolutional KAN architecture (generated with alexlenail-svg)

5.2.1 MNIST Benchmark

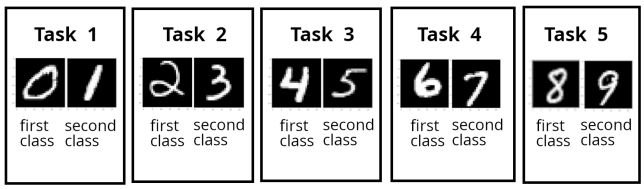


Figure 5.9: MNIST Task-Incremental (TI) Split

The Split-MNIST task is a widely used benchmark in continual learning for evaluating a model's ability to adapt to new tasks while retaining knowledge from previous tasks. In this task, the MNIST dataset, which consists of grayscale images of handwritten digits, is spitted into 5 different tasks each containing images from two different

classes as shown in figure 5.9. This setup enables us to tests the model's capacity to identify the features that make up the images of the digits and transfer that knowledge to following tasks quickly (measured by the Average Accuracy metric) while resisting to catastrophic forgetting as it encounters progressively more tasks (measured by the Average Forgetting metric).

For this experiment, we used the original MNIST dataset which was processed as follows:

- **Training set:** 60,000 images
- **Test set:** 10,000 images
- **Tasks:** 5 tasks as shown in figure 5.9
- **Input size:** 28x28 pixels (flattened to 784 dimensions)

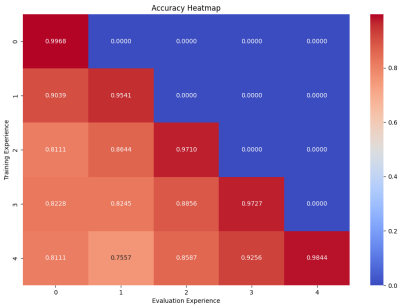
Each model was trained sequentially on these 5 tasks with the assistance of the generative replay strategy plugin - a rehearsal based method explained previously in 2.2.2 - and the performance was evaluated after each task to measure the level of forgetting. The accuracy was recorded on both the current task and all previously learned tasks. Table 5.3 outlines the architecture and hyperparameters used for the MLP and KAN models.

These results highlight the effectiveness of the Convolutional-KAN architecture in reducing catastrophic forgetting, when used along with a generative replay buffer. The network by itself experienced significant forgetting which could be attributed to the small grid size chosen for the architecture. This choice prevents splines from adequately partition the grid (the domain of each activation function) from capturing the set of features corresponding to classes from each task. We further discuss this observation is provided later in 6.1. on It is also important to note that smaller networks (KAN:[784,5,10] - 80K parameters - and [784,7,10] - 120K parameters) achieved comparable results to the conventional MLP which can be useful for scenarios where memory and time are limited thus freeing space for a larger replay buffer to be used which would further reduce forgetting.

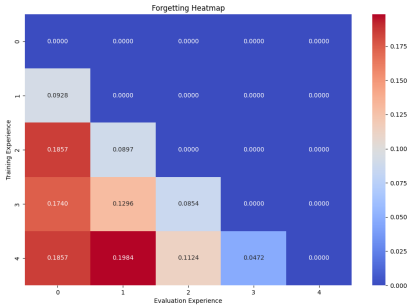
Model	Architecture	Grids/Order	Parameters	Optimizer	Learning Rate	Epochs
MLP	[784,1024,512,10]	N/A	160K	Adam	0.001	10
KAN	Conv:[784,32,64] KAN:[784,10]	10/2	150K	SGD	0.001	10

Table 5.3: Parameters used for each network on MNIST Permutations task

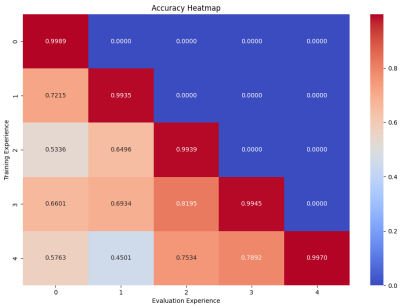
Figure 5.10 displays the Learning Accuracies and the Average Forgetting of both models at every increment. By choosing the amount of parameters to be approximately the same for both models and with the assistance of the generative replay architecture we observe that KANs exprience significantly less forgetting (**0.1359** VS 0.3529) while achieving a significantly higher Average Accuracy (**0.867** VS 0.713).



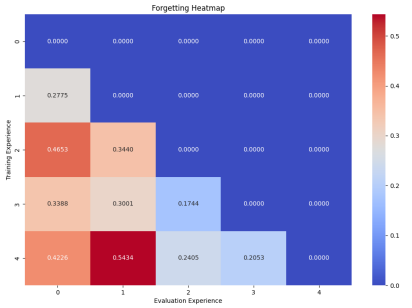
(a) Average Accuracy with Convolutional KAN model - Generative Replay (AA =**0.867**)



(b) Average Forgetting with Convolutional KAN model - Generative Replay (F =**0.1359**)



(c) Average Accuracy with conventional MLP - Generative Replay ($AA = 0.713$)



(d) Average forgetting with conventional MLP - Generative Replay ($F = 0.3529$)

Figure 5.10: Comparison of average accuracy and forgetting metrics of conventional and KAN-modified CNN architectures on Split-CIFAR-100 dataset for 20 experiences

5.2.2 CIFAR-10/100 Class Incremental Learning

To further challenge the robustness of KANs, we extended our evaluation to the include the CIFAR-100 dataset. Unlike the MNIST dataset, CIFAR-100 consists of colored images of real-world objects from 100 distinc classes. These datasets are commonly used in computer vision tasks, making them a more complex and realistic benchmark for continual learning.

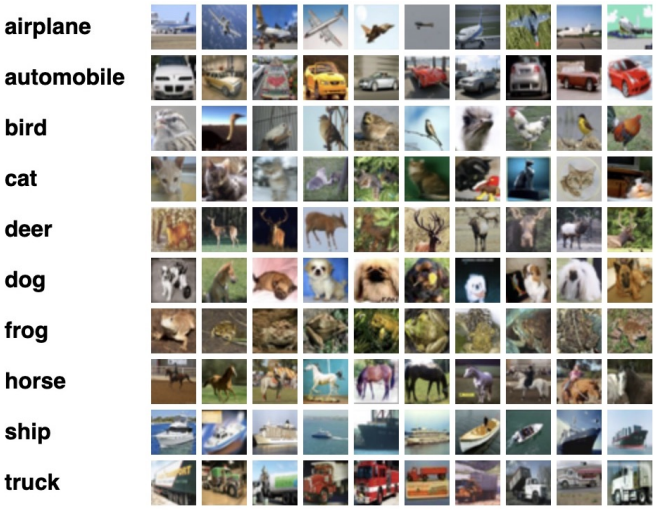


Figure 5.11: CIFAR-10 dataset

For this experiment, we used the CIFAR-100 dataset which was processed as follows:

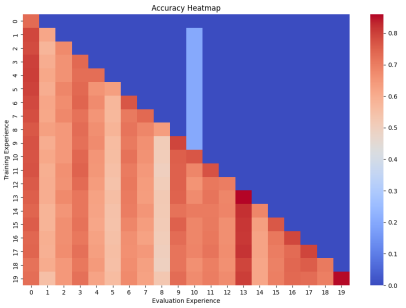
- **Training set:** 50,000 images
- **Test set:** 10,000 images
- **Tasks:** 20 tasks, each with 5 classes
- **Input size:** 32x32 pixels (flattened to 3072 dimensions)

We again contacted task incremental learning to train both networks by sequentially introducing new tasks to each of them. To do that, we divided the classes into 20 tasks, each containing 5 classes and with the help of an episodic rehearsal strategy we obtained the results shown in 5.12. The details of the model architectures and training parameters are provided in Table 5.4.

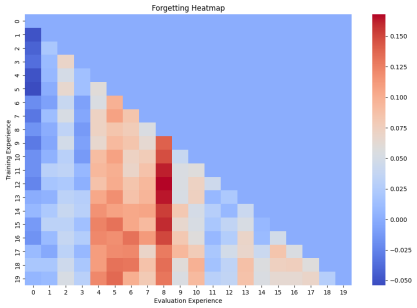
Model	Architecture	Grids/Order	Parameters	Optimizer	Learning Rate	Epochs
MLP	[3072,2048,1024,100]	N/A	9M	Adam	0.001	20
KAN	Conv:[3072,32,16] KAN:[10,20,20,10,100]	30/3	3M	SGD	0.001	20

Table 5.4: Parameters used for each network on CIFAR-100 tasks

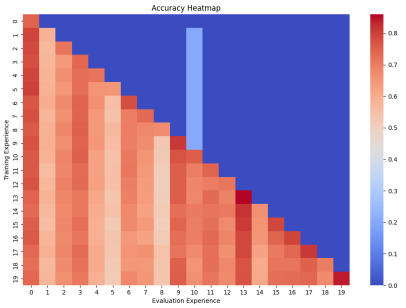
As illustrated in Figure 5.12, both models were able to achieve almost identical accuracies and experience the same amount of forgetting. However, Convolutional KANs used 3 times less parameters than MLPs and yet managed to experienced slightly less forgetting. The almost identical behaviour of these networks suggest a very strong



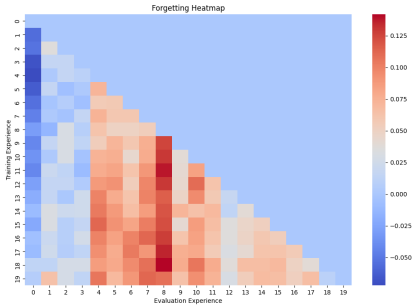
(a) Average Accuracy with Convolutional-KAN model - LAMAML Strategy ($AA = 0.679$)



(b) Average Forgetting with Convolutional-KAN model - LAMAML Strategy ($F =$**0.0619**)



(c) Average Accuracy with conventional MLP - LAMAML Strategy ($AA = 0.679$)



(d) Forgetting with conventional MLP - LAMAML Strategy ($F = 0.0655$)

Figure 5.12: Comparison of Average Accuracy and Forgetting metrics of MLP and Convolutional-KAN architectures on Split-CIFAR-100 dataset for 20 experiences using Generative Replay

influence of the limited amount of training samples compared to the range unique images presented per class. This result suggests that while KANs provide a more memory-efficient method for continual learning, they may require further refinement or to handle the increased complexity and diversity of data found in more challenging vision tasks.

Chapter 6

Discussion

6.1 Interpretation of Results

6.1.1 Univariate Function Approximation

Our initial experiments with the mixture of Gaussians in one dimension validated the claim by Liu et al. that KANs are effective at mitigating catastrophic forgetting. The results demonstrated that KANs could maintain their performance across multiple learning experiences, while MLPs showed significant performance degradation as new tasks were introduced. This is likely due to the localized nature of KANs, which allows them to compartmentalize the knowledge learned from each task.

6.1.2 Extension to Multivariate Functions

The extension of the Gaussian mixture to two dimensions further tested the robustness of KANs. While MLPs struggled to retain knowledge of previous tasks, KANs continued to show strong retention, albeit with a coarser grid that limited their ability to produce smooth function approximations. This finding suggests that while KANs are effective in simple scenarios, their performance might degrade as the complexity of the input space increases, particularly when the grid size is not sufficiently fine to capture the function's nuances.

6.1.3 Image Classification Tasks

When applied to more complex vision datasets like MNIST and CIFAR-100, KANs continued to outperform MLPs in terms of reducing catastrophic forgetting, especially when paired with generative replay strategies. However, the gap in performance highlighted a critical limitation: KANs, despite their advantages in retention, require careful tuning of grid sizes and order to balance between memory retention and the ability to generalize across tasks. Moreover, the relatively rough predictions observed in KANs suggest that they may be better suited for scenarios where high precision in function approximation is less critical, or where grid refinement can be feasibly increased.

Chapter 7

Conclusion & Future work

7.1 Conclusion

In this study, we have empirically shown that the original claim of the authors of KAN paper [1] holds only on special domains, but does not apply for generic datasets. However, the network's ability to achieve superior performance with a significantly less amount of parameters can be exploited by rehearsal based methods to increase the size of their buffer which will enable more samples to be stored in them. This could allow more samples to be revisited in the same amount of time it would have taken a conventional MLP to train to achieve the same result. Despite these findings, we discussed potential adaptations the KAN architecture could do to overcome the shortcomings of the current implementation. As the main contributions of this study we enlist the following:

- Provided empirical evidence that KANs are able to overcome catastrophic forgetting when incrementally trained on univariate and multivariate analytical functions
- Rigorously tested the ability of the current KAN architecture to battle catastrophic forgetting and concluded that KANs on their current form and alone are not adequate to do so.
- Subsequently, disproved the generic claim made by Liu et al in [1] by providing the example of a slightly more complex benchmark (CIFAR-100)
- Provided suggestions on how the current architecture can be modified to address some of the issues identified during this study.

7.2 Future Work

The results of this study suggest several avenues for future research. Bellow we enlist some of the most interesting that worth exploring:

- **Adaptive Grid Techniques:** One of the most interesting properties of KANs which was not explored during this study was grid adaptation. This is the ability of splines to further subdivide a single polynomial curved defined over an interval $[a, b]$ into a combination of splines without affecting the rest of the grid. Developing adaptive grid techniques that can dynamically adjust polynomials based on the complexity of the task could improve task isolation and therefore is worth exploring.

- **Understand relation of Network Hyperparameters and forgetting:** While testing and fine tuning the models, it was noted that the variable for the grid size and the amount of activation functions per layer for the network were directly affecting its ability to battle forgetting. Developing an understanding of how these are related could enable and guide further studies to efficiently choose their hyperparameters and therefore accelerate their testings.

- **Test for simpler real-world scenarios:** As mentioned in the experiments section, the main obstacle that prevented KANs from overcoming catastrophic forgetting is learning how to store the latent representations provided by the convolutional network. Since images are complicated structures are therefore are made out of multiple complex features, learning to efficiently map that many features into grids seemed difficult. However attempting to apply the same mapping for a simpler domain in the real world could be simpler and much more efficient with KANs.

Bibliography

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

[2] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024.

[3] Ranjan Kumar Dash, Tu N. Nguyen, Korhan Cengiz, and Aditi Sharma. Fine-tuned support vector regression model for stock predictions. *Neural Computing and Applications*, 35(32):23295–23309, 2021.

[4] Xiaoyu You, Mi Zhang, Daizong Ding, Fuli Feng, and Yuanmin Huang. Learning to learn the future: Modeling concept drifts in time series prediction. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 2434–2443, New York, NY, USA, 2021. Association for Computing Machinery.

[5] Ronald Kemker, Angelina Abitino, Marc McClure, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. *CoRR*, abs/1708.02072, 2017.

[6] Stephen Grossberg. *How Does a Brain Build a Cognitive Code?*, pages 1–52. Springer Netherlands, Dordrecht, 1982.

[7] Dongwan Kim and Bohyung Han. On the stability-plasticity dilemma of class-incremental learning, 2023.

[8] J.A. Feldman and D.H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6(3):205–254, 1982.

[9] Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychol. Rev.*, 97(2):285–308, 1990.

[10] Florence L. Goodenough. Edward lee thorndike: 1874-1949. *The American Journal of Psychology*, 63(2):291–301, 1950.

[11] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165, 1989.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017.

[13] R A Fisher. On the mathematical foundations of theoretical statistics. *Philos. Trans. R. Soc. Lond.*, 222(594-604):309–368, January 1922.

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017.

[15] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.

[16] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[17] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[18] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[19] Chen Zhuang, Shaoli Huang, Gong Cheng, and Jifeng Ning. Multi-criteria selection of rehearsal samples for continual learning. *Pattern Recognition*, 132:108907, 2022.

[20] G.A. Carpenter and S. Grossberg. The art of adaptive pattern recognition by a self-organizing neural network. *Computer*, 21(3):77–88, 1988.

[21] Gail Carpenter and Stephen Grossberg. Adaptive resonance theory. Technical report, Boston University Center for Adaptive Systems and Department of Cognitive Sciences, 1998.

[22] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[23] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks, 2018.

[24] Johannes Schmidt-Hieber. The kolmogorov-arnold representation theorem revisited, 2021.

[25] Basim Azam and Naveed Akhtar. Suitability of kans for computer vision: A preliminary investigation, 2024.

[26] Ziyao Li. Kolmogorov-arnold networks are radial basis function networks, 2024.

[27] Subhransu S. Bhattacharjee. Torchkan: Simplified kan model with variations. `https://github.com/1ssb/torchkan/`, 2024.

[28] Zhiqiu Lin, Jia Shi, Deepak Pathak, and Deva Ramanan. The clear benchmark: Continual learning on real-world imagery, 2022.

[29] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning, 2022.

[30] Antonio Carta, Lorenzo Pellegrini, Andrea Cossu, Hamed Hemati, and Vincenzo Lomonaco. Avalanche: A pytorch library for deep continual learning. *Journal of Machine Learning Research*, 24(363):1–6, 2023.

[31] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks, 2015.

[32] Kai Fong Ernest Chong. A closer look at the approximation capabilities of neural networks. In *International Conference on Learning Representations*, 2020.

[33] George V. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

Appendix A

Additional Material

A.1 Universal Approximation Theorem

Theorem 2. *Any continuous multivariate function* $f(\mathbf{x})$ *can be approximated up to a chosen approximation threshold* ε *given a set of weights* $\mathbf{w}$*, biases* $\mathbf{b}$ *and non-linear, non-polynomial activation functions* σ *as follows (see also [32] and [33]):*

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{i=1}^{N(\varepsilon)} \alpha_i \sigma(\mathbf{w_i} \cdot \mathbf{x} + b_i) \tag{A.1}$$

A.2 Comparison between MLP and KAN

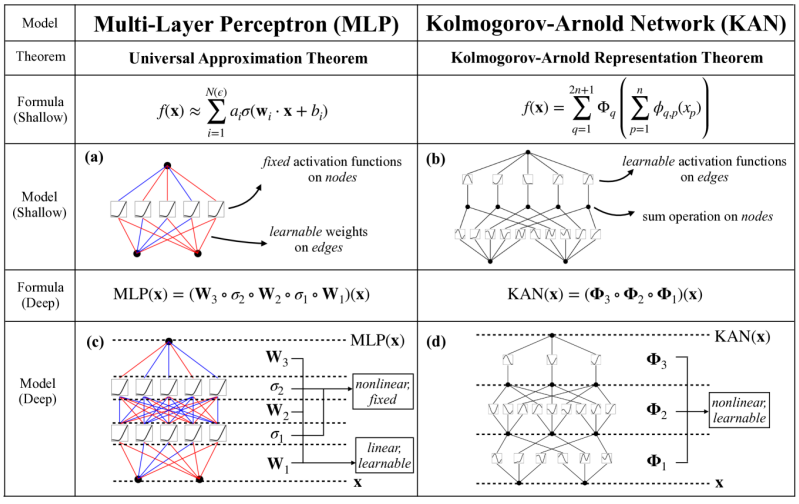


Figure A.1: Multi-Layer Perceptrons (MLPs) VS Kolmogorov-Arnold Networks (KANs) - Image from [1]

A.3 Trained KAN for the 2D function (domain-incremental)

Observing the last layer of activation functions after the end of training on each experience, it is clear that the shape of splines becomes more and more defined. After the network has seen all experiences, the last layer of splines contains a few curves that are similar to the projection of peaks along the x and y axis. However the structure of the network doesn't provide much intuition about the latent representations learnt.

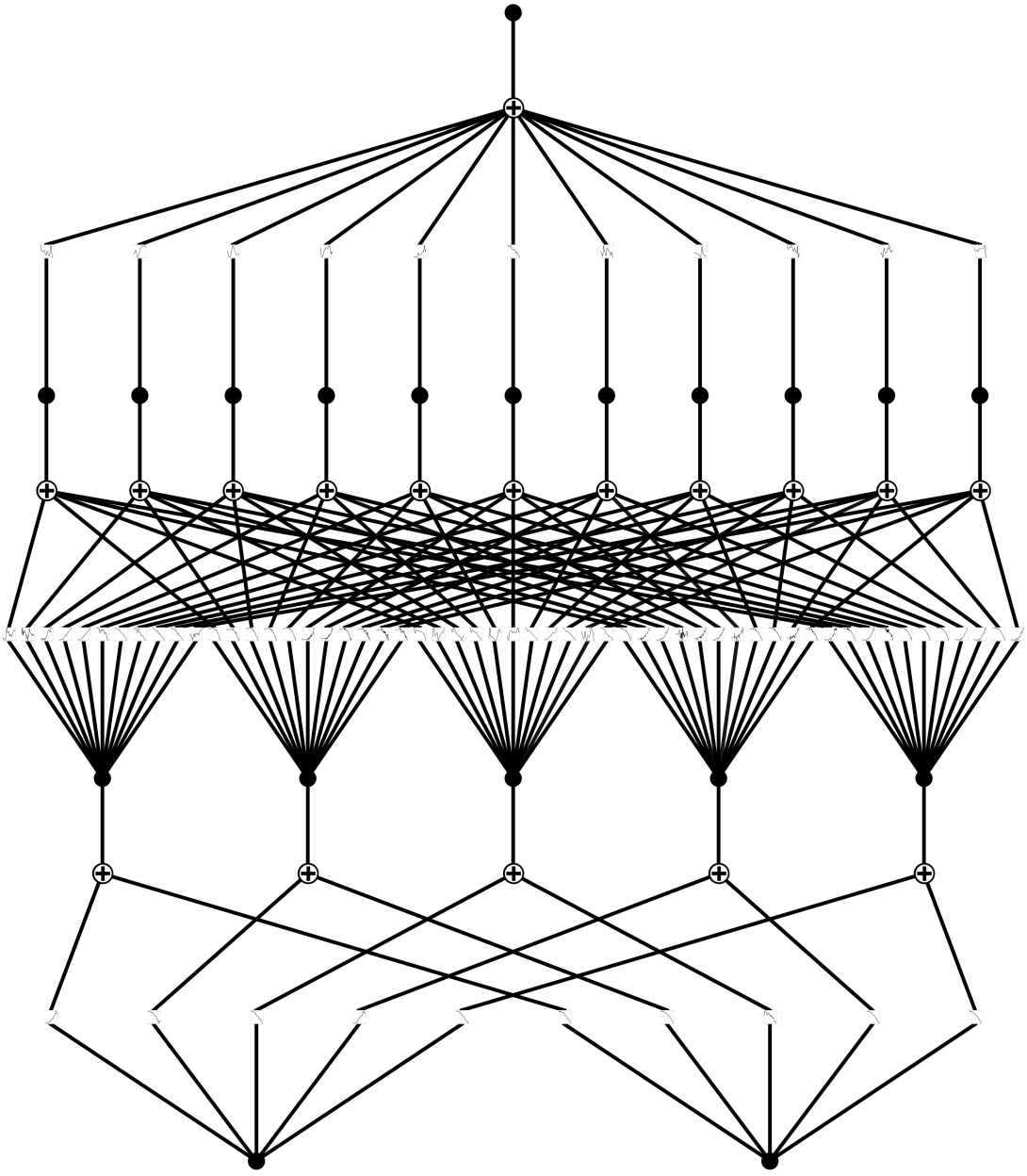


Figure A.2: Trained KAN model