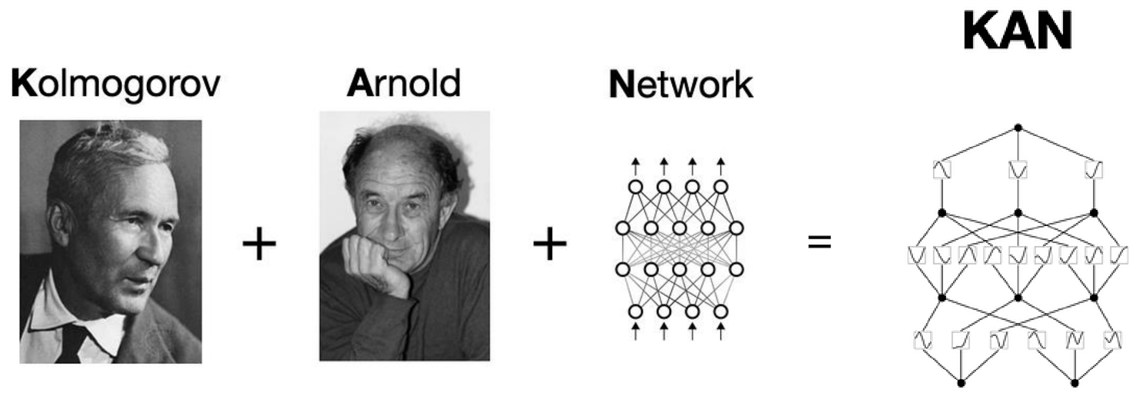
KAN: Kolmogorov-Arnold Networks



Index

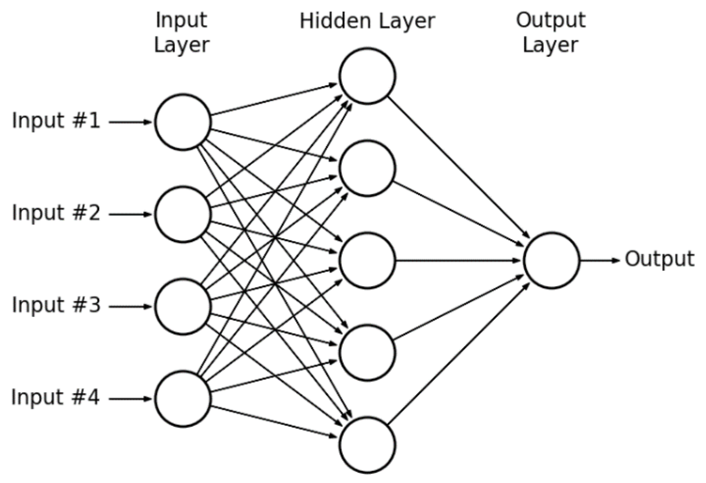
Universal Approximation Theorem

The Universal Approximation Theorem (**UAT**) says that:

"A neural network with a single hidden layer containing a finite number of neurons can be used to approximate ***any*** *continuous function to any desired precision (given suitable non-linear activation functions)"*

$$f(x) = \sum_{i=1}^{N(\epsilon)} a_i \sigma(w_i * x + b_i) \quad \text{s.t.} \quad |f(x) - \hat{f}(x)| < \epsilon$$

MLPs:
the «Standard» Neural Networks



A **Multi-Layer Perceptron (MLP)** is a feedforward neural network composed of fully connected layers of neurons. Each neuron applies a **linear transformation** followed by a **non-linear activation function**, enabling the network to model complex relationships between inputs and outputs.

According to the **Universal Approximation Theorem (UAT)**, an MLP with a single hidden layer containing enough neurons can approximate any continuous function on a bounded domain. Increasing the number of neurons generally improves the quality of approximation. Furthermore, adding additional hidden layers allows the network to **represent functions more efficiently**, exponentially expanding its capacity to approximate highly complex mappings.

Kolmogorov-Arnold representation Theorem

The Kolmogorov-Arnold approximation theorem (**KAT**) says that:

«Any multivariate continuous function can be represented as a composition of a finite number of univariate continuous function of a single variable.»

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

KAT applied to Neural Networks

Introduction, part I

KAT shifts the focus from **approximation** (as in classical MLPs) to **exact functional representation**, suggesting that a properly structured network could **natively encode** the intrinsic functional dependencies among input variables.

Embedding this principle into neural architectures is expected to yield models that are:

- **More efficient**, since the functional structure is captured explicitly;
- **More precise**, due to reduced approximation error;
- **More interpretable**, as each univariate function can be directly analyzed;
- **Less data- and compute-intensive**, because learning focuses on meaningful transformations rather than arbitrary parameter fitting.

KAT applied to Neural Networks

Introduction, part II

Neural Networks that embody the Kolmogorov-Arnold theorem are called **KANs.**

KANs completely **remove** traditional **linear weights** from the network, with all the linear weights being replaced by **learnable univariate activation functions**, with their own learnable coefficients, on the edges themselves.

These activation function are bult using basis functions, like polynomials or ***B-Splines.*** These basis functions are «building blocks» that are combined to create more complex functions. By learning the coefficients of these basis functions, KANs can achieve highly flexible and expressive activation functions on each connection.

KANs leverage the strengths of both B-Splines and MLPs, while avoiding their weaknesses.

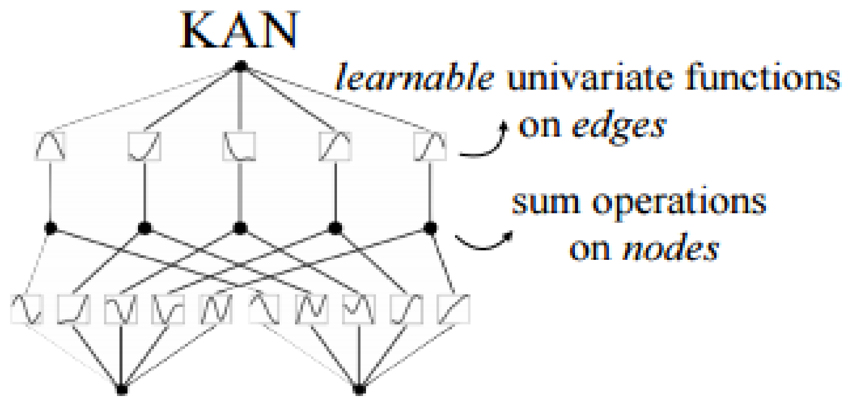
Kolmogorov-Arnold Networks

General Architecture

A KAN consists of a series of interconnected univariate sub-networks, each responsible for learning distinct features of the data.

A general KAN is a composition of L layers, where each layer Φ_l is a function matrix of $\phi_{l,n,m}$:

$$\mathbf{x}_{l+1} = \underbrace{\begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix}}_{\mathbf{\Phi}_l} \mathbf{x}_l,$$


KAN
learnable univariate functions on *edges*
sum operations on *nodes*

Given an input vector $x \in R$, the output of a KAN is then:

$$\mathrm{KAN}(x) = (\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_0)(x)$$

Kolmogorov-Arnold Networks

B-Splines: our activation functions

As anticipated before, the vanilla KAN implements the univariate activation functions using **B-splines**, which provide localized function approximation capabilities.

Therefore, each activation function $\phi(x)$ includes a spline function and a basis function $b(x)$:

$$\phi(x) = w_b b(x) + w_s \mathrm{spline}(x)$$

Where:

- $b(x)$ is a sigmoid linear unit function:

$$b(x) = silu(x) = \frac{1}{1+e^{-x}}$$

- $\mathrm{spline}(x)$ is parameterized as a linear combination of B-Splines, where the coefficient c_i are the **learnable** ones:

$$\mathrm{spline}(x) = \sum_i c_i B_i(x)$$

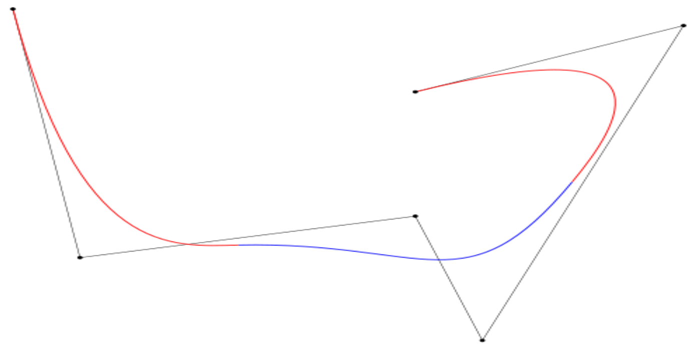
Kolmogorov-Arnold Networks

B-Splines: a geometric POV

B-splines are **piece-wise polynomial curves** composed of a sequence of a **lower-order polynomial segments.** They extend the concept of Bézier curves by introducing a control grid defined by multiple control points, allowing smoother transitions and local control over the curve's shape.

From a **geometric** point of view, we define a B-Spline curve as an interpolated curve where each control point is multiplied by a basis function, that represents the **contribution** of that point to the final curve:

$$S(t) = \sum_{i=0}^{n} P_i \, N_{i,k}\,(t)$$



Kolmogorov-Arnold Networks

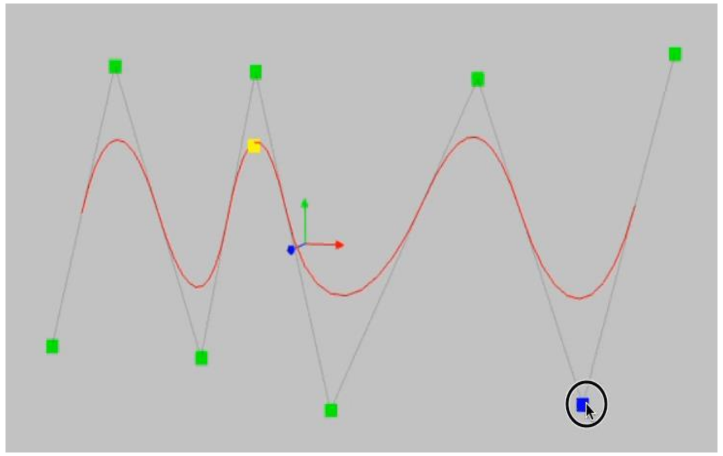
B-Splines: the Locality property

As said before, B-Splines provide **localized** function approximation capabilities.
But what does it mean?

Recall the spline function: $$\mathrm{spline}(x) = \sum_i c_i B_i(x)$$

Basically, it means that each $B_i(x)$ only affects a limited region, such that the total function above is **locally controlled** — changing one coefficient c_i only modifies the curve or the function within the local region/part where $B_i(x)$ is active. This is very different from **polynomials**, where changing one coefficient affects the entire curve globally.

In other words, we can adjust different parts of the approximation **independently,** without disturbing distant parts of the function.



Kolmogorov-Arnold Networks

Interpretability: how it is achieved

Explicit Functional Form:
Each edge in a KAN represents a learned spline-based univariate function, enabling direct inspection of how inputs are transformed.

Local Transparency:
The local support of B-spline activations allows fine-grained analysis of which input regions most influence the network's response.

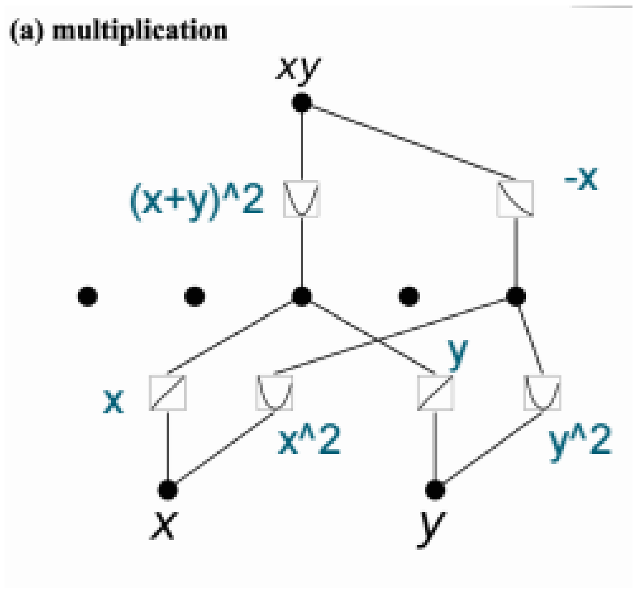
Compositional Structure:
Following the Kolmogorov–Arnold theorem, KANs decompose multivariate mappings into interpretable compositions of simpler functions.

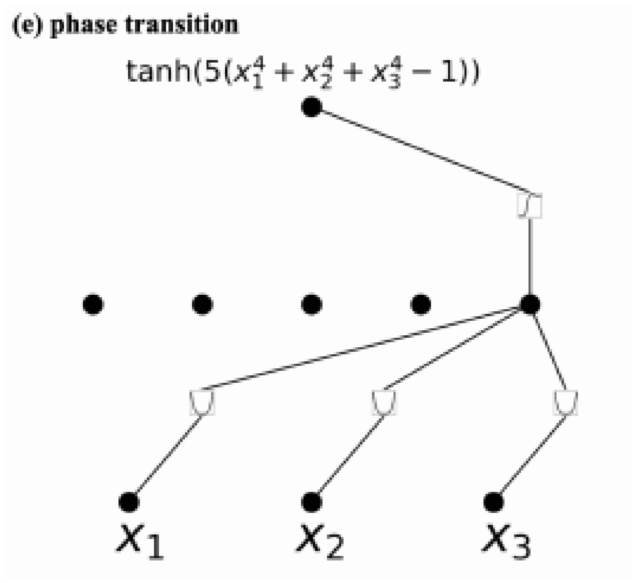
Reduced Black-Box Behavior:
Unlike MLPs with fixed nonlinearities, KANs learn adaptive nonlinearities that can be visualized, constrained, or analyzed directly.

Kolmogorov-Arnold Networks

Interpretability: a visual explanation

In the case of symbolic formulas, KANs are able to reveal their compositional structure:





Kolmogorov-Arnold Networks

Benefits vs Challenges

- Enhanced Accuracy: The learnable activation functions in KANs have the potential to capture more complex relationships in data compared to MLPs, ensuring more flexibility.
- Interpretability: KANs offer a degree of interpretability. By analyzing the learned coefficients of the basis functions, we can gain insights into how the network makes decisions.
- Continual Learning: As new data is added, a KAN retains the previously learned fit and can also adapt to the new changes. This happens due to the locality property of B-Splines.
- Reduced Model Complexity (structural efficiency): By replacing the typical weight matrices in MLPs with spline-based functions that act on edges, KANs dramatically reduce the number of parameters. This reduction in complexity often leads to more efficient training processes and faster convergence rates.

Kolmogorov-Arnold Networks

Benefits vs Challenges

- Increased Training Time (computational efficiency): Learning coefficients for the basis functions can be computationally expensive, in terms of RAM and longer training time compared to MLPs.
- No batch computation: KANs, moreover, are not GPU-efficient because batch computation (computing multiple data points parallely) is not supported yet.
- Hyperparameter Tuning: Finding the optimal configuration of the hyperparams related to the selection and number of basis functions used requires an heavy effort.
- Overfitting: KANs can overfit to everything. This is because the learned activations can bend freely to fit fine-grained variations in the data. Moreover, if the regularization or smoothness constraints are weak, the spline segments can start fitting noise or outliers, not just the true signal.

Key differences

		MLPs	KANs	Explanation
Theoretical Foundation		Universal Approximation Theorem	Kolmogorov-Arnold Representation theorem	The Universal Approximation Theorem deals with **approximating** functions using neural networks. In contrast, the Kolmogorov-Arnold theorem provides a way to **exactly represent** continuous functions through sums of univariate functions, which is expected to make networks more accurate.
Interpretability		Black boxes, due to their intrinsic opaque nature	Allow for a certain degree of interpretability	MLP is more a “blackbox” model, whereas a KAN is supposed to be more **interpretable**. Since KANs learn univariate functions at all levels, which can be inspected if needed, it is easy to determine the structure learned by the network using those formulas.
Parameters count		$\sim N^2 L$	$\sim N^2 L(G+K)$	While MLPs appear to be more efficient than KANs, a point to note is that based on their experiments, KANs usually don't require as much large N as MLPs do. This saves parameters while also achieving better generalization.
Polynom. regression task	Perfor—mances	Good	Excellent	KANs are even 10x slower than MLPs. But KANs considerably **outperform** MLPs by achieving significantly lower test loss across a range of parameter, and at much lower network depth (number of layers).
	Runtime	Fast training	Slow training	
Continual Learning		Not possible. Catastrophic forgetting	Possible	There is no notion of **locality-storage of knowledge** in MLPs, which is found in humans and possible in KANs thanks to the B-Splines.

Adapting KANs to Anomaly Detection

Advantages

Universal Function Approximation: KANs can approximate any continuous function, enabling them to model the underlying patterns in normal time series data accurately.

Sensitivity to Anomalies: Their precise modeling of normal behavior makes them highly sensitive to deviations from expected patterns.

Hierarchical Feature Learning: By stacking layers, KANs can capture both local and global patterns, essential for detecting different types of anomalies.

Interpretability: Each learned spline encodes a transparent input-to-activation mapping, enabling fine-grained understanding of how specific time-series patterns lead to anomaly detection.

Fourier-based Transformations: We can also use Fourier features, which can enable KANs to efficiently capture periodic and so global patterns common in time series data.

Adapting KANs to Anomaly Detection

Why Fourier?

We showed before how B-Splines provide localized function approximation capabilities.

However, this localization property presents a notable consideration in anomaly detection contexts. Since anomalous patterns typically manifest as localized features, **B-splines** may **inadvertently fit these outliers**, potentially compromising model accuracy.

Hence, according to [1], we could redesign KANs replacing the B-spline function with **Fourier** series. Fourier series have local smoothness compared to spline functions, while their natural **periodicity** allows for better modeling of **global** patterns.

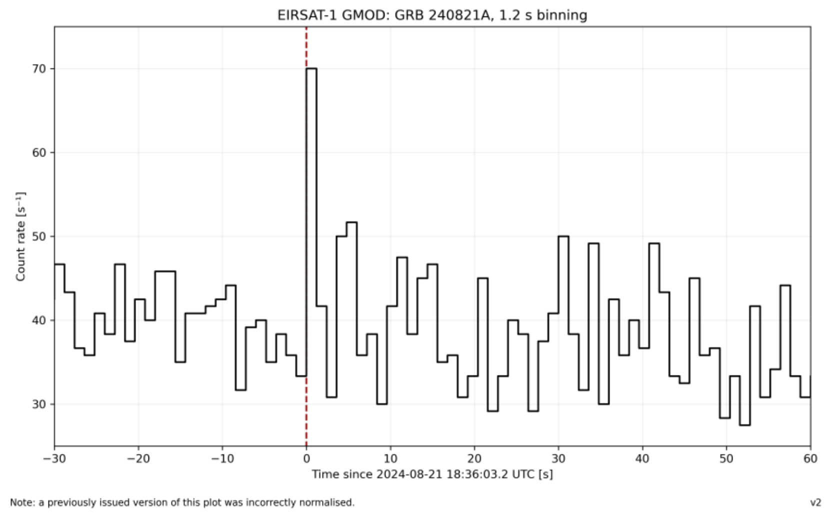
[1] Zhou *et al.*, *KAN-AD: Time Series Anomaly Detection with Kolmogorov-Arnold Networks,* arXiv:2411.00278, 2024.

Adapting KANs to Anomaly Detection

Gamma Ray Bursts (GRBs) case

In **GRB detection**, where anomalies appear as **short**, **intense deviations** in an otherwise stable signal, that's why **Fourier-based KANs wouldn't be appropriate.** This is because Fourier-KANs promote global consistency, but GRB detection needs local sensitivity.

Indeed, in GRB detection, bursts *are the anomalies we want to capture*, so we **don't want to suppress locality** — the standard B-spline version of KANs might be *more appropriate* because it can *emphasize local deviations*.



Early Results