

KAN Tutorial Slides

Daniel Precioso Francisco kuárez
October 8, 2025

Kolmogorov-Arnold Representation Theorem

Let $\Omega \subset \mathbb{R}^d$ be a bounded domain and let $f : \Omega \rightarrow \mathbb{R}$ be a continuous function; i.e. $f \in C(\Omega)$. Then there exist continuous univariate functions

$$\Phi_q : \mathbb{R} \rightarrow \mathbb{R}, \quad q = 1, \dots, 2d + 1;$$

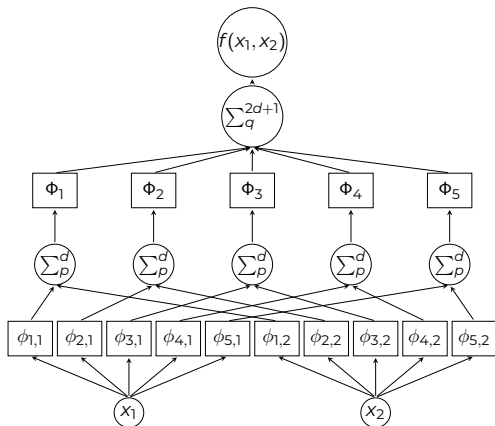
and continuous univariate functions

$$\phi_{pq} : \mathbb{R} \rightarrow \mathbb{R}, \quad p = 1, \dots, d; \quad q = 1, \dots, 2d + 1;$$

such that for every $\mathbf{x} = (x_1, \dots, x_d) \in \Omega$,

$$f(\mathbf{x}) = \sum_{q=1}^{2d+1} \Phi_q \left(\sum_{p=1}^d \phi_{pq}(x_p) \right).$$

Kolmogorov–Arnold Representation Theorem

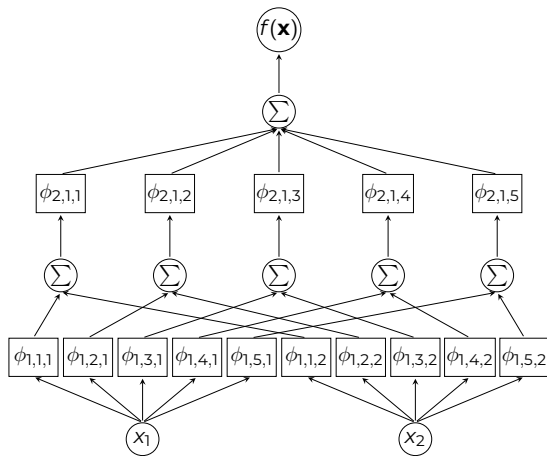


The theorem states that any $f(x_1, x_2)$ can be written as a sum of univariate compositions.

The diagram shows this expression visually: each block represents a component of the decomposition.

Together, they form a **Kolmogorov–Arnold Network (KAN)**.

Kolmogorov–Arnold Networks

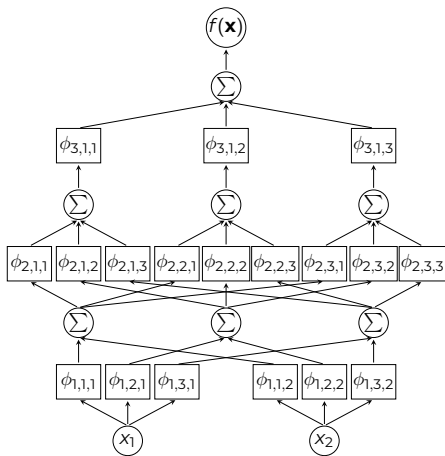


In a network setting, each univariate function is written as $\phi_{d,p,q}$, where:

- d : layer depth
- p : output node index
- q : input node index

This network is a **KAN [2,5,1]**: it has 2 inputs, one hidden layer with 5 nodes, and 1 output.

Kolmogorov–Arnold Networks



KAN [2,3,3,1]: 2 inputs, two hidden layers of 3, 1 output.

Why go deeper?

- **Theory:** Any continuous $f(\mathbf{x})$ admits a shallow KAN $[l, 2l + 1, 1]$.
- **Practice:** Deeper KANs can model non-continuous functions. Depth improves expressivity.

B-Splines

$\phi_{d,p,q}$ can be chosen from any family of continuous univariate functions.
A common choice is the **B-spline** family.

A B-spline of degree k is defined as:

$$B_k(x) = \sum_{i=1}^n P_i N_{i,k}(x)$$

where n is the number of control points,
 $N_{i,k}$ are the basis functions of degree k ,
and P_i are the control points (spline weights).

B-Splines - Basis Function

The basis functions follow the standard **Cox–de Boor recursive definition**:

$$N_{i,0}(x) = \begin{cases} 1, & t_i \leq x < t_{i+1} \\ 0, & \text{otherwise} \end{cases}$$

$$N_{i,k}(x) = \frac{x - t_i}{t_{i+k} - t_i} N_{i,k-1}(x) + \frac{t_{i+k+1} - x}{t_{i+k+1} - t_{i+1}} N_{i+1,k-1}(x), \quad k > 0$$

where $t_i \in [t_1, t_m]$ is the **knot vector**, a non-decreasing sequence of real numbers of length $m = n + k + 1$.

B-Splines - Knot Vector

Knot vector $\mathbf{t} = (t_1, \dots, t_m)$ with

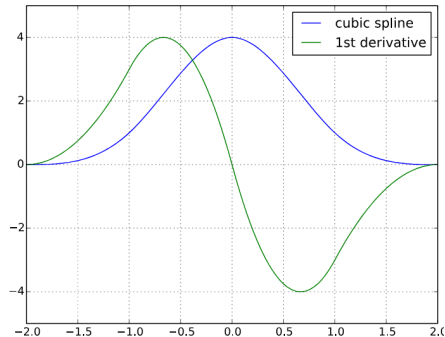
$$m = n+k+1, \quad a = t_k+1, \quad b = t_{m-k}$$

Clamped uniform knots on $[a, b]$:

$$t_1 = \dots = t_{k+1} = a,$$

$$t_{n+1} = \dots = t_{n+k+1} = b,$$

$$t_{k+j} = a + \frac{j}{n-k}(b-a), \quad j = 1, \dots, n-k$$



$$\mathbf{t} = (-2, -2, -2, -2, -1, 0, 1, 2, 2, 2, 2)$$

$$\mathbf{P} = (0, 0, 0, 6, 0, 0, 0)$$

B-Splines as KAN Edges

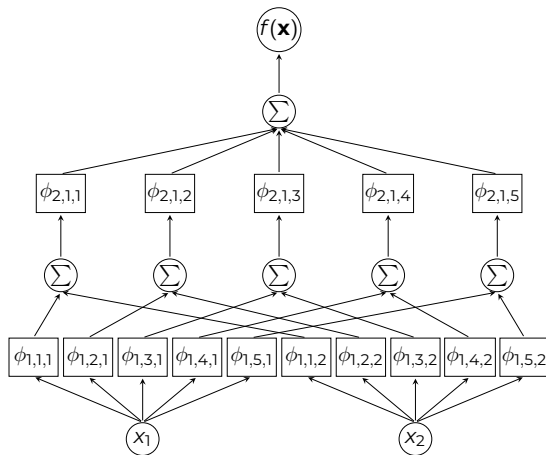
All univariate functions share the same spline degree k and number of control points n . Each $\phi_{d,p,q}$ combines a basis function (similar to residual connections) with a B-spline expansion:

$$\phi(x) = w_b b(x) + \sum_{i=1}^n P_i N_{i,k}(x)$$

Here, w_b is the learnable weight of the basis function, and the control point coefficients P_i scale the individual B-spline functions directly. We choose the basis as:

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

KAN Parameters



Hyperparameters

- n : number of control points.
- k : B-spline degree.

Learnable parameters (for each edge)

- P_i : control points, $i \in [1, n]$.
- w_b : basis weight.

KAN Backpropagation

Loss function is L2 (RMkE):

$$L = \|y - \hat{y}\|_2 = \|f(\mathbf{x}) - \hat{f}_d(\mathbf{x})\|_2 = \left\| f(\mathbf{x}) - \sum_q \phi_{d,q}(\mathbf{x}) \right\|_2$$

Where d is the last layer, and $p = 1$ because we have a single output. The coefficients of that layer are $P_{d,q,i}$.

$$\frac{\partial L}{\partial P_{d,q,i}} = \frac{\partial L}{\partial \hat{f}_d(\mathbf{x})} \cdot \frac{\partial \hat{f}_d(\mathbf{x})}{\partial P_{d,q,i}}$$

And for the previous layer $d - 1$:

$$\frac{\partial L}{\partial P_{d-1,p,q,i}} = \frac{\partial L}{\partial \hat{f}_d(\mathbf{x})} \cdot \frac{\partial \hat{f}_d(\mathbf{x})}{\partial \hat{f}_{d-1,p}(\mathbf{x})} \cdot \frac{\partial \hat{f}_{d-1,p}(\mathbf{x})}{\partial P_{d-1,p,q,i}}$$

Capabilities of KANs with B-Splines

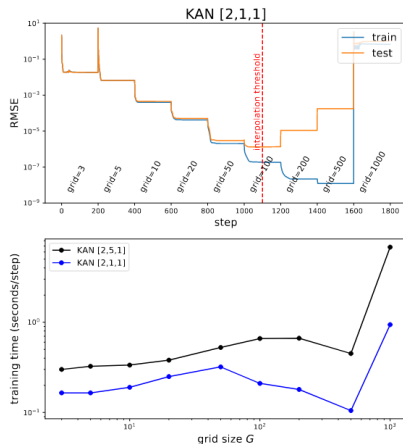
- **nrid extension:** progressively increase model capacity by refining the spline grid without retraining from scratch.
- **Continual learning:** local support ensures new information affects only nearby regions, reducing catastrophic forgetting.
- **kparsity:** regularization and pruning remove redundant components, simplifying the model without major accuracy loss.
- **kymbolic regression:** univariate structure enables conversion of learned functions into interpretable closed-form expressions.

Capabilities - nrid Extension

nrid extension refines a trained KAN by adding more spline knots without restarting training.

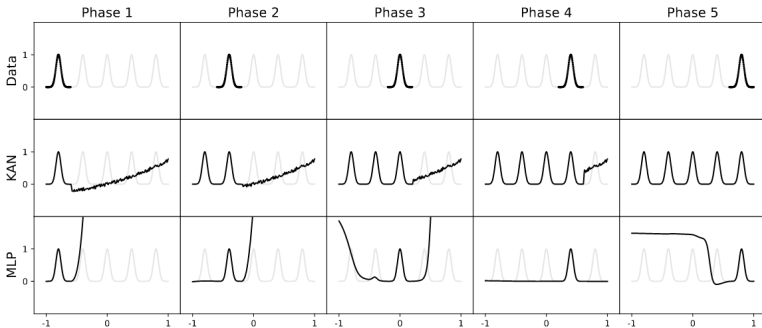
- Train on a coarse grid first.
- Add knots to increase resolution and capacity.
- Initialize new coefficients by least-squares fitting.
- Continue training to improve accuracy.

Test loss often improves until the parameter count roughly matches the number of data points.



Capabilities - Continual Learning

Because B-splines have **local support**, updates to $\phi(x)$ in one region of the input space affect only nearby points. This locality mitigates **catastrophic forgetting**, a common issue in MLPs where learning new data can overwrite previously acquired knowledge.

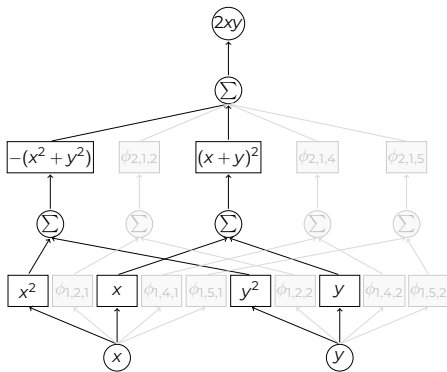


Capabilities - kparsity

kparsity removes unnecessary components, revealing the essential structure of our target function.

- **Regularization** drives many spline weights toward zero.
- Irrelevant edges can be pruned after training.
- The result is a compact, interpretable network.

kparsity helps towards **interpretability**.



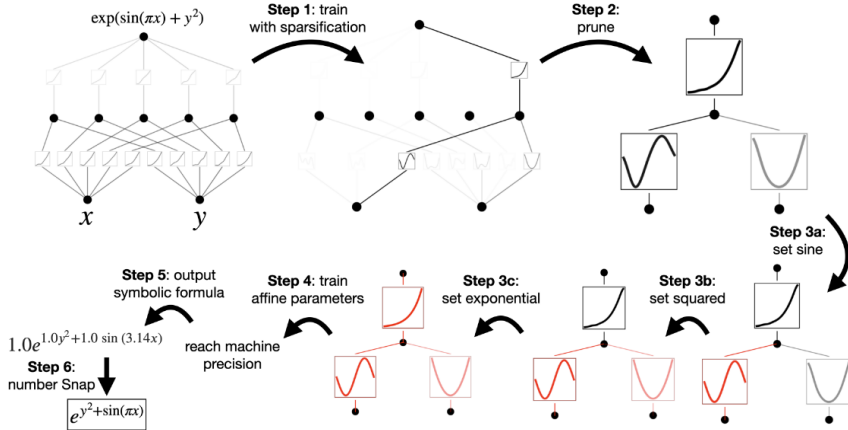
Capabilities - kymbolic Regression

KANs provide an interpretable path from neural models to closed-form expressions:

- Each learned $\phi_{d,p,q}$ is a univariate function, which can often be approximated by simple analytic forms (e.g., sin, exp, log).
- After training, these functions are “snapped” to symbolic templates via affine fitting, producing human-readable equations.
- The resulting network can be viewed as a composition graph of symbolic functions approximating $f(x)$.

This makes KANs suitable not only for prediction but also for **discovering interpretable laws** from data.

KAN Train ksteps



Limitations of KANs

- **Parameter & memory blow-up.** For comparable width, KAN layers require substantially more parameters and activations than FC layers; memory scales poorly with I, O, B and grid/spline settings (n, k) .
- **Convergence to sharp minima.** Hessian spectrum analyses show KANs tend to sharper minima \Rightarrow weaker generalization (vision benchmarks).
- **Underperformance at scale.** On kciML (Neural ODEs), vision (Mixer/DeiT), and operator learning (FNO), KANs typically underperform MLPs despite higher cost.
- **Runtime/feasibility.** Longer training times and frequent OOM at moderate batch sizes on 16nB nPUs when scaling depth/width.

Limitations - Parameter Complexity

Fully Connected (FC) layer with O outputs and I inputs:

$$O \times (I + 1)$$

KAN layer (B-spline) with two shortcut weights, n control points, k order:

$$I \times O \times (2 + n + k)$$

Even “width-matched” KANs are often an order of magnitude larger than FC/MLP layers.

Limitations - Memory Footprint (Training)

Fully Connected (FC) layer accounting for both forward and reverse pass (one matrix each), and the parameter count:

$$O \times B + O \times B + O \times (I + 1)$$

KAN layer (B-spline)

$$I \times (1+n+k) \times B + I \times (n+k-1) \times B + I \times O \times (2+n+k).$$

KANs allocate extra tensors for B-spline evaluation and De Boor recursion.

Limitations - Convergence & Generalization

- Large-batch training tends to *sharp* minima (many large positive eigenvalues); *flat* minima correlate with better generalization.
- Empirically, KAN variants show Hessian spectra with **more positive eigenvalues** than MLP counterparts, *even at small batches*.
- **Effect:** test-time accuracy lags behind MLPs on vision tasks; e.g., DeiT on CIFAR-100 shows $\sim 8\text{--}12\%$ lower Top-1 for B-spline KANs with **more** parameters.

Empirical Evidence Across Domains

Neural ODEs (kciML).

- On Lotka–Volterra, Pleiades, Spiral ODE, KAN-ODEs reach **higher final loss**; MLP-ODEs fit dynamics better.



Computer Vision.

- MLP-Mixer/Conv-Mixer/DeiT: KAN variants converge to **sharper minima** and **lower accuracy**; parameter counts are 2–3× higher for KANs at similar or worse accuracy.

Operator Learning (FNO).

- Replacing lift/projection MLPs with KANs yields **slightly worse** test loss without residuals; with residuals, small gains appear but at **prohibitive memory cost**.

References

-  Liu, Ziming et al. (Feb. 2025). *KAN: Kolmogorov-Arnold Networks*. en. arXiv:2404.19756 [cs]. DOI: [10.48550/arXiv.2404.19756](https://doi.org/10.48550/arXiv.2404.19756). URL: <http://arxiv.org/abs/2404.19756> (visited on 09/19/2025).
-  Pal, Avik and Dipankar Das (n.d.). “Understanding the Limitations of B-Spline KANs: Convergence Dynamics and Computational Efficiency”. en. In: ().