

Introducción

Inspirados en el teorema de representación de Kolmogorov–Arnold, Ziming Liu propone las Kolmogorov–Arnold Networks (KANs) como una alternativa a los Multi-Layer Perceptron (MLP). A diferencia de las MLP, que emplean funciones de activación fijas en cada nodo ("neurona"), las KAN incorporan funciones de activación aprendibles en cada arista ("peso") parametrizadas por funciones univariadas como splines.

Theorem 10.3.1 (Kolmogorov) Any continuous function $f(x_1, \dots, x_n)$ defined on I_n , $n \geq 2$, can be written in the form

$$f(x_1, \dots, x_n) = \sum_{j=1}^{2n+1} \chi_j \left(\sum_{i=1}^n \psi_{ij}(x_i) \right),$$

where χ_j , ψ_{ij} are continuous functions of one variable and ψ_{ij} are monotone functions which are not dependent on f .

Figure 1. Teorema de representación de Kolmogorov–Arnold de Deep Learning Architectures [1]

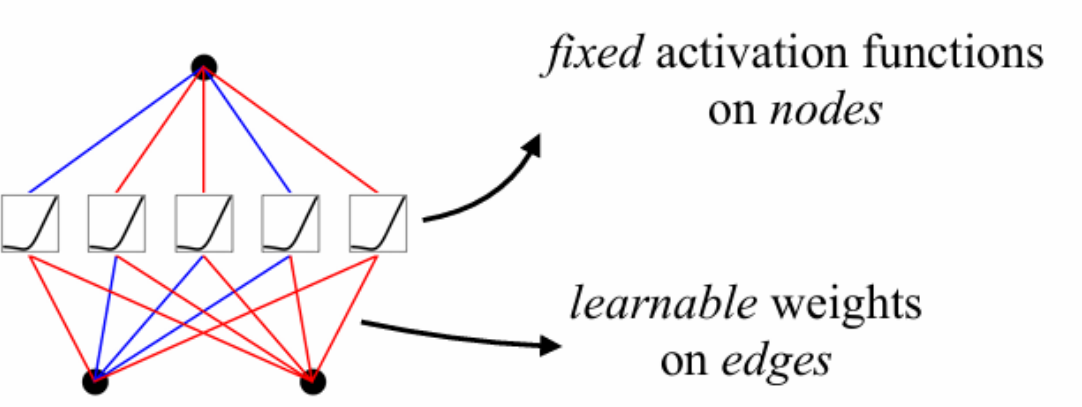
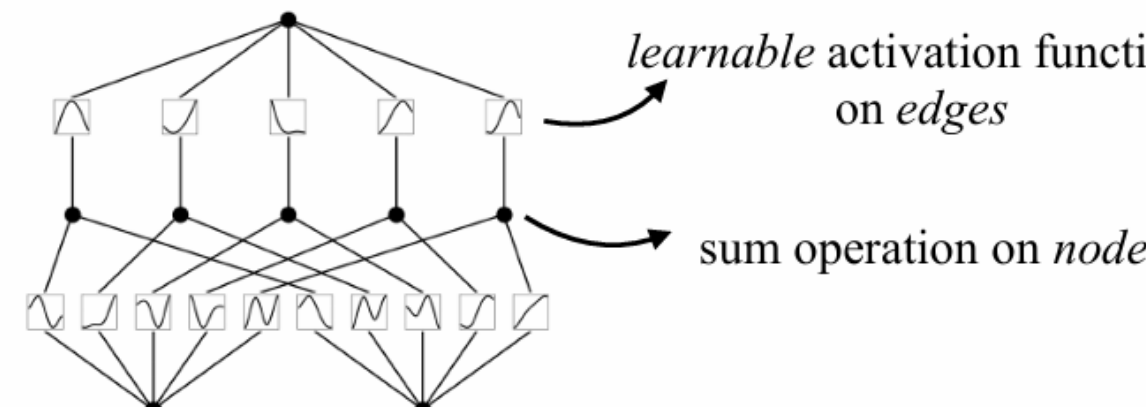
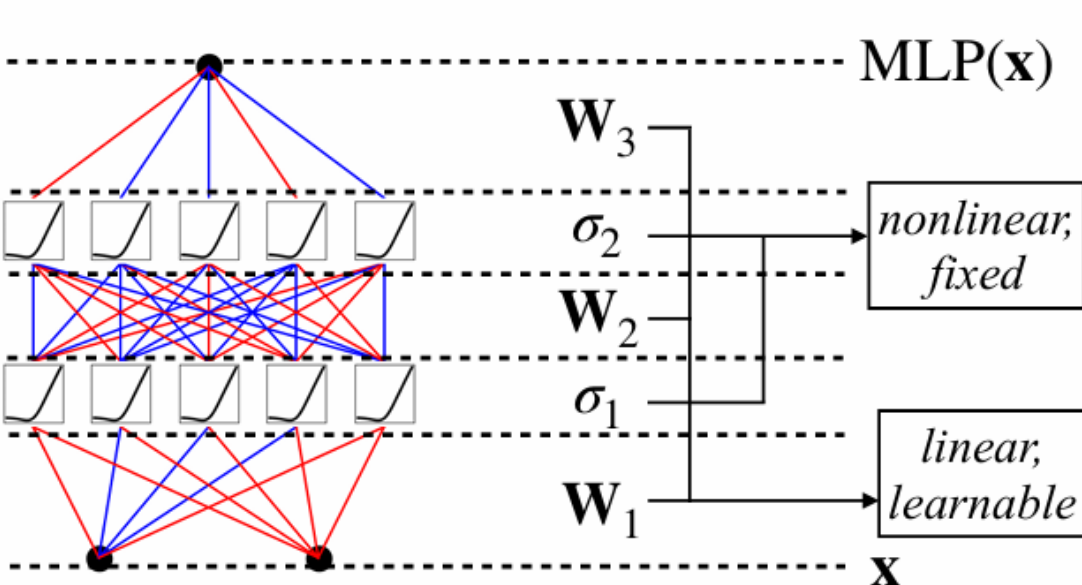
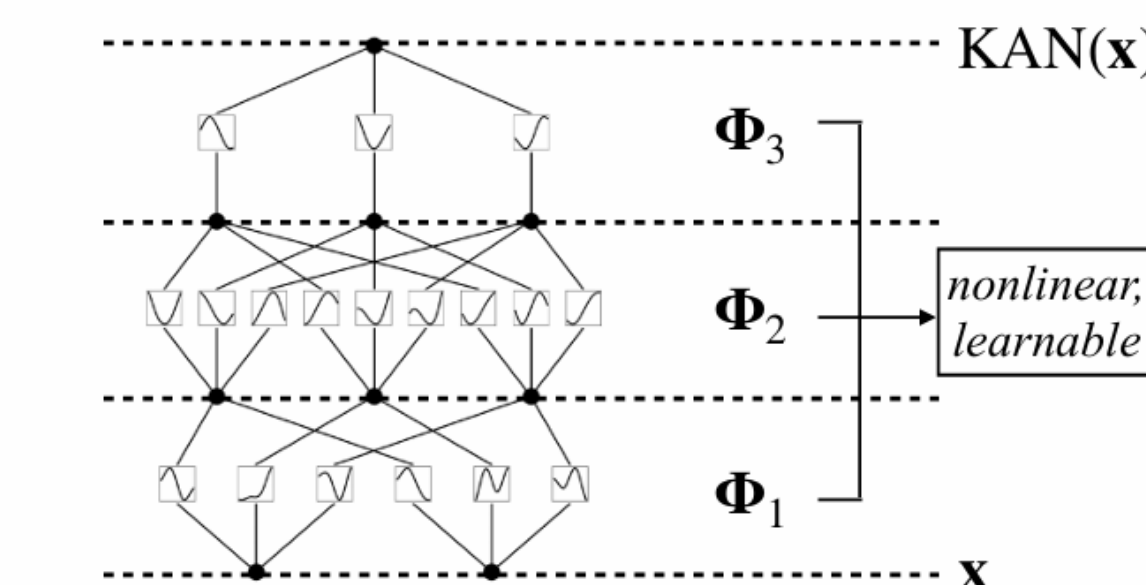
Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{M(\epsilon)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
Model (Shallow)	(a) 	(b) 
Formula (Deep)	$\text{MLP}(\mathbf{x}) = (\mathbf{W}_3 \circ \sigma_2 \circ \mathbf{W}_2 \circ \sigma_1 \circ \mathbf{W}_1)(\mathbf{x})$	$\text{KAN}(\mathbf{x}) = (\Phi_3 \circ \Phi_2 \circ \Phi_1)(\mathbf{x})$
Model (Deep)	(c) 	(d) 

Figure 2. MLPs vs KANs

Detalles de implementación

- Residual activation functions:
- Initializaiton sacles:
- Update of spline grids:

texto texto

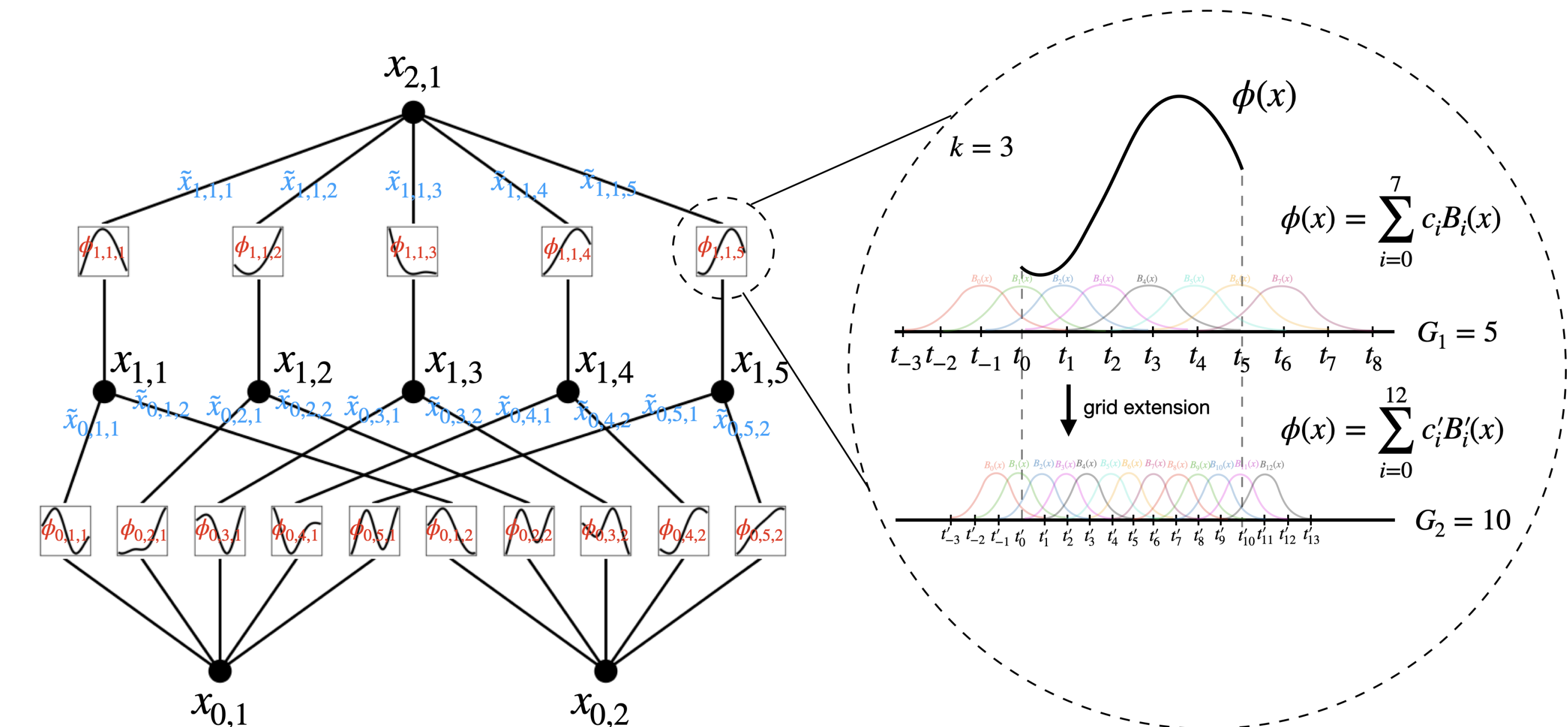


Figure 3. B-splines texto texto texto texto texto

texto texto

Implementaciones:

- Original del autor <https://github.com/KindXiaoming/pykan>
- Eficiente en PyTorch <https://github.com/Blealtan/efficient-kan>

Teorema de aproximación

Sea $f : [0, 1]^{n_0} \rightarrow \mathbb{R}$ tal que admite una representación

$$f = (\Phi_{L-1} \circ \Phi_{L-2} \circ \dots \circ \Phi_1 \circ \Phi_0) \mathbf{x},$$

$$= \left(\sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1, i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \left(\dots \left(\sum_{i_2=1}^{n_2} \phi_{2, i_2} \left(\sum_{i_1=1}^{n_1} \phi_{1, i_1} \left(\sum_{i_0=1}^{n_0} \phi_{0, i_0}(x_{i_0}) \right) \right) \right) \dots \right) \right) \right)_{i_L=1, \dots, n_L}$$

donde $\phi_{l,i,j} \in \mathbb{C}^{k+1}([0, 1])$, entonces:

Dado G grilla, $\exists C > 0$ y $\phi_{l,i,j}^G$ B-spline de orden k (\equiv degree) tal que para cada $0 \leq m \leq k$:

$$\left\| f - (\Phi_{L-1}^G \circ \Phi_{L-2}^G \circ \dots \circ \Phi_1^G \circ \Phi_0^G) \mathbf{x} \right\|_{C^m} \leq C G^{m-(k+1)}.$$

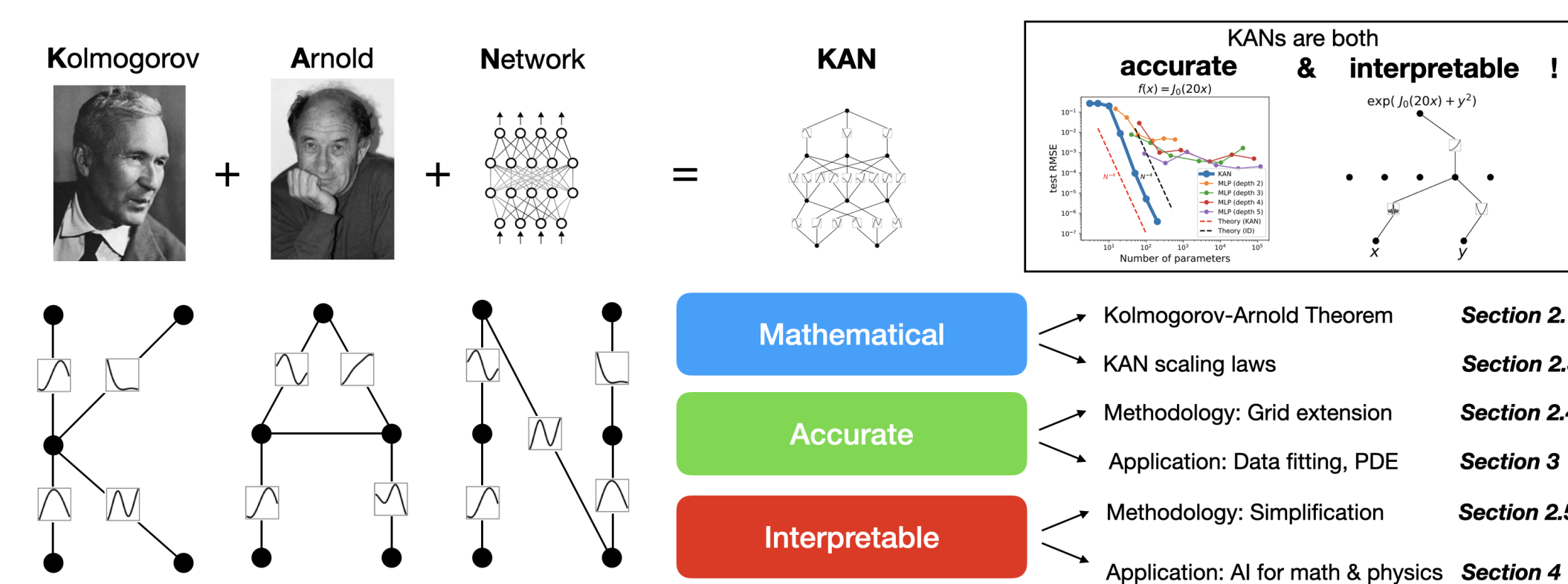
Donde

$$\|g\|_{C^m} := \max_{0 \leq \beta \leq m} \sup_{x \in [0,1]^n} |D^\beta g(x)|.$$

mide la magnitud de las derivadas hasta el orden m .

texto texto

Algunos resultados o experimentos



texto texto

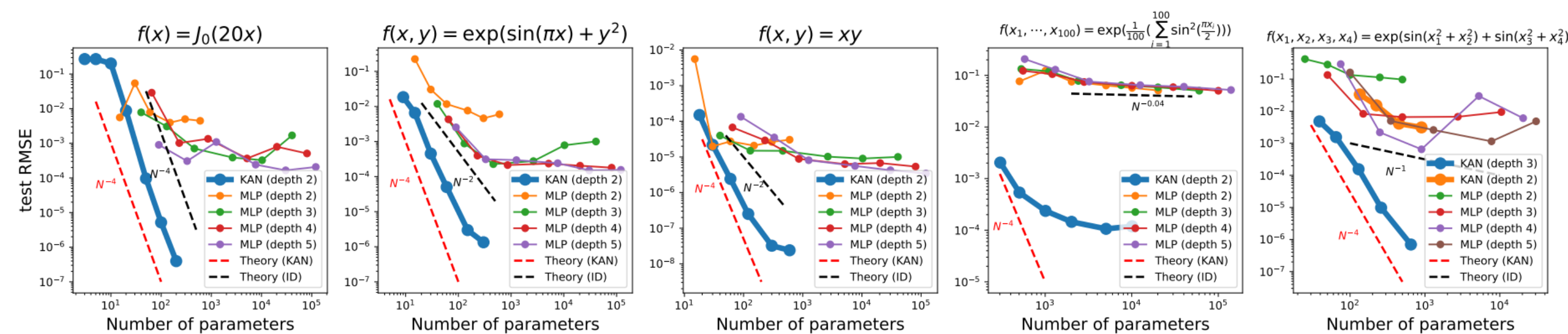


Figure 4. texto texto texto texto texto

texto texto

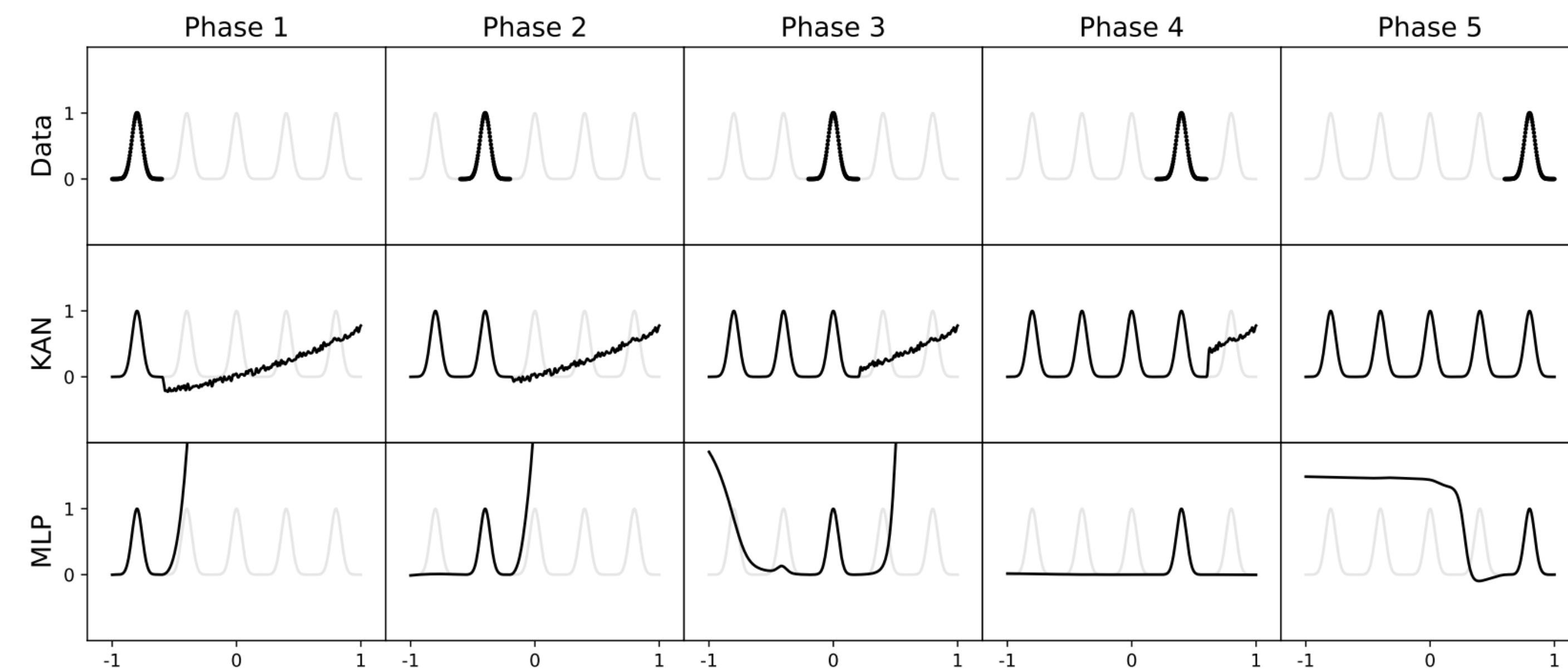


Figure 5. texto texto texto texto texto

texto texto

Referencias

- Ovidiu Calin. *Deep Learning Architectures A Mathematical Approach* p[296]. Springer Cham, 2020.
- Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan:kolmogorov–arnold networks. *Accepted by International Conference on Learning Representations (ICLR) 2025.*, 2024.

Este póster emplea figuras de de [2] bajo la licencia CC BY 4.0.