

KAN: Kolmogorov–Arnold Networks

Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruele,
James Halverson, Marin Soljačić, Thomas Y. Hou, Max Tegmark
Preprint ArXiv

Kolmogorov–Arnold representation theorem

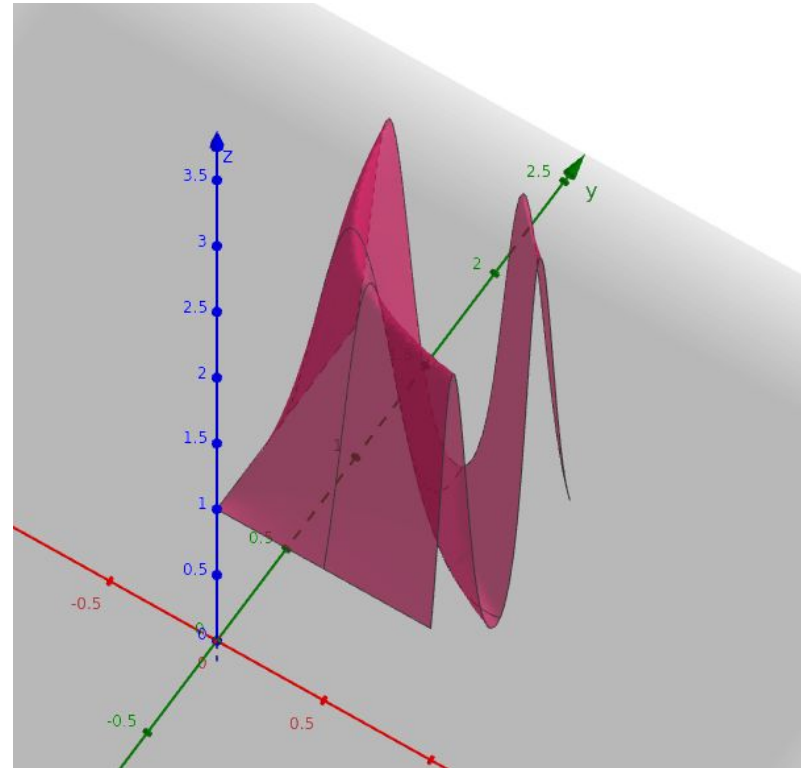
Pour une fonction continue f , de $[0, 1]^n$ dans $\mathbb{R}$, il existe $(2n + 1)(n + 1)$ fonctions de $\mathbb{R}$ dans $\mathbb{R}$ telles que :

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

Kolmogorov–Arnold representation theorem

Un premier exemple :

$$f(x, y) = \exp(\sin(10xy))$$



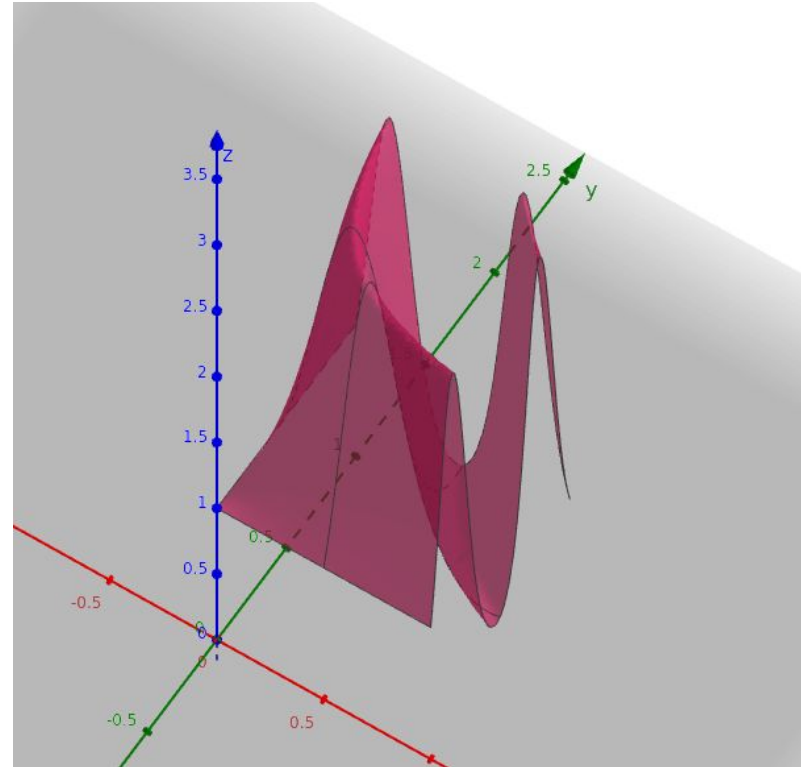
Kolmogorov–Arnold representation theorem

Un premier exemple :

$$f(x, y) = \exp(\sin(10xy))$$

On remarque que :

$$10xy = \exp(10 \log(x) + \log(y))$$



Kolmogorov–Arnold representation theorem

Un premier exemple :

$$f(x, y) = \exp(\sin(10xy))$$

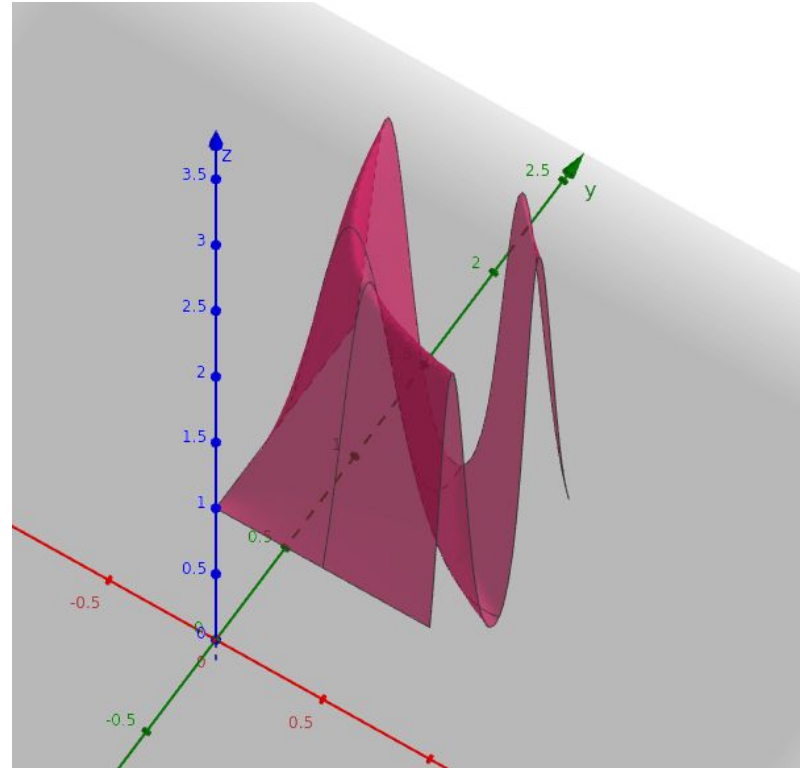
On remarque que :

$$10xy = \exp(10 \log(x) + \log(y))$$

Donc on prend:

$$\Phi_1(x) = \exp(\sin(10 \exp(x)))$$

$$\Phi_{1,\{1,2\}}(x) = \log(x)$$



Kolmogorov–Arnold representation theorem

- En un sens : la seule opération multivarié qui existe est la somme.
- En pratique :
 - Il existe plusieurs versions du théorème, affaiblissant la formule, ou durcissant certaines conditions.
 - Il semblerait qu'il existe une preuve constructive, mais j'ai pas eu le temps de lire le papier en question (et potentiellement ça ne reste pas exploitable).
 - Rien ne garantit que les Φ soient continues en pratique !

KA-Networks

Les idées principales :

- Affaiblir le théorème en autorisant plus de compositions pour éviter les fonctions discontinues
- Paramétriser les fonctions Phi pour qu'elles puissent être apprises

$$f(\mathbf{x}) = \sum_{i_{L-1}=1}^{n_{L-1}} \phi_{L-1,i_L,i_{L-1}} \left(\sum_{i_{L-2}=1}^{n_{L-2}} \cdots \left(\sum_{i_2=1}^{n_2} \phi_{2,i_3,i_2} \left(\sum_{i_1=1}^{n_1} \phi_{1,i_2,i_1} \left(\sum_{i_0=1}^{n_0} \phi_{0,i_1,i_0}(x_{i_0}) \right) \right) \right) \cdots \right)$$

Note: il s'agit de la version un peu trop grosse mais plus facile à décoder dans une présentation de 10m

Paramétrisation des fonctions Phi :

Residual activation functions. We include a basis function $b(x)$ (similar to residual connections) such that the activation function $\phi(x)$ is the sum of the basis function $b(x)$ and the spline function:

$$\phi(x) = w (b(x) + \text{spline}(x)) . \quad (2.10)$$

We set

$$b(x) = \text{silu}(x) = x/(1 + e^{-x}) \quad (2.11)$$

in most cases. $\text{spline}(x)$ is parametrized as a linear combination of B-splines such that

$$\text{spline}(x) = \sum_i c_i B_i(x) \quad (2.12)$$

where c_i s are trainable. In principle w is redundant since it can be absorbed into $b(x)$ and $\text{spline}(x)$. However, we still include this w factor to better control the overall magnitude of the activation function.

Paramétrisation des fonctions Phi :

Theorem 2.1 (Approximation theory, KAT). *Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$. Suppose that a function $f(\mathbf{x})$ admits a representation*

$$f = (\Phi_{L-1} \circ \Phi_{L-2} \circ \dots \circ \Phi_1 \circ \Phi_0)\mathbf{x}, \quad (2.14)$$

as in Eq. (2.7), where each one of the $\Phi_{l,i,j}$ are $(k+1)$ -times continuously differentiable. Then there exists a constant C depending on f and its representation, such that we have the following approximation bound in terms of the grid size G : there exist k -th order B-spline functions $\Phi_{l,i,j}^G$ such that for any $0 \leq m \leq k$, we have the bound

$$\|f - (\Phi_{L-1}^G \circ \Phi_{L-2}^G \circ \dots \circ \Phi_1^G \circ \Phi_0^G)\mathbf{x}\|_{C^m} \leq CG^{-k-1+m}. \quad (2.15)$$

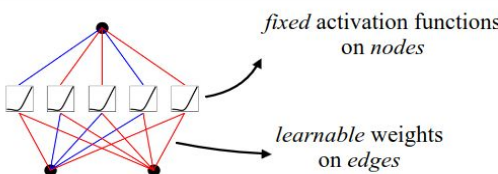
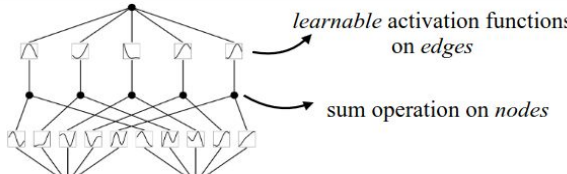
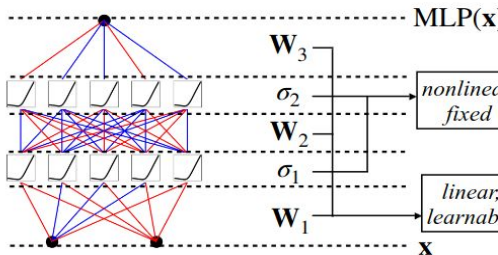
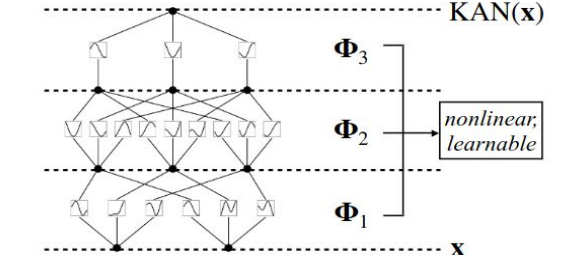
Here we adopt the notation of C^m -norm measuring the magnitude of derivatives up to order m :

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

Petit(s) commentaire(s) :

- Le théorème justifie que, si f est une composition de fonctions, alors la forme imposée sur les Φ est suffisante
- Ils disent que la formule est indépendante de la dimension car “on ne fait que des fonctions 1d de toutes façon”.
 - Pour une famille de fonction de dimension croissante, on pourrait potentiellement arriver à faire tendre L vers l’infini, tout comme C .

Bref retour sur les KAN :

Model	Multi-Layer Perceptron (MLP)	Kolmogorov-Arnold Network (KAN)
Theorem	Universal Approximation Theorem	Kolmogorov-Arnold Representation Theorem
Formula (Shallow)	$f(\mathbf{x}) \approx \sum_{i=1}^{N(e)} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$	$f(\mathbf{x}) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^n \phi_{q,p}(x_p) \right)$
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) 

Resultats (toy datasets):

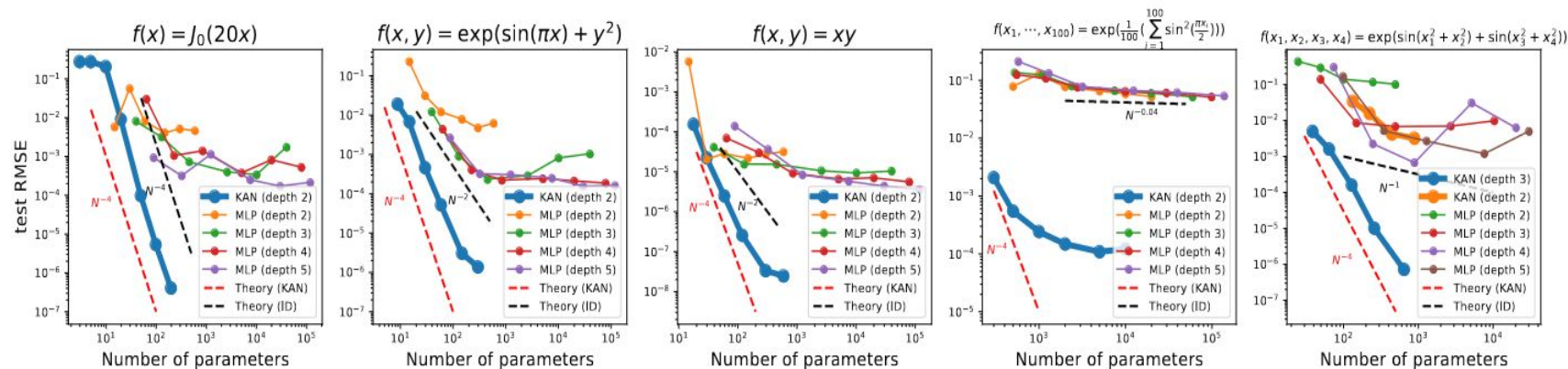
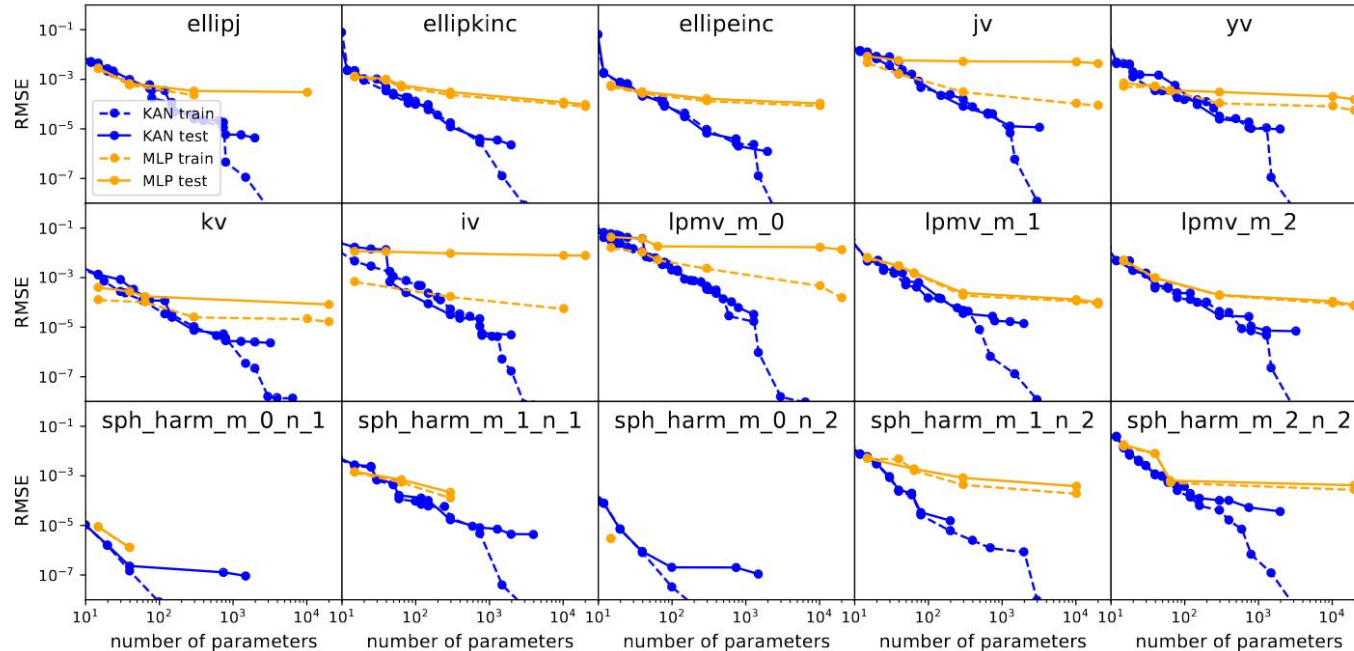


Figure 3.1: Compare KANs to MLPs on five toy examples. KANs can almost saturate the fastest scaling law predicted by our theory ($\alpha = 4$), while MLPs scales slowly and plateau quickly.

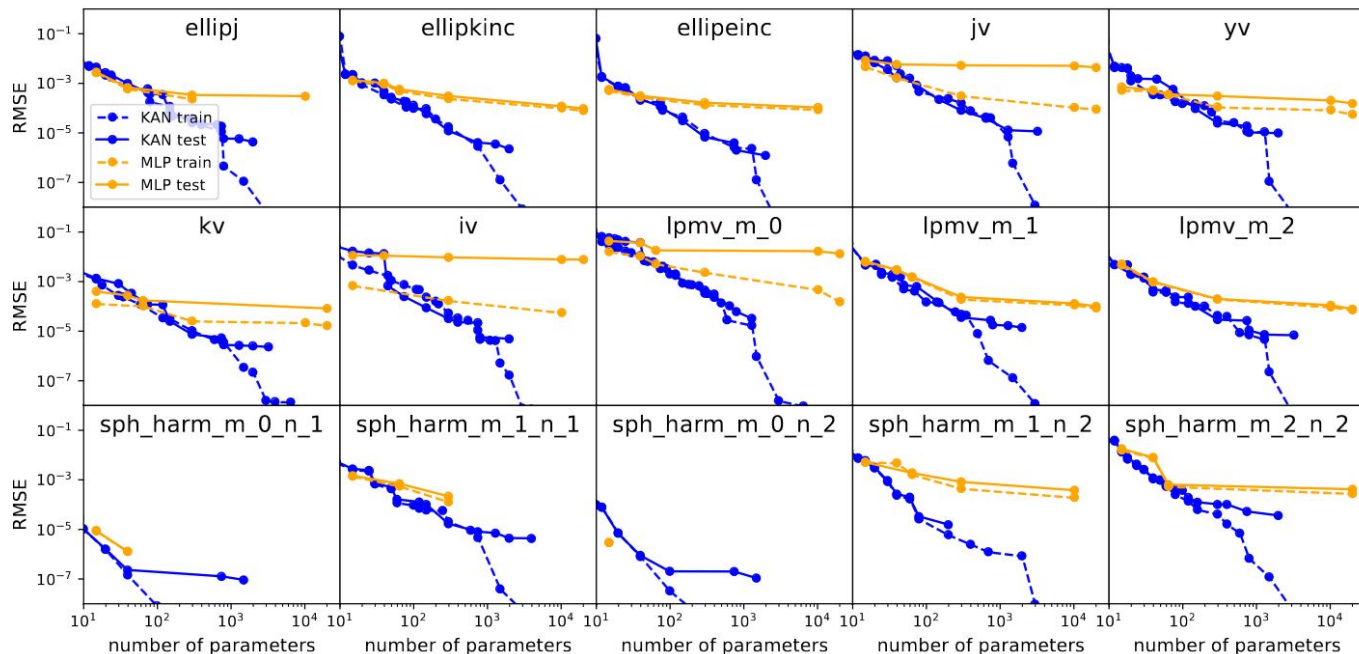
- Plus précis que les MLP
- Atteints (parfois), la limite théorique des splines

Resultats (toy datasets):



- Overfit beaucoup plus facilement
- Que des fonctions 2D...

Resultats (toy datasets):



- Overfit beaucoup plus facilement
- Que des fonctions 2D... Autre exemple (Feynman dataset): au plus 6D

Resultats (toy datasets):

We consider a Poisson equation with zero Dirichlet boundary data. For $\Omega = [-1, 1]^2$, consider the PDE

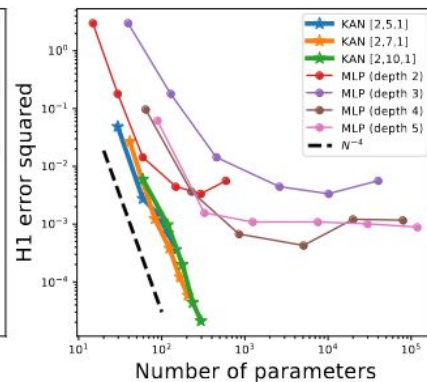
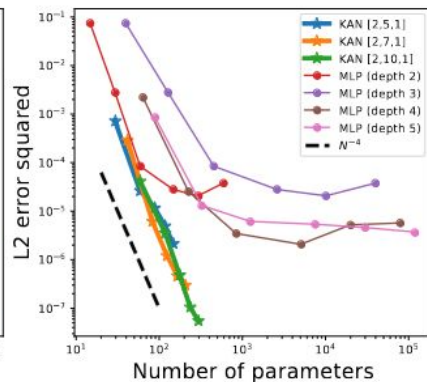
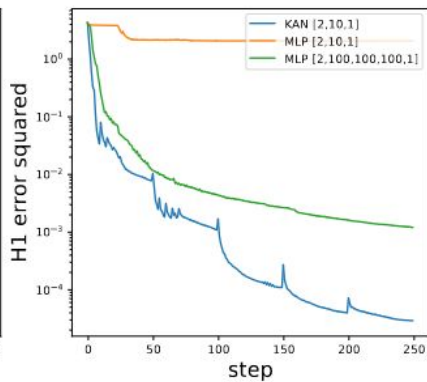
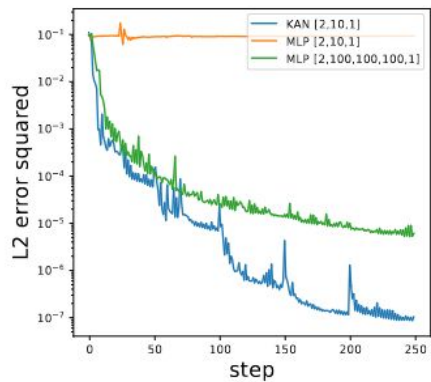
$$\begin{aligned} u_{xx} + u_{yy} &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{3.2}$$

We consider the data $f = -\pi^2(1 + 4y^2) \sin(\pi x) \sin(\pi y^2) + 2\pi \sin(\pi x) \cos(\pi y^2)$ for which $u = \sin(\pi x) \sin(\pi y^2)$ is the true solution. We use the framework of physics-informed neural networks

(PINNs) [22, 23] to solve this PDE, with the loss function given by

$$\text{loss}_{\text{pde}} = \alpha \text{loss}_i + \text{loss}_b := \alpha \frac{1}{n_i} \sum_{i=1}^{n_i} |u_{xx}(z_i) + u_{yy}(z_i) - f(z_i)|^2 + \frac{1}{n_b} \sum_{i=1}^{n_b} u^2,$$

Resultats (toy datasets):



Resultats (pas toy datasets):

- ???
- Personnellement : je n'ai rien compris à leurs exemples de physique quantique et de théorie des noeuds.
- On va dire que ça marche hein :)

Symbolic regression :

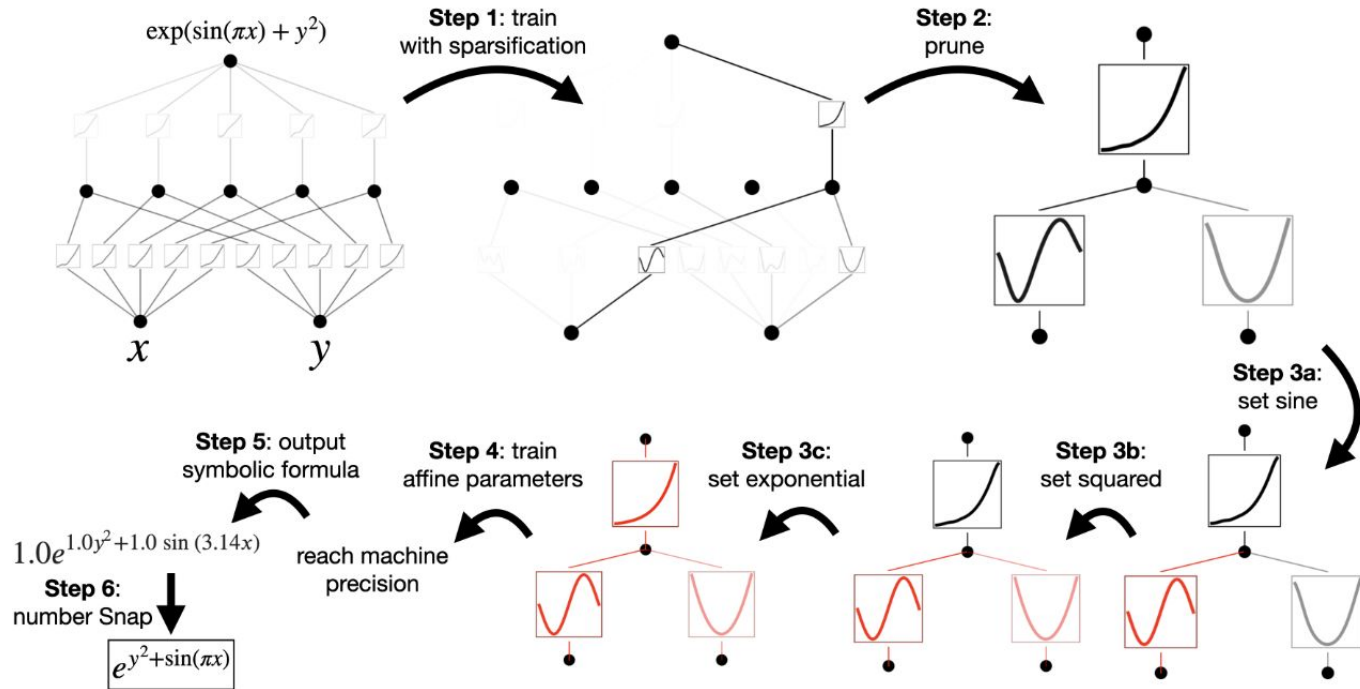


Figure 2.4: An example of how to do symbolic regression with KAN.

Symbolic regression :

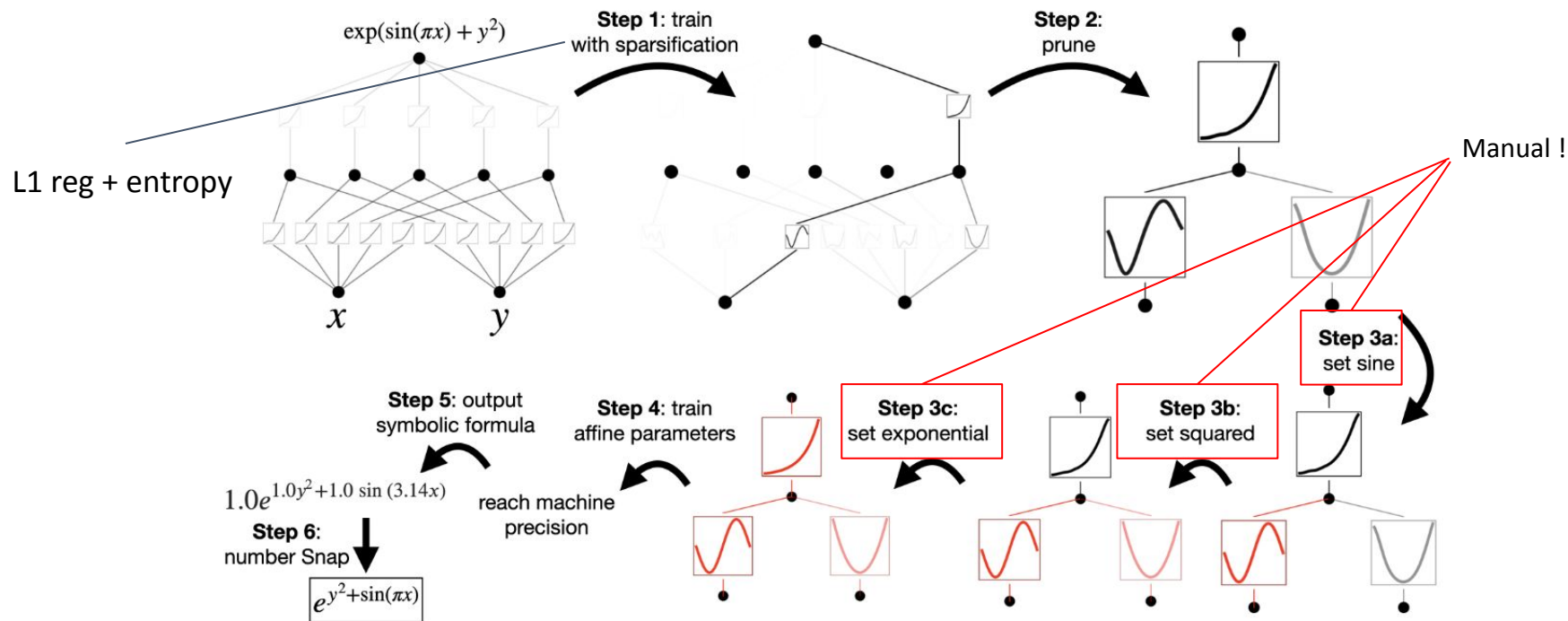


Figure 2.4: An example of how to do symbolic regression with KAN.

Leur image de fin :

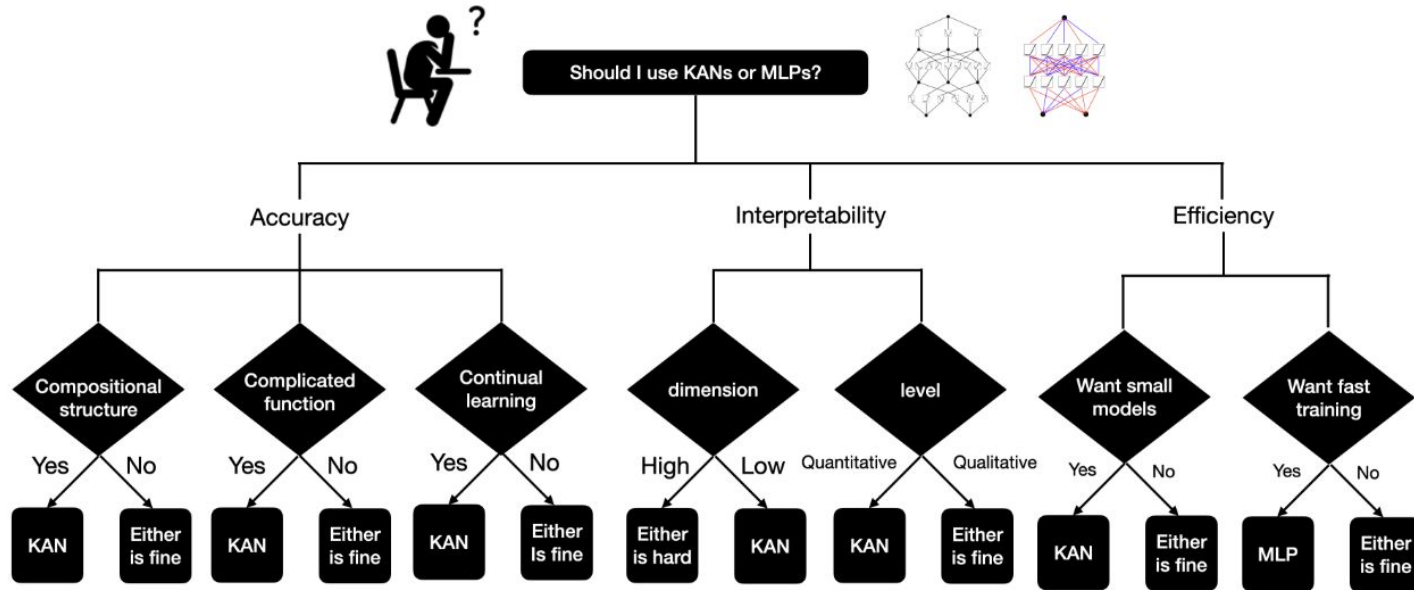


Figure 6.1: Should I use KANs or MLPs?

Ce sont juste des MLP ?

- Beaucoup de gens disent : “De toute façon, ce ne sont que des MLP déguisés”
 - Toutes les fonctions le sont (Approx universelle)
 - Les CNN aussi, c’est pas pour ça qu’on ne fait que des MLP...
 - Oui, c’est “facile” de convertir. Mais la conversion brute est plus lourde (et pas faisable dans le cas symbolique)...
 - On peut aussi convertir des MLP en arbre de décision...

FIN