KAN: Kolmogorov–Arnold Networks

Liu, Ziming, et al. "Kan: Kolmogorov-arnold networks."
arXiv preprint arXiv:2404.19756 (2024).

Presenter:
PhD Student @ CSE, Yonsei Univ.
Jin-Duk Park

Reading Group Material

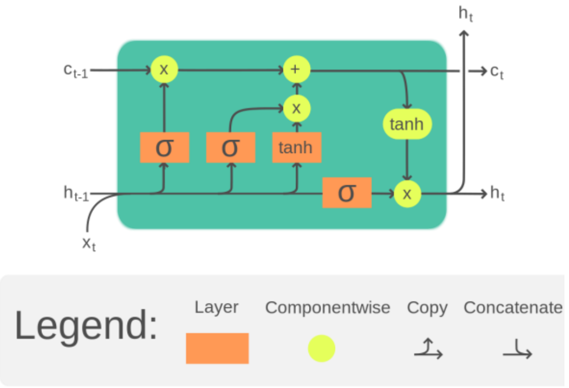
2024.05.29

What Current AI Heavily Relies On

- Modern AI relies on MLP (multi-layer perceptron) architecture

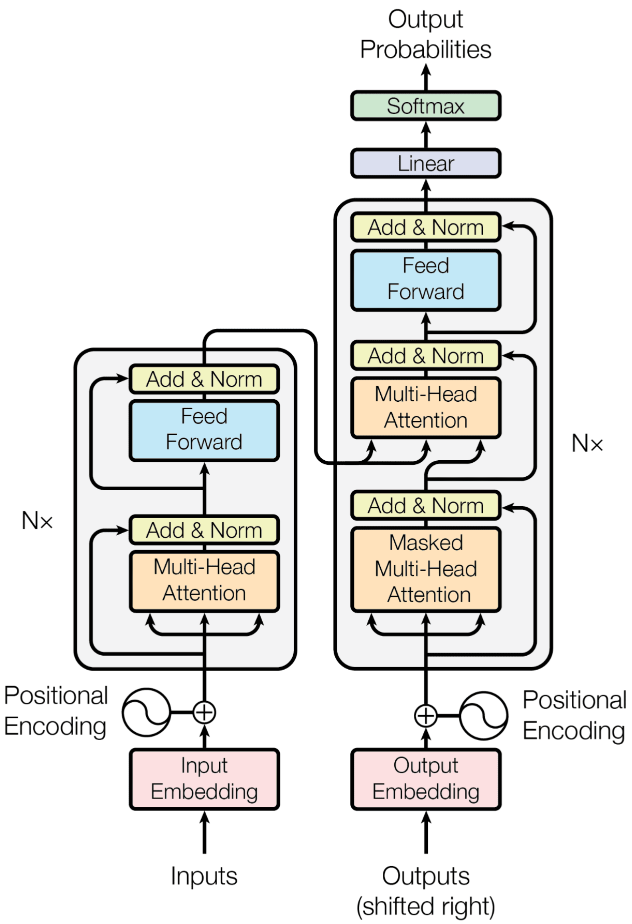
- Transformer, RNN, LSTM, CNN... : based on MLP

- However, are MLPs the best nonlinear regressors we can build?
  - less efficient
  - less interpretable



https://ko.wikipedia.org/wiki/%EC%9E%A5%EB%8B%A8%EA%B8%B0_%EB%A9%94%EB%AA%A8%EB%A6%AC

LSTM

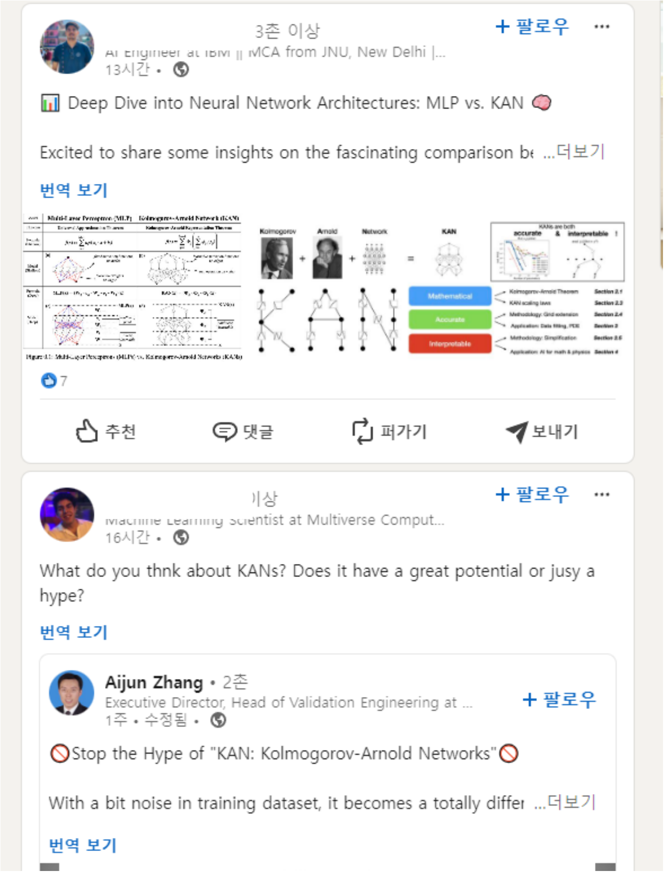


Transformer

Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

KAN: Alternatives to MLPs?

- KAN is proposed as alternatives to MLPs!
- Kolmogorov–Arnold Networks (KAN) is getting lots of attention nowadays!: LinkedIn, X (Twitter), Reddit...



Universal Approximation Theorem (UAT)

- UAT is a theoretical background of modern deep learning
- UAT says that a single MLP layer can approximate any continuous functions

Universal approximation theorem — Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset X of a Euclidean $\mathbb{R}^n$ space to a Euclidean space $\mathbb{R}^m$. Let $\sigma \in C(\mathbb{R}, \mathbb{R})$. Note that $(\sigma \circ x)_i = \sigma(x_i)$, so $\sigma \circ x$ denotes σ applied to each component of x.

Then σ is not polynomial if and only if for every $n \in \mathbb{N}$, $m \in \mathbb{N}$, compact $K \subseteq \mathbb{R}^n$, $f \in C(K, \mathbb{R}^m), \varepsilon > 0$ there exist $k \in \mathbb{N}$, $A \in \mathbb{R}^{k \times n}$, $b \in \mathbb{R}^k$, $C \in \mathbb{R}^{m \times k}$ such that

$$\boxed{\sup_{x \in K} \| f(x) - g(x) \| < \varepsilon}$$

where $g(x) = C \cdot \boxed{(\sigma \circ (A \cdot x + b))}$

https://en.wikipedia.org/wiki/Universal_approximation_theorem

Hornik, Kurt; Stinchcombe, Maxwell; White, Halbert (January 1989). "Multilayer feedforward networks are universal approximators". Neural Networks. 2 (5): 359–366.

Kolmogorov-Arnold Representation Theorem (KAT)

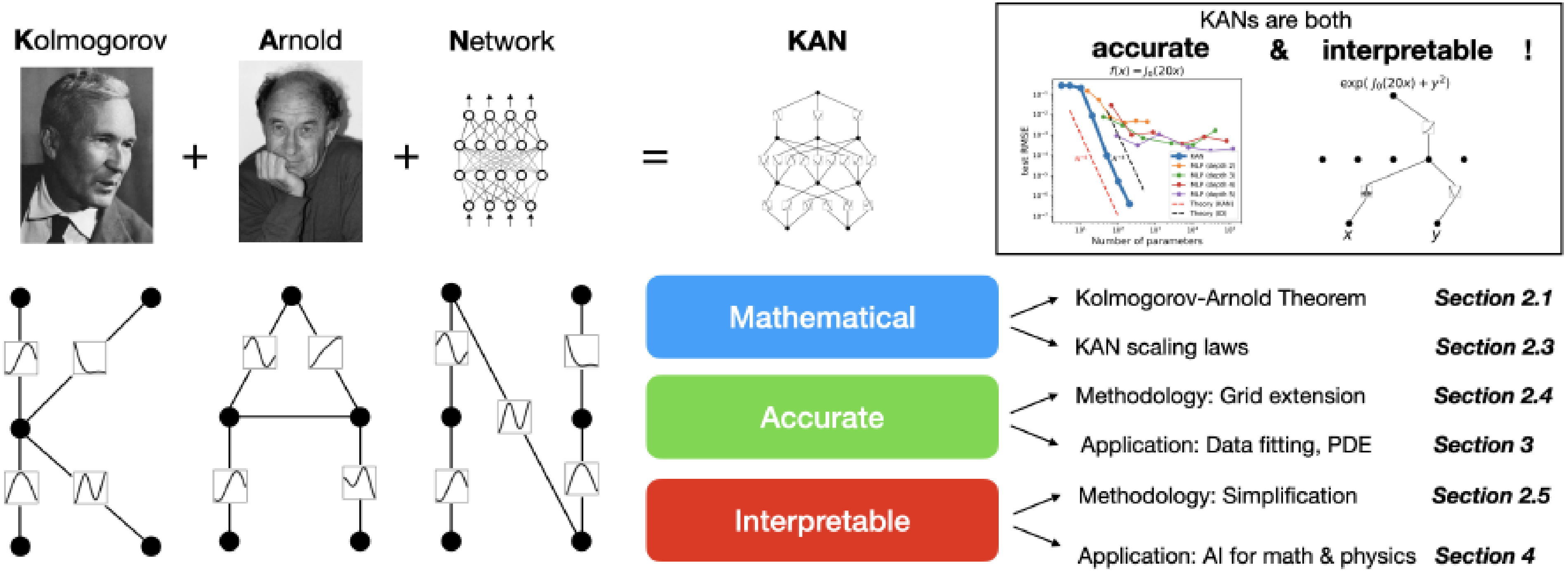
- KAT is a representation theroem
- It says that any continous f can be represented as a finite “composition” of continous functions of a sinlge variable and addition

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

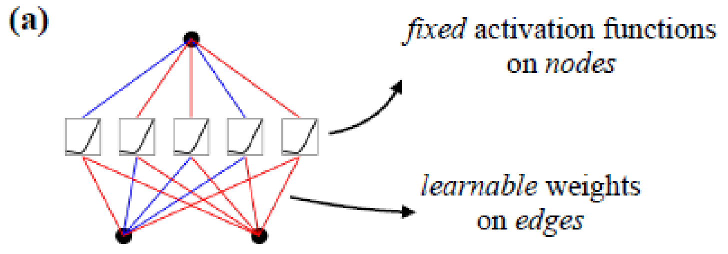
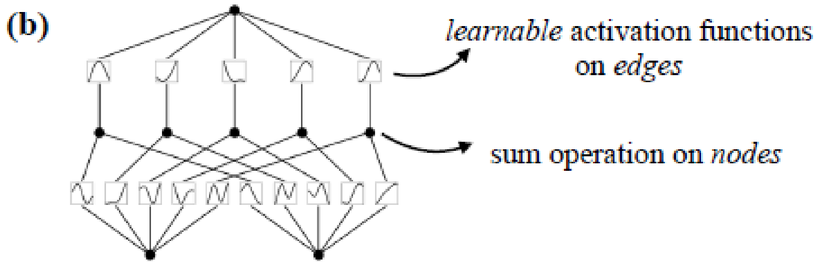
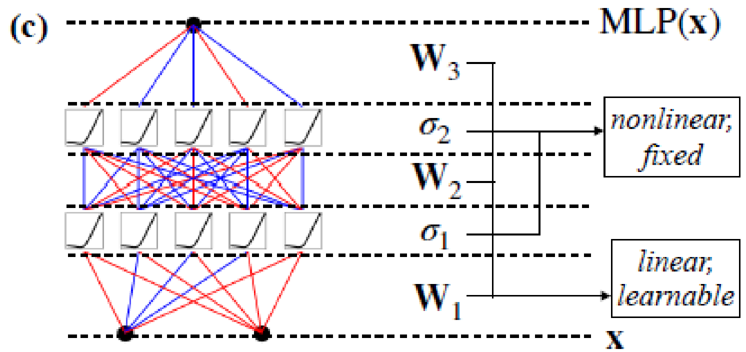
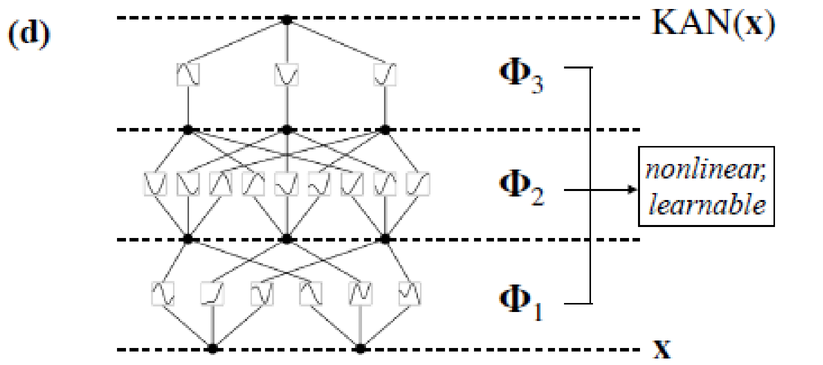
where $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$.

KAN: Overview

- Built upon KAT, KAN is..
  - Mathematical
  - Accurate
  - Interpretable !



KAN: Overview

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

KAN Architecture

- Revisit KAT

- Generalize KAT
-> KAN layer

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

1. 2n+1 width
2. Only two-layer non-linearities

generalize

Arbitrary widths and depths -> KAN

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

KAN Architecture

- KAN layer

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

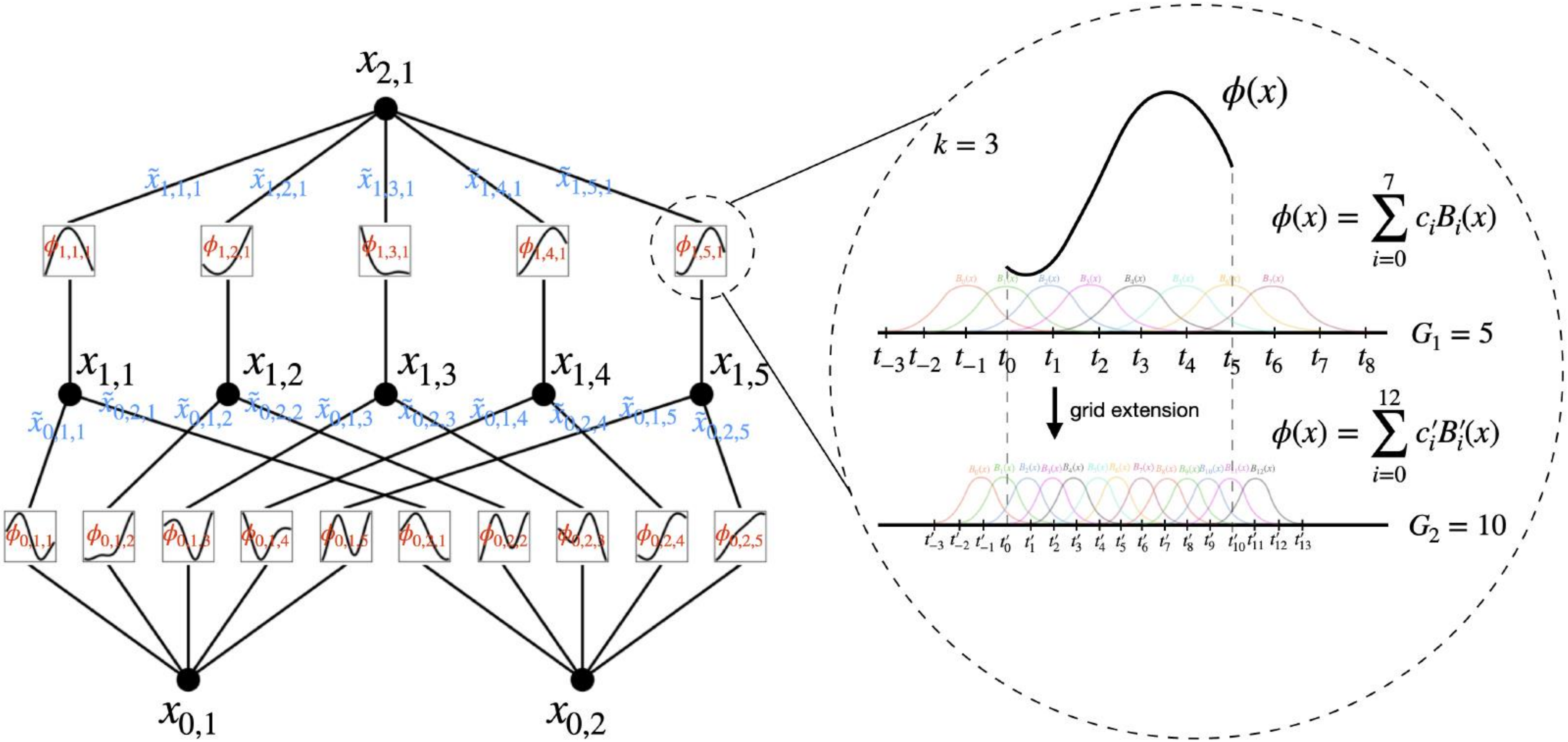
- Deep KAN

$$\boxed{\mathrm{KAN}(\mathbf{x}) = (\mathbf{\Phi}_{L-1} \circ \mathbf{\Phi}_{L-2} \circ \cdots \circ \mathbf{\Phi}_1 \circ \mathbf{\Phi}_0)\mathbf{x}.}$$

$$\Updownarrow$$

$$^{**}\mathrm{MLP}(\mathbf{x}) = (\mathbf{W}_{L-1} \circ \sigma \circ \mathbf{W}_{L-2} \circ \sigma \circ \cdots \circ \mathbf{W}_1 \circ \sigma \circ \mathbf{W}_0)\mathbf{x}.$$

KAN Architecture



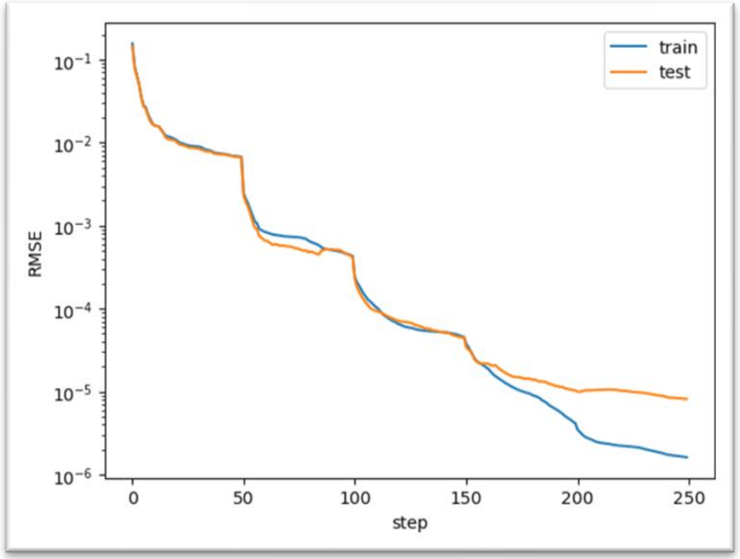
Grid Extention

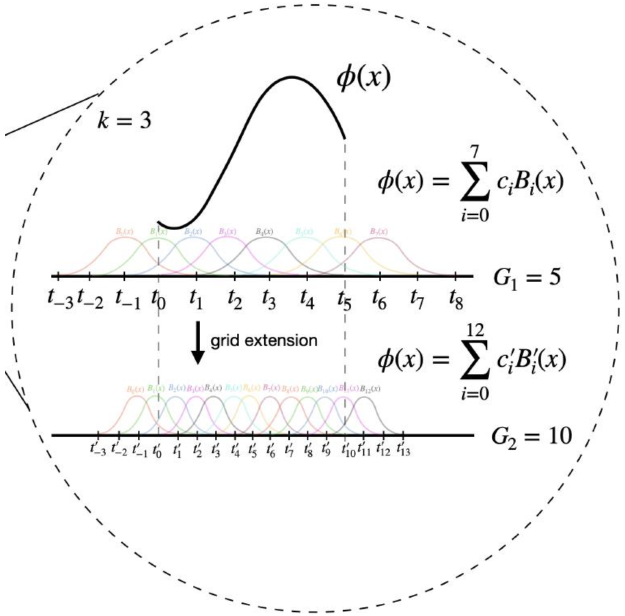
- Neural scaling law: for improving accuracy

- Contrast to MLP,
  KAN: easy to fine-graining
  (w/o retraining larger model)

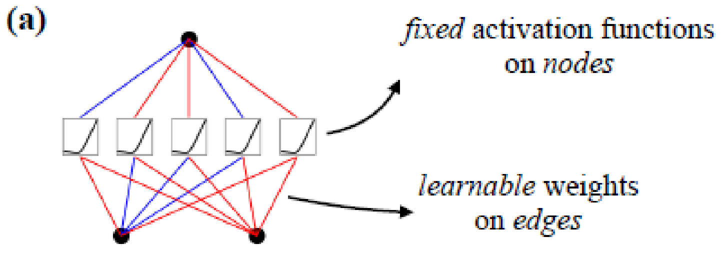
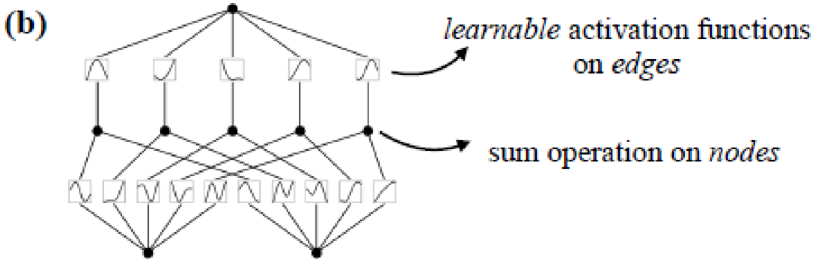
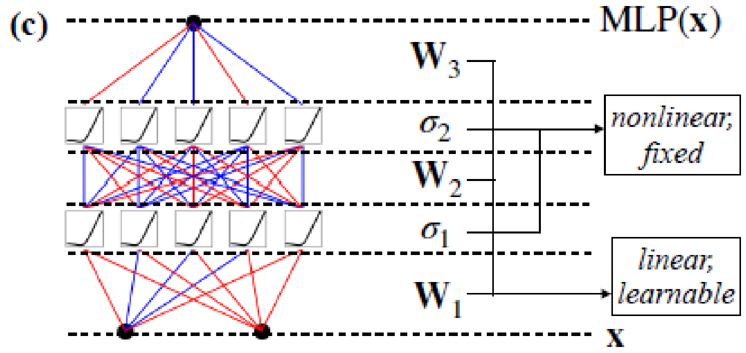
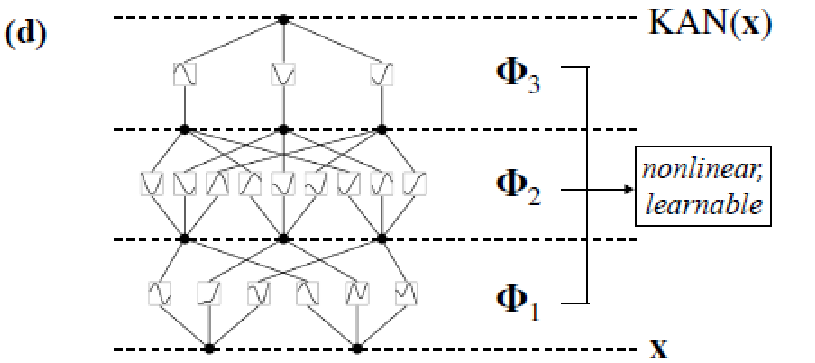
- Coarse grid to fine grid while training

$$\{c'_j\} = \underset{\{c'_j\}}{\operatorname{argmin}} \underset{x\sim p(x)}{\mathbb{E}} \left(\sum_{j=0}^{G_2+k-1} c'_j B'_j(x) - \sum_{i=0}^{G_1+k-1} c_i B_i(x) \right)^2,$$





Comparison: KAN(KAT) vs MLP(UAT)

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

Implementation Details

- Residual activation functions

Scaling factor, trainable

$$\phi(x) = w\,(b(x) + \mathrm{spline}(x))\,.$$

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

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

Trainable

B-splines (basis)

- Initialization

  - $c_i \sim N(0, \sigma^2)$ with small σ (~zero init)
  - $w \sim Xavier$ [1]

[1] Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the thirteenth international conference on artificial intelligence and statistics. JMLR Workshop and Conference Proceedings, 2010.

Sparsification

- Sparse regularization + pruning -> interpretable KAN

- Sparse regularization

$$\ell_{\text{total}} = \ell_{\text{pred}} + \lambda \left(\mu_1 \sum_{l=0}^{L-1} |\Phi_l|_1 + \mu_2 \sum_{l=0}^{L-1} S(\Phi_l) \right)$$

- Pruning (node level)

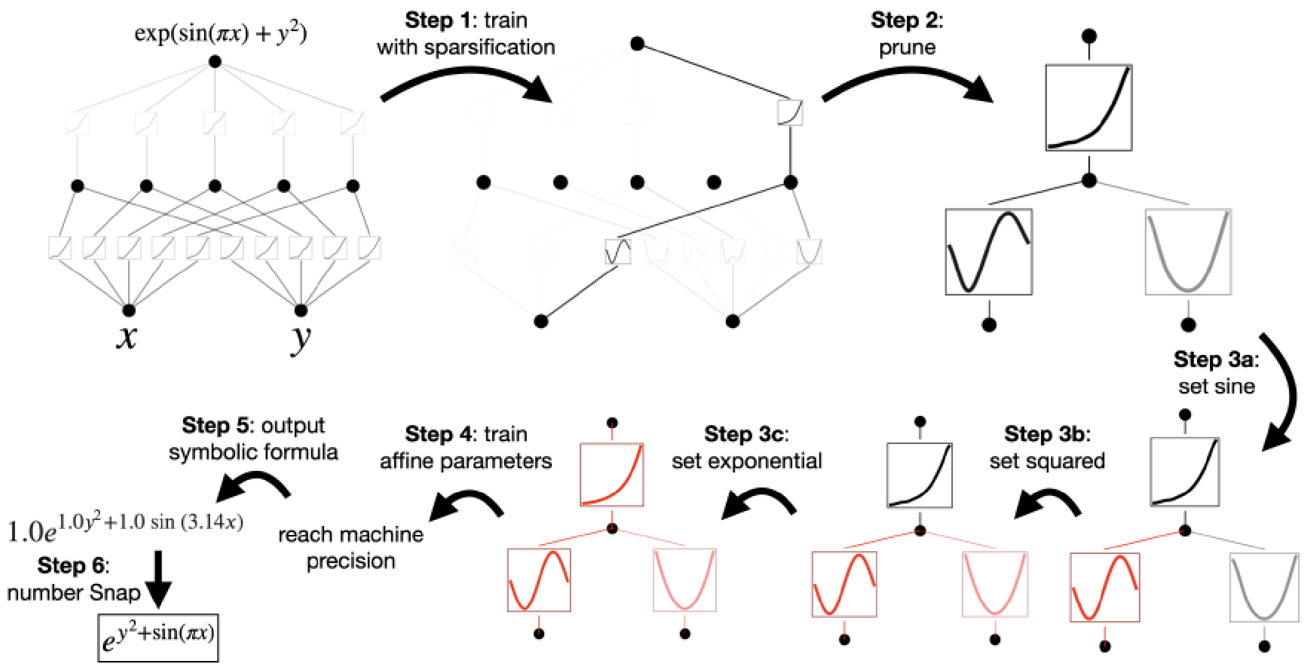
Incoming score

$$I_{l,i} = \max_k(|\phi_{l-1,k,i}|_1),$$

Outgoing score

$$O_{l,i} = \max_j(|\phi_{l+1,j,i}|_1),$$

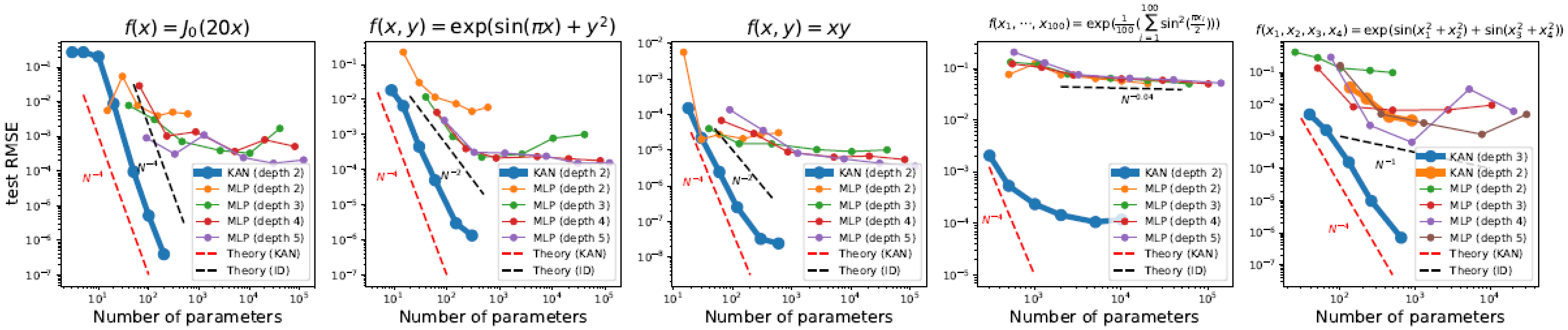
Select important node having
higher incoming/outgoing score



- Symbolification

Sympy to compute symbolic formula of the output node

Toy Example & Special Functions



- KANs can almost saturate the steeper red lines
- While MLPs are not

- Toy example

(1) $\boxed{f(x) = J_0(20x),}$ which is the Bessel function. Since it is a univariate function, it can be represented by a spline, which is a $[1, 1]$ KAN.

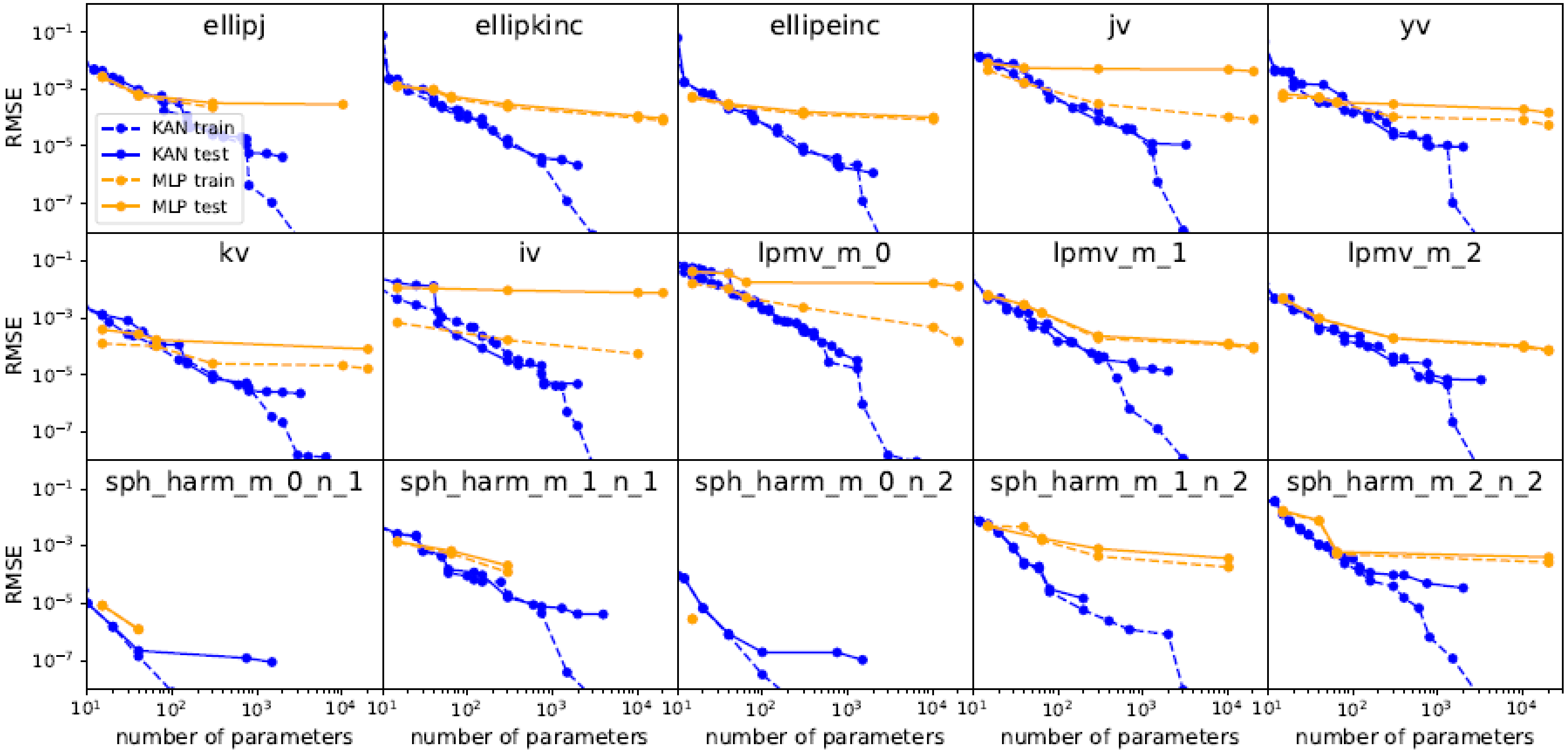
(2) $\boxed{f(x, y) = \exp(\sin(\pi x) + y^2).}$ We know that it can be exactly represented by a $[2, 1, 1]$ KAN.

(3) $\boxed{f(x, y) = xy.}$ We know from Figure 4.1 that it can be exactly represented by a $[2, 2, 1]$ KAN.

(4) A high-dimensional example $\boxed{f(x_1, \cdots, x_{100}) = \exp(\frac{1}{100}\sum_{i=1}^{100}\sin^2(\frac{\pi x_i}{2}))}$ which can be represented by a $[100, 1, 1]$ KAN.

(5) A four-dimensional example $\boxed{f(x_1, x_2, x_3, x_4) = \exp(\frac{1}{2}(\sin(\pi(x_1^2 + x_2^2)) + \sin(\pi(x_3^2 + x_4^2))))}$ which can be represented by a $[4, 4, 2, 1]$ KAN.

Toy Example & Special Functions (Cont'd)



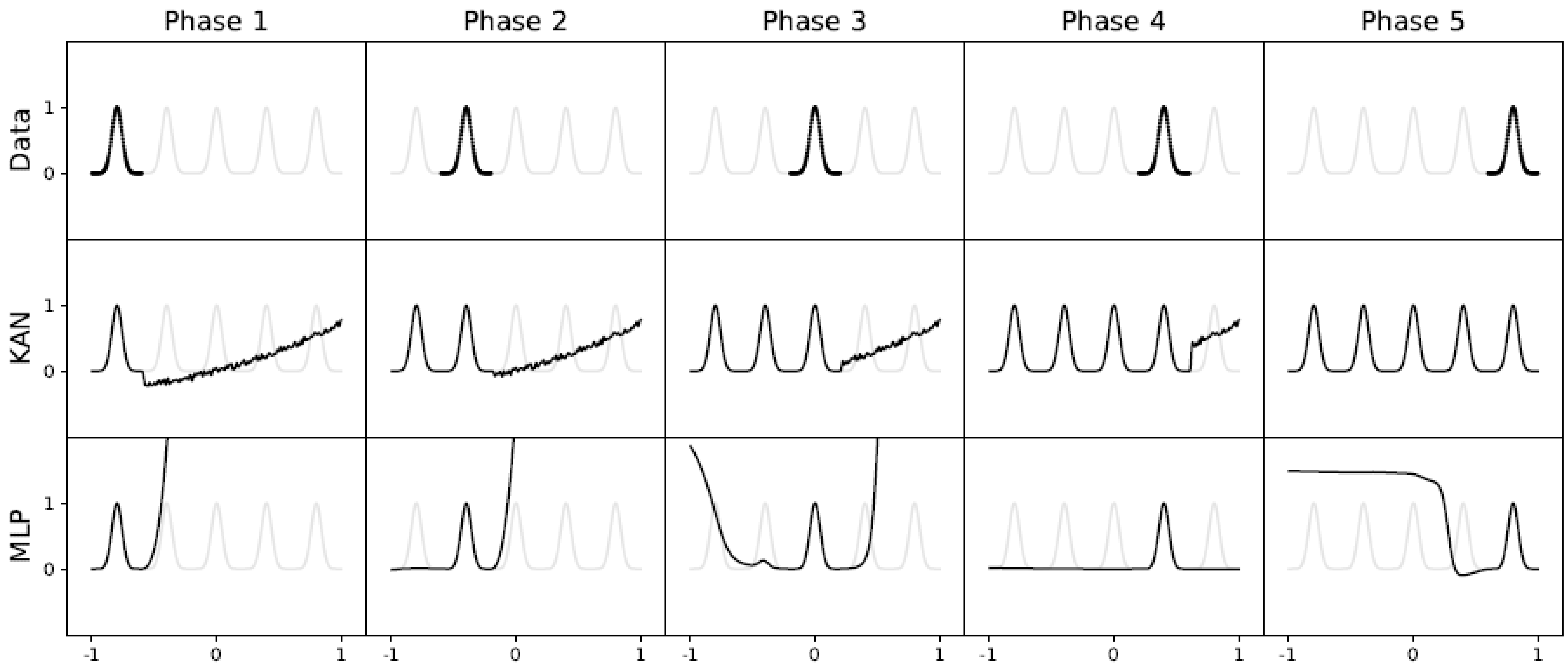
- Performs well than MLPs, on special functions

Feynman Datasets

- Many physics equations collected by Feynman

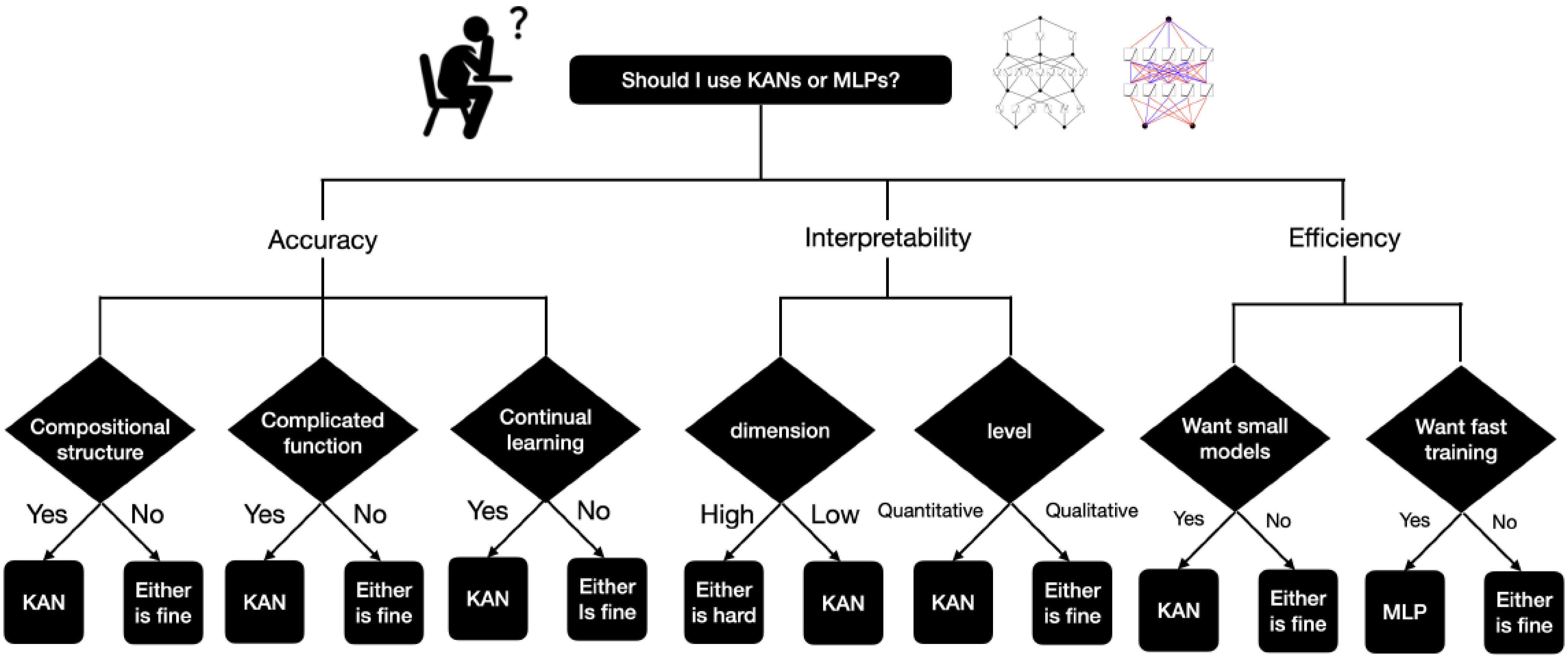
Feynman Eq.	Original Formula	Dimensionless formula	Variables	Human-constructed KAN shape	Pruned KAN shape (smallest shape that achieves RMSE < 10^{-2})	Pruned KAN shape (lowest loss)	Human-constructed KAN loss (lowest test RMSE)	Pruned KAN loss (lowest test RMSE)	Unpruned KAN loss (lowest test RMSE)	MLP loss (lowest test RMSE)
I.6.2	$\exp(-\frac{\theta^2}{2\sigma^2})/\sqrt{2\pi\sigma^2}$	$\exp(-\frac{\theta^2}{2\sigma^2})/\sqrt{2\pi\sigma^2}$	θ, σ	[2,2,1,1]	[2,2,1]	[2,2,1,1]	7.66×10^{-5}	$\mathbf{2.86 \times 10^{-5}}$	4.60×10^{-5}	1.45×10^{-4}
I.6.2b	$\exp(-\frac{(\theta-\theta_1)^2}{2\sigma^2})/\sqrt{2\pi\sigma^2}$	$\exp(-\frac{(\theta-\theta_1)^2}{2\sigma^2})/\sqrt{2\pi\sigma^2}$	θ, θ_1, σ	[3,2,2,1,1]	[3,4,1]	[3,2,2,1,1]	1.22×10^{-3}	$\mathbf{4.45 \times 10^{-4}}$	1.25×10^{-3}	7.40×10^{-4}
I.9.18	$\frac{Gm_1m_2}{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}$	$\frac{a}{(b-1)^2+(c-d)^2+(e-f)^2}$	a, b, c, d, e, f	[6,4,2,1,1]	[6,4,1,1]	[6,4,1,1]	$\mathbf{1.48 \times 10^{-3}}$	8.62×10^{-3}	6.56×10^{-3}	1.59×10^{-3}
I.12.11	$q(E_f + Bv\sin\theta)$	$1 + a\sin\theta$	a, θ	[2,2,2,1]	[2,2,1]	[2,2,1]	2.07×10^{-3}	1.39×10^{-3}	9.13×10^{-4}	$\mathbf{6.71 \times 10^{-4}}$
I.13.12	$Gm_1m_2(\frac{1}{r_2} - \frac{1}{r_1})$	$a(\frac{1}{b} - 1)$	a, b	[2,2,1]	[2,2,1]	[2,2,1]	7.22×10^{-3}	4.81×10^{-3}	2.72×10^{-3}	$\mathbf{1.42 \times 10^{-3}}$
I.15.3x	$\frac{x-ut}{\sqrt{1-(\frac{u}{c})^2}}$	$\frac{1-a}{\sqrt{1-b^2}}$	a, b	[2,2,1,1]	[2,1,1]	[2,2,1,1,1]	7.35×10^{-3}	1.58×10^{-3}	1.14×10^{-3}	$\mathbf{8.54 \times 10^{-4}}$
I.16.6	$\frac{u+v}{1+\frac{uv}{c^2}}$	$\frac{a+b}{1+ab}$	a, b	[2,2,2,2,2,1]	[2,2,1]	[2,2,1]	1.06×10^{-3}	1.19×10^{-3}	1.53×10^{-3}	$\mathbf{6.20 \times 10^{-4}}$
I.18.4	$\frac{m_1r_1+m_2r_2}{m_1+m_2}$	$\frac{1+ab}{1+a}$	a, b	[2,2,2,1,1]	[2,2,1]	[2,2,1]	3.92×10^{-4}	$\mathbf{1.50 \times 10^{-4}}$	1.32×10^{-3}	3.68×10^{-4}
I.26.2	$\arcsin(n\sin\theta_2)$	$\arcsin(n\sin\theta_2)$	n, θ_2	[2,2,2,1,1]	[2,2,1]	[2,2,2,1,1]	1.22×10^{-1}	$\mathbf{7.90 \times 10^{-4}}$	8.63×10^{-4}	1.24×10^{-3}
I.27.6	$\frac{1}{\frac{1}{d_1}+\frac{n}{d_2}}$	$\frac{1}{1+ab}$	a, b	[2,2,1,1]	[2,1,1]	[2,1,1]	2.22×10^{-4}	$\mathbf{1.94 \times 10^{-4}}$	2.14×10^{-4}	2.46×10^{-4}
I.29.16	$\sqrt{x_1^2 + x_2^2 - 2x_1x_2\cos(\theta_1 - \theta_2)}$	$\sqrt{1 + a^2 - 2a\cos(\theta_1 - \theta_2)}$	a, θ_1, θ_2	[3,2,2,3,2,1,1]	[3,2,2,1]	[3,2,3,1]	2.36×10^{-1}	3.99×10^{-3}	$\mathbf{3.20 \times 10^{-3}}$	4.64×10^{-3}
I.30.3	$I_{*,0}\frac{\sin^2(\frac{n\theta}{2})}{\sin^2(\frac{\theta}{2})}$	$\frac{\sin^2(\frac{n\theta}{2})}{\sin^2(\frac{\theta}{2})}$	n, θ	[2,3,2,2,1,1]	[2,4,3,1]	[2,3,2,3,1,1]	3.85×10^{-1}	$\mathbf{1.03 \times 10^{-3}}$	1.11×10^{-2}	1.50×10^{-2}
I.30.5	$\arcsin(\frac{\lambda}{nd})$	$\arcsin(\frac{a}{n})$	a, n	[2,1,1]	[2,1,1]	[2,1,1,1,1,1]	2.23×10^{-4}	$\mathbf{3.49 \times 10^{-5}}$	6.92×10^{-5}	9.45×10^{-5}
I.37.4	$I_* = I_1 + I_2 + 2\sqrt{I_1I_2}\cos\delta$	$1 + a + 2\sqrt{a}\cos\delta$	a, δ	[2,3,2,1]	[2,2,1]	[2,2,1]	7.57×10^{-5}	$\mathbf{4.91 \times 10^{-6}}$	3.41×10^{-4}	5.67×10^{-4}
I.40.1	$n_0\exp(-\frac{mgx}{k_bT})$	n_0e^{-a}	n_0, a	[2,1,1]	[2,2,1]	[2,2,1,1,1,2,1]	3.45×10^{-3}	5.01×10^{-4}	$\mathbf{3.12 \times 10^{-4}}$	3.99×10^{-4}
I.44.4	$nk_bT\ln(\frac{V_2}{V_1})$	$n\ln a$	n, a	[2,2,1]	[2,2,1]	[2,2,1]	$\mathbf{2.30 \times 10^{-5}}$	2.43×10^{-5}	1.10×10^{-4}	3.99×10^{-4}
I.50.26	$x_1(\cos(\omega t) + \alpha\cos^2(\omega t))$	$\cos a + \alpha\cos^2 a$	a, α	[2,2,3,1]	[2,3,1]	[2,3,2,1]	$\mathbf{1.52 \times 10^{-4}}$	5.82×10^{-4}	4.90×10^{-4}	1.53×10^{-3}
II.2.42	$\frac{k(T_2-T_1)A}{d}$	$(a - 1)b$	a, b	[2,2,1]	[2,2,1]	[2,2,2,1]	8.54×10^{-4}	7.22×10^{-4}	1.22×10^{-3}	$\mathbf{1.81 \times 10^{-4}}$
II.6.15a	$\frac{3}{4\pi\epsilon}\frac{p_dz}{r^5}\sqrt{x^2 + y^2}$	$\frac{1}{4\pi}c\sqrt{a^2 + b^2}$	a, b, c	[3,2,2,2,1]	[3,2,1,1]	[3,2,1,1]	2.61×10^{-3}	3.28×10^{-3}	1.35×10^{-3}	$\mathbf{5.92 \times 10^{-4}}$
II.11.7	$n_0(1 + \frac{p_dE_f\cos\theta}{k_bT})$	$n_0(1 + a\cos\theta)$	n_0, a, θ	[3,3,3,2,2,1]	[3,3,1,1]	[3,3,1,1]	7.10×10^{-3}	8.52×10^{-3}	5.03×10^{-3}	$\mathbf{5.92 \times 10^{-4}}$
II.11.27	$\frac{n\alpha}{1-\frac{n\alpha}{3}}\epsilon E_f$	$\frac{n\alpha}{1-\frac{n\alpha}{3}}$	n, α	[2,2,1,2,1]	[2,1,1]	[2,2,1]	2.67×10^{-5}	4.40×10^{-5}	$\mathbf{1.43 \times 10^{-5}}$	7.18×10^{-5}
II.35.18	$\frac{n_0}{\exp(\frac{\mu_mB}{k_bT})+\exp(-\frac{\mu_mB}{k_bT})}$	$\frac{n_0}{\exp(a)+\exp(-a)}$	n_0, a	[2,1,1]	[2,1,1]	[2,1,1,1]	4.13×10^{-4}	1.58×10^{-4}	$\mathbf{7.71 \times 10^{-5}}$	7.92×10^{-5}
II.36.38	$\frac{\mu_mB}{k_bT} + \frac{\mu_m\alpha M}{\epsilon c^2k_bT}$	$a + \alpha b$	a, α, b	[3,3,1]	[3,2,1]	[3,2,1]	2.85×10^{-3}	$\mathbf{1.15 \times 10^{-3}}$	3.03×10^{-3}	2.15×10^{-3}
II.38.3	$\frac{YAx}{d}$	$\frac{a}{b}$	a, b	[2,1,1]	[2,1,1]	[2,2,1,1,1]	1.47×10^{-4}	$\mathbf{8.78 \times 10^{-5}}$	6.43×10^{-4}	5.26×10^{-4}
III.9.52	$\frac{p_dE_f}{h}\frac{\sin^2((\omega-\omega_0)t/2)}{((\omega-\omega_0)t/2)^2}$	$a\frac{\sin^2(\frac{b-c}{2})}{(\frac{b-c}{2})^2}$	a, b, c	[3,2,3,1,1]	[3,3,2,1]	[3,3,2,1,1,1]	4.43×10^{-2}	3.90×10^{-3}	2.11×10^{-2}	$\mathbf{9.07 \times 10^{-4}}$
III.10.19	$\mu_m\sqrt{B_x^2 + B_y^2 + B_z^2}$	$\sqrt{1 + a^2 + b^2}$	a, b	[2,1,1]	[2,1,1]	[2,1,2,1]	2.54×10^{-3}	1.18×10^{-3}	8.16×10^{-4}	$\mathbf{1.67 \times 10^{-4}}$
III.17.37	$\beta(1 + \alpha\cos\theta)$	$\beta(1 + \alpha\cos\theta)$	α, β, θ	[3,3,3,2,2,1]	[3,3,1]	[3,3,1]	1.10×10^{-3}	5.03×10^{-4}	$\mathbf{4.12 \times 10^{-4}}$	6.80×10^{-4}

Continual Learning



- Toy example dataset
- KAN is more immune to catastropic forgetting
- Structure re-organization only occurs locally on KAN, while MLP is not

Should I use KANs or MLPs?



Takeaway Messages

- KAN can be a potential alternatives to MLP
- However, KAN is only applied to limited dataset & settings

  - Still, a lot of research projects are now going on though… ☺
  (refer to: https://github.com/mintisan/awesome-kan)

Library

- pykan : Offical implementation for Kolmogorov Arnold Networks | Stars 12k
- efficient-kan : An efficient pure-PyTorch implementation of Kolmogorov-Arnold Network (KAN). | Stars 2.8k
- FastKAN : Very Fast Calculation of Kolmogorov-Arnold Networks (KAN) | Stars 194
- FasterKAN : FasterKAN = FastKAN + RSWAF bases functions and benchmarking with other KANs. Fastest KAN variation as of 5/13/2024, 2 times slower than MLP in backward speed. | Stars 18
- TorchKAN : Simplified KAN Model Using Legendre approximations and Monomial basis functions for Image Classification for MNIST. Achieves 99.5% on MNIST using Conv+LegendreKAN. | Stars 68
- FourierKAN : Pytorch Layer for FourierKAN. It is a layer intended to be a substitution for Linear + non-linear activation | Stars 559
- ChebyKAN : Kolmogorov-Arnold Networks (KAN) using Chebyshev polynomials instead of B-splines. | Stars 241
- GraphKAN : Implementation of Graph Neural Network version of Kolmogorov Arnold Networks (GraphKAN) | Stars 173
- FCN-KAN : Kolmogorov–Arnold Networks with modified activation (using fully connected network to represent the activation) | Stars 84
- X-KANeRF : KAN based NeRF with various basis functions like B-Splines, Fourier, Radial Basis Functions, Polynomials, etc | Stars 84
- Large Kolmogorov-Arnold Networks : Variations of Kolmogorov-Arnold Networks (including CUDA-supported KAN convolutions) | Stars 96
- FastKAN : Very Fast Calculation of Kolmogorov-Arnold Networks (KAN) | Stars 194
- xKAN : Kolmogorov-Arnold Networks with various basis functions like B-Splines, Fourier, Chebyshev, Wavelets etc

Project

- KAN-GPT : The PyTorch implementation of Generative Pre-trained Transformers (GPTs) using Kolmogorov-Arnold Networks (KANs) for language modeling | Stars 594
- KAN-GPT-2 : Training small GPT-2 style models using Kolmogorov-Arnold networks.(despite the KAN model having 25% fewer parameters!). | Stars 72
- KANeRF : Kolmogorov-Arnold Network (KAN) based NeRF | Stars 118
- Vision-KAN : KAN for Vision Transformer | Stars 54
- Simple-KAN-4-Time-Series : A simple feature-based time series classifier using Kolmogorov–Arnold Networks | Stars 66
- KANU_Net : U-Net architecture with Kolmogorov-Arnold Convolutions (KA convolutions) | Stars 9
- kanrl : Kolmogorov-Arnold Network for Reinforcement Leaning, initial experiments | Stars 199
- kan-diffusion : Applying KANs to Denoising Diffusion Models with two-layer KAN able to restore images almost as good as 4-layer MLP (and 30% less parameters). | Stars 17
- KAN4Rec : Implementation of Kolmogorov-Arnold Network (KAN) for Recommendations | Stars 25
- CF-KAN : Kolmogorov-Arnold Network (KAN) implementation for collaborative filtering (CF) | Stars 8
- X-KANeRF : X-KANeRF: KAN-based NeRF with Various Basis Functions to explain the the NeRF formula | Stars 84
- KAN4Graph : Implementation of Kolmogorov-Arnold Network (KAN) for Graph Neural Networks (GNNs) and Tasks on Graphs | Stars 33

It's mine! ☺

"Success is not final, failure is not fatal:
it is the courage to continue that counts."
- Winston Churchill

Contact: jindeok6@yonsei.ac.kr
Web Page: https://jindeok.github.io/jdpark/