KAN: Kolmogorov–Arnold Networks: A review

Vikas Dhiman

May 8, 2024

Why review this? On Apr 30, 2024, [Liu et al., 2024] appears on ArXiV and by May 7th, I have heard about this paper from multiple students, from whom I do not hear about new papers. It must be special, I thought. I decided to take a look.

If I was professionally reviewing this paper, I would accept the paper with major revisions. The paper has enough contributions to deserve a publication. But some of the claims need to be toned down, interpretations need to be clarified and comparisons with spline based neural networks be made.

Outline I make 4 major critques of the paper

1. MLPs have learnable activation functions as well
2. The content of the paper does not justify the name, Kolmogorov-Arnold networks (KANs).
3. KANs are MLPs with spline-basis as the activation function.
4. KANs do not beat the curse of dimensionality.

MLPs have learnable activation functions as well The authors claim in the abstract,

> While MLPs have fixed activation functions on nodes ("neurons"), KANs have learnable activation functions on edges ("weights"). KANs have no linear weights at all – every weight parameter is replaced by a univariate function parametrized as a spline.

This is not a helpful description because one can interpret MLPs as having "learnable activation functions" as well; it depends on the definition what you call the "activation function". Consider a two layer MLP with input $\mathbf{x} \in \mathbb{R}^n$, weights W_1, W_2 (ignore biases for now) and activation function σ,

$$f(\mathbf{x}) = W_2\sigma(W_1\mathbf{x}) = W_2\phi_1(\mathbf{x}). \tag{1}$$

If I define $\phi_1(\mathbf{x}) = \sigma(W_1\mathbf{x})$ and call $\phi_1(.)$ as the activation function, then I have a learnable activation function in an MLP. Same with Figure 0.1, it is a *reinterpretation*, not *redesign* of MLPs as claimed.

What's in the name How do KAN's actually use Kolmogorov-Arnold Theorem (KAT)? The theorem is not actually useful in the development of KANs. KANs are only inspired by KAT not based on it.

So what is Kolmogorov-Arnold Theorem? The paper describes it as the decomposition of any smooth function $f : [0, 1]^n \to \mathbb{R}$ in terms of finite basis function $\phi_q^{(2)} : \mathbb{R} \to \mathbb{R}^1$ and $\phi_{q,p} : [0, 1] \to \mathbb{R}$.

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

If you plan to use Kolmogorov-Arnold Theorem (KAT), you have understand the central claim of the KAT theorem and how is this theorem different the nearest competitor (Universal Approximation Theorem

[1] Slight change in notation from the paper

(UAT)). Universal approximation theorem states that any function can be approximated by a wide enough 2-layer neural network.

$$f(\mathbf{x}) = \sum_{q=1}^{\infty} w_q^{(2)} \sigma \left(\sum_{p=1}^{n} w_{q,p}^{(1)} x_p \right) \qquad \text{where } W_2 = [w_q^{(2)}]_{q=1}^{\infty} \text{ and } W_1 = [[w_{q,p}^{(1)}]_{q=1}^{\infty}]_{p=1}^{n} \tag{3}$$

I wrote the MLP in terms of summation instead of matrix multiplication to draw parallels between UAT and KAT. There are two main differences between UAT and KAT,

1. UAT deals with linear layers with common activation function (like sigmoid [Cybenko, 1989], ReLU, tanh) while KAT deals with arbitrary functions, possibly "non-smooth and even fractal".
2. UAT needs possibly infinite hidden units for exact approximation while KAT only needs $2n+1$ "hidden units".

I would claim that central point of KAT is about needing only $2n+1$ hidden units, otherwise it is a weaker theorem than UAT. Does the KAN paper make use of the $2n+1$ hidden units consistently? No. But they justify rest of the paper to be based on KAT by saying,

> However, we are more optimistic about the usefulness of the Kolmogorov-Arnold theorem for machine learning. First of all, we need not stick to the original Eq. (2.1) which has only two-layer non- linearities and a small number of terms (2n + 1) in the hidden layer: we will generalize the network to arbitrary widths and depths.

Okay. But aren't we back to Universal approximation theorem then?

There is one aspect of KAT that the authors highlight, "In a sense, they showed that the only true multivariate function is addition, since every other function can be written using univariate functions and sum." This is a cool interpretation, but this interpretation does not separate KAT from UAT that is already being used in MLPs.

KANs are MLPs with spline-based activation functions In practice, the authors end up proposing a KAN residual layer whose each scalar function is written as,

$$\phi(x) = w(\text{silu}(x) + \text{spline}(x)) \tag{4}$$

$$\text{spline}(x) = \sum_{i=1}^{G} c_i B_i(x). \tag{5}$$

What are splines?[2] For the purpose of this section you do not need to know splines. By the way there exists a history of splines in neural networks which is not cited in this paper [Bohra et al., 2020, Aziznejad et al., 2020]. For now, assume splines are functions that are a result of a linear combination $c_i B_i(x)$ of a particular kind basis functions $B_i(x)$ (B-form splines). To reinterpret this scalar function as a MLP, let's rewrite this as,

$$\phi(x) = \sum_{i=1}^{k} c_i B_i(x) + \text{selu}(x) \tag{6}$$

$$= \underbrace{\begin{bmatrix} wc_1 & wc_2 & \dots & wc_k & w \end{bmatrix}}_{\mathbf{w}^\top} \underbrace{\begin{bmatrix} B_1(x) \\ B_2(x) \\ \vdots \\ B_G(x) \\ \text{selu}(x) \end{bmatrix}}_{\mathbf{b}(x)} \tag{7}$$

$$= \mathbf{w}^\top \mathbf{b}(x). \tag{8}$$

[2] https://personal.math.vt.edu/embree/math5466/lecture10.pdf

Here $\mathbf{w}$ contains the learnable parameters of the splines and $\mathbf{b}(x)$ is deterministic once the spline grid is fixed though it can be made learnable. Let's put this back into (2),

$$f(\mathbf{x}) = f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} \phi_q^{(2)} \left(\sum_{p=1}^{n} \phi_{p,q}^{(1)}(x_p) \right) \tag{9}$$

$$= \sum_{q=1}^{2n+1} \mathbf{w}_q^{(2)\top} \mathbf{b} \left(\sum_{p=1}^{n} \mathbf{w}_{p,q}^{(1)\top} \mathbf{b}(x_p) \right). \tag{10}$$

This is very close to an MLP if we consider $\mathbf{w}$s linear weights and the basis functions as activation functions, with the following differences,

1. The activation function $\mathbf{b}()$ is applied on the input side which is typically not a part of MLPs. However, it has been common to convert input into a set of feature vectors as a pre-processing step rather than providing MLPs the raw input.

2. Unlike $w_{p,q}^{(1)}$ being scalar in (3), $\mathbf{w}_{p,q}^{(1)}$ is a vector in (10). This is not a problem because it is still a linear combination of processed input values through a basis function $\mathbf{b}(x)$. To make this explicit, we write (10) as a matrix vector multiplication followed by activation functions.

To write (10) as a matrix vector product, consider only the first layer term ,

$$\sum_{p=1}^{n} \mathbf{w}_{p,q}^{(1)\top} \mathbf{b}(x_p) = \underbrace{\begin{bmatrix} \mathbf{w}_{1,1}^{(1)\top} & \cdots & \mathbf{w}_{1,n}^{(1)\top} \\ \mathbf{w}_{2,1}^{(1)\top} & \cdots & \mathbf{w}_{2,n}^{(1)\top} \\ \vdots & \ddots & \vdots \\ \mathbf{w}_{2n+1,1}^{(1)\top} & \cdots & \mathbf{w}_{2n+1,n}^{(1)\top} \end{bmatrix}_{(2n+1)\times nG}}_{\mathbf{W}^{(1)}} \underbrace{\begin{bmatrix} \mathbf{b}(x_1) \\ \vdots \\ \mathbf{b}(x_n) \end{bmatrix}_{nG\times 1}}_{\mathbf{B}(\mathbf{x})}. \tag{11}$$

You can apply this interpretation repeatedly,

$$f(\mathbf{x}) = \mathbf{W}^{(2)} \mathbf{B} \left(\mathbf{W}^{(1)} \mathbf{B}(\mathbf{x}) \right), \tag{12}$$

where $\mathbf{x} \in \mathbb{R}^n, \mathbf{B} : \mathbb{R}^n \to \mathbb{R}^{nG}, \mathbf{W}^{(1)} \in \mathbb{R}^{(2n+1)\times nG}, \mathbf{W}^{(2)} \in \mathbb{R}^{1\times(2n+1)G}$.

Here $\mathbf{B}(\mathbf{x})$ is unlike other activation functions. Instead of producing a scalar from a scalar, it produces G different values for each scalar value in the input.

The claim that KANs beat the curse of dimensionality is wrong The authors claim that,

> KANs with finite grid size can approximate the function well with a residue rate independent of the dimension, hence beating curse of dimensionality!

This is a huge claim and requires huge evidence. As outlined in the previous section, if all KANs can be written as MLPs then either both MLPs and KANs beat the curse of dimensionality or neither does.

My first objection is how "curse of dimensionality" is interpreted. Typically curse of dimensionality in machine learning is measured by the amount of data needed for training a function to a desired error.

I do not understand the proof of Theorem 2.1, especially the first step. It is not clear from what theorem in [de Boor, 2001] the first result follows. If page number or chapter is provided that would be great.

It is also counter intuitive because a singular grid size G is assumed for all the n input dimensions. What would the bound look like if each dimension of x is divided into different grid sizes.

References

[Aziznejad et al., 2020] Aziznejad, S., Gupta, H., Campos, J., and Unser, M. (2020). Deep neural networks with trainable activations and controlled lipschitz constant. *IEEE Transactions on Signal Processing*, 68:4688–4699.

[Bohra et al., 2020] Bohra, P., Campos, J., Gupta, H., Aziznejad, S., and Unser, M. (2020). Learning activation functions in deep (spline) neural networks. *IEEE Open Journal of Signal Processing*, 1:295–309.

[Cybenko, 1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314.

[de Boor, 2001] de Boor, C. (2001). *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer New York.

[Liu et al., 2024] Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. (2024). Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*.