Kolmogorov–Arnold Networks

VU Anh Thu LE Thi Minh Nguyet

Sorbonne Université

Introduction

The well-known MLP is powerful and widely used due to their expressive power guaranteed by the universal approximation theorem. However, they often function as black-box models, lacking interpretability. The paper [1] introduces Kolmogorov-Arnold Networks (KAN), a novel deep learning architecture that extends the Kolmogorov-Arnold representation theorem to arbitrary widths and depths. This generalization enables KAN to approximate non-smooth functions with impressive accuracy, while maintaining a reasonable level of interpretability, making it a potential alternative to MLP.

Key points of the network

KAN is better in compare to the common MLP in:

- Interpretability: Each connection is a function, making it easier to analyze.
- Higher accuracy: Achieves better performance compared to MLP in many tasks.

KAN is, however:

- Slow training due to recursive computation of order-k splines.

Network architecture

KAN has a layered structure similar to MLP, but unlike MLP, KAN places *learnable* activation functions on edges. Each learnable weight in an MLP is replaced by a learnable $1D$ function in a KAN. KAN's nodes simply sum incoming signals without applying any non-linearities.

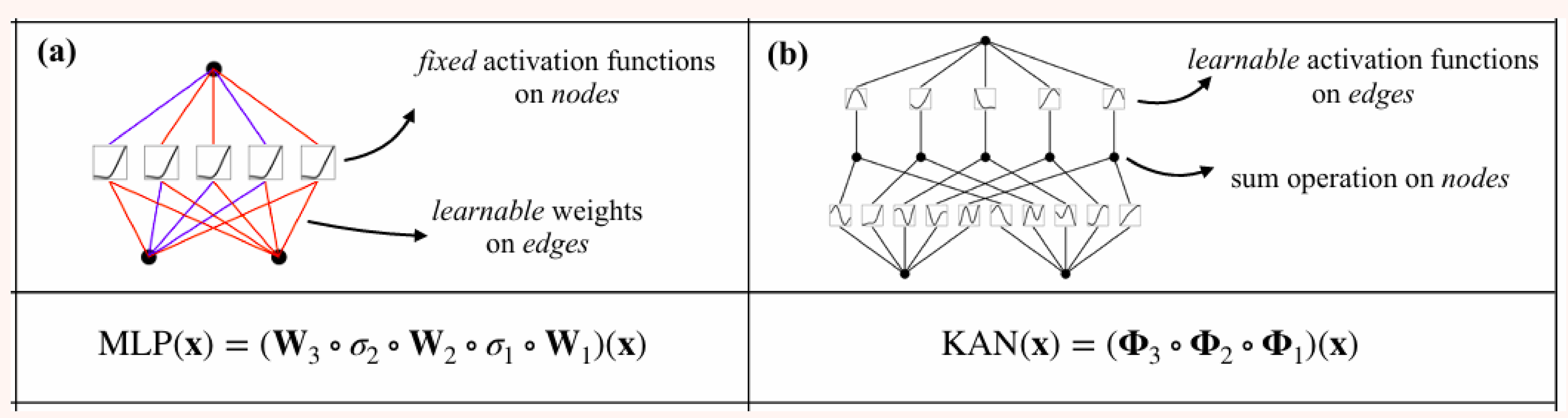

Figure 1. Comparison between MLP and KAN architectures

Specifically, each *learnable* activation function is of the form:

$$\phi(x) = w_b b(x) + w_s spline(x)$$

where $b(x)$ is a basis function (e.g. SiLU) that acts like a residual connection, and $spline(x)$ is parameterized as a linear combination of B-splines such that

$$spline(x) = \sum_i c_i B_i(x)$$

Here, w_b, w_s, and c_is are trainable and will be optimized using backpropagation.

Each activation function is initialized with $w_s = \frac{1}{\sqrt{\texttt{input\_dimension}}}$ and $spline(x) \approx 0$, while w_b is initialized according to the Xavier initialization.

Techniques

Here we present some tricks and techniques that can be used to improve the network's performance:

1. **Grid update**: This technique helps to improve accuracy by updating the grid range on the fly based on the statistics of input/activation ranges during training.
2. **Regularization**: Since the edges of KAN are functions, regularization is applied to the nodes. The L1 norm of an activation function is defined as its average magnitude over sampled inputs, i.e.,

$$|\phi|_1 = \frac{1}{N_p}\sum_{s=1}^{N_p}\left|\phi\left(x^{(s)}\right)\right|$$

For a KAN layer, the total L1 norm is the sum of all activation functions' norms, i.e.,

$$|\Phi|_1 = \sum_{i=1}^{n_{\text{in}}}\sum_{j=1}^{n_{\text{out}}}|\phi_{i,j}|_1$$

It is noted by the author that L1 regularization is insufficient for the sparsification of KANs. An entropy loss is thus introduced,

$$S(\mathbf{\Phi}) = \sum_{i=1}^{n_{\text{in}}}\sum_{j=1}^{n_{\text{out}}}\frac{|\phi_{i,j}|_1}{|\mathbf{\Phi}|_1}\log\left(\frac{|\phi_{i,j}|_1}{|\mathbf{\Phi}|_1}\right)$$

Results

We do the following experiments on KAN:

- Function $f(x,y) = xy$: To prove the necessity of grid update, we train KAN with and without grid update. Both trainings are run with 5 different seeds and use regularization with a coefficient of 0.01. For the case with grid update, the grid is updated three times uniformly from the first epoch to epoch 25. The result is as follows:

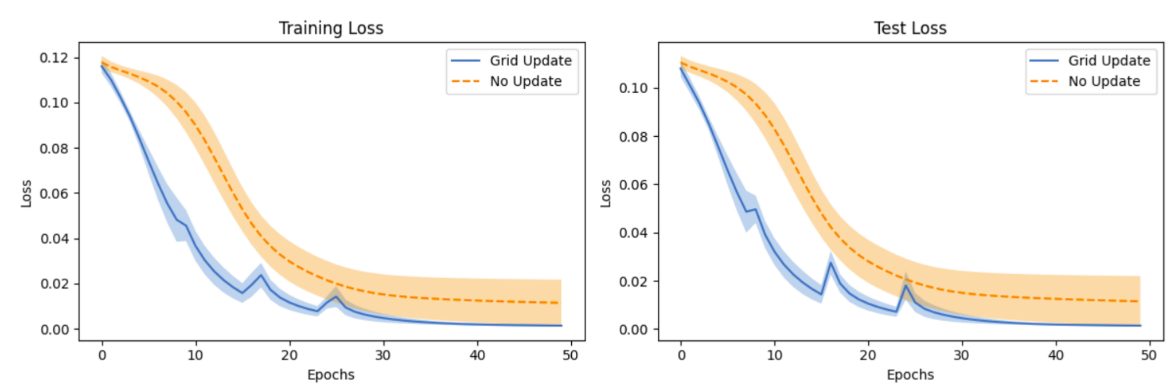

Figure 2. Fitting KAN ($[2,5,1]$, $G=3, k=2$) on $f(x,y)=xy$ using Adam with $\texttt{lr} = 5\times10^{-4}$ over 50 epochs

- Function $f(x,y) = \exp(\sin(\pi x) + y^2)$: We compare the performance of KAN and MLP with the same depth and number of parameters (120 for KAN and 121 for MLP). Both trainings are run with 5 different seeds. The result is as follows:

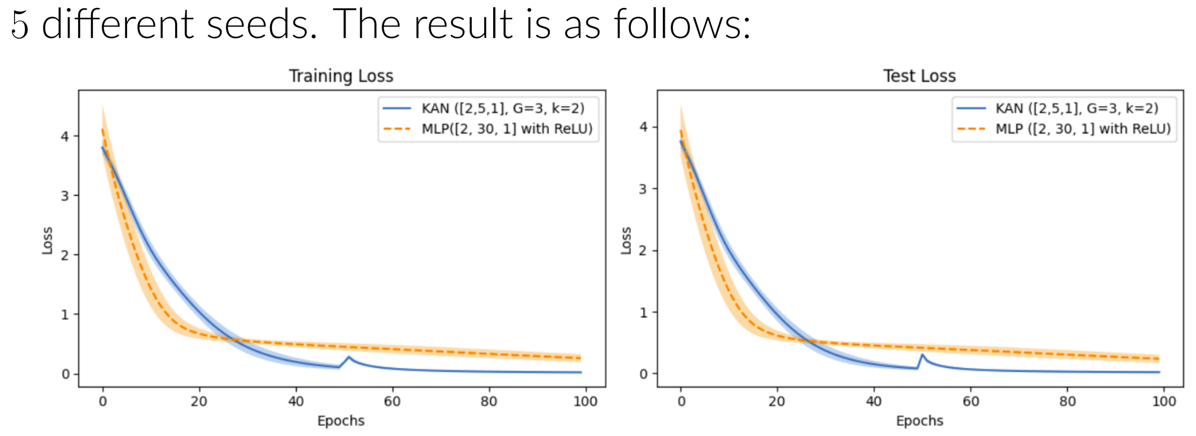

Figure 3. KAN and MLP loss on fitting $f(x,y) = \exp(\sin(\pi x) + y^2)$ using Adam with $\texttt{lr} = 5\times10^{-4}$ over 100 epochs

- Image fitting: We train KAN for the image fitting task on two different images: the Cameraman picture and Van Gogh's *The Starry Night*. The architecture is the same for both cases, with a width of $[2, 128, 128, 128, 128, 1]$, a grid size of 10, and a spline order of 3. We also use Adam optimizer with $\texttt{lr} = 10^{-3}$. The model is trained for 1000 epochs on the Cameraman picture and 500 epochs on *The Starry Night*. We then compare its performance to an MLP with the same number of parameters (results for the MLP are taken from [1]).

Image fitting on Cameraman picture

Figure 4. KAN

Figure 5. MLP

Image fitting on Van Gogh's *The Starry Night*

Figure 6. KAN

Figure 7. MLP

- MNIST: To test KAN' scalability for high-dimensional datasets, we train it on MNIST for a classification task. The KAN architecture used has a width of $[28\times28, 10, 10]$, a grid size of 3, and a spine order of 3. The model is trained over 20 epochs, using AdamW optimizer with $\texttt{lr} = 10^{-3}$. We then compare its performance to an MLP with the same architecture (results for the MLP are taken from [1]).

Model	Train Loss	Test Loss	Train Accuracy	Test Accuracy
MLP	2.3×10^{-1}	2.8×10^{-1}	93.7%	92.5%
KAN	1.40×10^{-1}	2.37×10^{-1}	95.9%	93.5%

Table 1. Comparison of MLP and KAN performance.

References

[1] Ziming Liu et al. KAN: Kolmogorov–Arnold Networks. *arXiv:2404.19756v4*, 2024.