Generalized Kolmogorov-Arnold Network for Implicit Neural Representation

Yisi Luo

***Abstract*—The main aim of this letter is to study the potential of Kolmogorov-Arnold network (KAN) for implicit neural representation (INR). We first show that KAN can be reformulated as a special type of MLP with multiple activation functions performed on inputs after repetition. Inspired by this representation, we propose a generalized KAN (GKAN) that employs multiple sinusoidal activation functions with different frequencies to continuously represent signals under the INR framework. Through analysis from the neural tangent kernel and Lipschitz smooth perspectives, we reveal the underlying merits of GKAN as compared with traditional MLP structures for INR. Numerical experiments on image, video, point cloud, and neural radiance field representations showcase the effectiveness of the proposed GKAN for continuously representing signals.**

***Index Terms*—Kolmogorov-Arnold network, implicit neural representation.**

I. Introduction

Recently, Kolmogorov-Arnold network (KAN) [1] has emerged as an alternative to traditional MLPs. The structure of KAN mainly consists of learnable activation functions parameterized by basis functions with learnable linear weights. It was shown that [1] KAN is more capable of dealing with (re)discovering mathematical and physical laws with the help of the compositional structure of learnable activation functions. However, the effectiveness and efficiency of the classical KAN for fitting/processing more sophisticated signals such as multi-dimensional image and point cloud can be further improved [2], [3], which leaves an open area for researchers concerning these aspects.

In this letter, we study the potential of KAN for implicit neural representation (INR) [4]–[6]. The INR refers to continuously representing a signal with a deep neural network (DNN), where the DNN takes a multi-dimensional coordinate vector as input and is expected to output the corresponding value at that coordinate. Previous works focused on designing effective positional encoding, activation function, or decomposition strategies to facilitate INRs [4], [5], [7]–[9], all based on traditional MLPs. Differently, the motivation of this work is to utilize the KAN with a generalized structure to replace the traditional MLP in INR. Intuitively, the KAN employs multiple basis functions to form learnable activation functions through learnable weights, which should have the similar expressive power against MLP. However, through a prophase attempt of using KAN [1] for INR, it is a tendency that KAN is not as effective as state-of-the-art MLP-based INR methods [4], [6], especially for sophisticated signal representations such as images and point clouds.

In this work, we aim to improve the effectiveness of KAN for INR. Specifically, the contributions of this work are as follows:

- We first show the internal connections between KAN and MLP with a direct reformulation of the KAN layer. Concretely, we show that KAN can be seen as a special type of MLP by using multiple activation functions performed on inputs after repetition.
- Inspired by this reformulation, we proposed a generalized KAN (GKAN) for INR. The proposed GKAN employs multiple sinusoidal activation functions with different frequencies to more effectively represent signals under the continuous representation framework.
- We reveal the underlying advantages of GKAN for continuous representation from the neural tangent kernel and Lipschitz smooth perspectives with theoretically grounded justifications.
- Numerical experiments for image, video, point cloud, and neural radiance field (NeRF) representations showcase the effectiveness and superiority of the proposed GKAN for continuous representation.

In the rest of this letter, we introduce the proposed GKAN and related justifications in Sec. II. Numerical experimental results are given in Sec. III. Sec. IV concludes this paper.

II. The Proposed Method

A. Reformulation of KAN

In this subsection, we reformulate KAN as a special type of MLP with multiple activation functions. A single KAN [1] layer $\Phi(\cdot):\mathbb{R}^n\to\mathbb{R}^n$ is formulated as

$$\Phi(\mathbf{x})=\begin{bmatrix}\phi_{1,1}(\cdot) & \phi_{1,2}(\cdot) & \cdots & \phi_{1,n}(\cdot)\\ \phi_{2,1}(\cdot) & \phi_{2,2}(\cdot) & \cdots & \phi_{2,n}(\cdot)\\ \vdots & \vdots & & \vdots\\ \phi_{n,1}(\cdot) & \phi_{n,2}(\cdot) & \cdots & \phi_{n,n}(\cdot)\end{bmatrix}\mathbf{x}, \tag{1}$$

where each function $\phi_{i,j}(\cdot):\mathbb{R}\to\mathbb{R}$ is parameterized by

$$\phi_{i,j}(x)=w^{i,j}(b(x)+\sum_{k=1}^{K}c_k^{i,j}B_k(x)). \tag{2}$$

Here, $\{c_k^{i,j}\}_{k=1}^K$ are trainable parameters. In the original configuration of KAN [1], the basis function $\{B_k(\cdot)\}_{k=1}^K$ is chosen as the spline functions. However, it is not necessary to solely use the splines. Other basis functions can also be considered as

Yisi Luo is with the School of Mathematical and Statistics, Xi'an Jiaotong University, Xi'an, P.R.China. (e-mail: yisiluo1221@foxmail.com)

shown in recent works [2], [3]. Stacking multiple KAN layers $\{\Phi_l(\cdot)\}_{l=1}^{L}$ serially results in a KAN formulated as

$$\mathrm{KAN}(\mathbf{x}) = \Phi_L(\Phi_{L-1}(\cdots\Phi_1(\mathbf{x}))) : \mathbb{R}^n \to \mathbb{R}. \tag{3}$$

Intuitively, KAN constructs learnable activation functions by learning a weighted sum of the selected basis functions. However, the learnable weights $\{c_k\}_{k=1}^{K}$ and the basis functions $\{B_k(\cdot)\}_{k=1}^{K}$ can be disentangled from the representation, which makes KAN more like an MLP with learnable weight matrices and activation functions $\{B_k(\cdot)\}_{k=1}^{K}$, as shown in the following observation.

Proposition 1. *Given a KAN layer $\Phi(\cdot)$ formulated as (1) & (2), there exist a weight matrix $\mathbf{W} \in \mathbb{R}^{n\times n(K+1)}$ and a fixed function $B(\cdot) : \mathbb{R}^n \to \mathbb{R}^{n(K+1)}$ without learnable parameters such that $\Phi(\mathbf{x}) = \mathbf{W}B(\mathbf{x})$.*

Proof. Every component function $\phi_{i,j}(\cdot)$ of the KAN layer $\Phi(\cdot)$ can be formulated as

$$\phi_{i,j}(\cdot) = \underbrace{\begin{bmatrix} w^{i,j} & w^{i,j}c_1^{i,j} & \cdots & w^{i,j}c_K^{i,j}\end{bmatrix}}_{\mathbf{c}^{i,j}\in\mathbb{R}^{1\times(K+1)}} \underbrace{\begin{bmatrix} b(\cdot) \\ B_1(\cdot) \\ \vdots \\ B_K(\cdot)\end{bmatrix}}_{\hat{B}(\cdot):\mathbb{R}\to\mathbb{R}^{K+1}}. \tag{4}$$

Hence we have

$$\begin{aligned}\Phi(\mathbf{x}) &= \begin{bmatrix} \mathbf{c}^{1,1}\hat{B}(\cdot) & \mathbf{c}^{1,2}\hat{B}(\cdot) & \cdots & \mathbf{c}^{1,n}\hat{B}(\cdot) \\ \mathbf{c}^{2,1}\hat{B}(\cdot) & \mathbf{c}^{2,2}\hat{B}(\cdot) & \cdots & \mathbf{c}^{2,n}\hat{B}(\cdot) \\ \vdots & \vdots & & \vdots \\ \mathbf{c}^{n,1}\hat{B}(\cdot) & \mathbf{c}^{n,2}\hat{B}(\cdot) & \cdots & \mathbf{c}^{n,n}\hat{B}(\cdot)\end{bmatrix}\mathbf{x} \\ &= \underbrace{\begin{bmatrix} \mathbf{c}^{1,1} & \mathbf{c}^{1,2} & \cdots & \mathbf{c}^{1,n} \\ \mathbf{c}^{2,1} & \mathbf{c}^{2,2} & \cdots & \mathbf{c}^{2,n} \\ \vdots & \vdots & & \vdots \\ \mathbf{c}^{n,1} & \mathbf{c}^{n,2} & \cdots & \mathbf{c}^{n,n}\end{bmatrix}}_{\mathbf{W}\in\mathbb{R}^{n\times n(K+1)}} \underbrace{\begin{bmatrix} \hat{B}(\cdot) & & & \\ & \hat{B}(\cdot) & & \\ & & \ddots & \\ & & & \hat{B}(\cdot)\end{bmatrix}}_{B(\cdot):\mathbb{R}^n\to\mathbb{R}^{n(K+1)}}\mathbf{x}.\end{aligned} \tag{5}$$

Here, the activation function $B(\cdot)$ is defined by repeating $\hat{B}(\cdot)$ n times, and $B(\cdot)$ contains no learnable parameters. Equivalently, this can be done by repeating the input vector $\mathbf{x}$ n times instead and rearranging the weight matrix:

$$\Phi(\mathbf{x}) = \underbrace{\begin{bmatrix} \texttt{unfold}(\mathbf{c}^{1,1^T}, \mathbf{c}^{1,2^T}, \cdots, \mathbf{c}^{1,n^T}) \\ \texttt{unfold}(\mathbf{c}^{2,1^T}, \mathbf{c}^{2,2^T}, \cdots, \mathbf{c}^{2,n^T}) \\ \cdots \\ \texttt{unfold}(\mathbf{c}^{n,1^T}, \mathbf{c}^{n,2^T}, \cdots, \mathbf{c}^{n,n^T})\end{bmatrix}}_{n\times n(K+1)} \underbrace{\begin{bmatrix} b(\mathbf{x}) \\ B_1(\mathbf{x}) \\ \vdots \\ B_K(\mathbf{x})\end{bmatrix}}_{n(K+1)\times 1}, \tag{6}$$

where $\texttt{unfold}(\cdot) : \mathbb{R}^{(K+1)\times n} \to \mathbb{R}^{1\times n(K+1)}$ is the unfolding operator from a matrix to a vector and the scalar-to-scalar function $b(\mathbf{x})$ (also $B_k(\mathbf{x})$) is applied on each element of the input vector $\mathbf{x}$ and returns a vector. □

From Proposition 1, we see that KAN can be written in a similar way as MLP, i.e., the learnable weight $\mathbf{W}$ and the activation function $B(\cdot)$ can be disentangled. The difference is how the activation function $B(\cdot)$ is designed. The MLP is composed of a single activation function (for example $B_1(\cdot)$) and a learnable weight matrix $\mathbf{W}$, i.e., $\Phi'(\mathbf{x}) = \mathbf{W}B_1(\mathbf{x})$[1]. Differently, in KAN, the input vector is first being repeated $K+1$ times, and $K+1$ different activation functions (i.e., $b(\cdot), B_1(\cdot), \cdots, B_K(\cdot)$) are used to process the inputs after repetition, as shown in (6). After the activation, both MLP and KAN employ a learnable weight matrix $\mathbf{W}$ to gather the outputs. Especially, MLP can be seen as a special case of KAN when only using one basis function.

Corollary 1. *Consider a KAN layer in (1). When the component function $\phi_{i,j}(\cdot)$ is parameterized by just on basis function, e.g., $\phi_{i,j}(x) = c^{i,j}B_1(x)$, then KAN recovers to the MLP with activation function $B_1(\cdot)$, i.e., $\Phi(\mathbf{x}) = \mathbf{W}B_1(\mathbf{x})$, where $\mathbf{W}$ follows (5) by setting $\mathbf{c}^{i,j} = c^{i,j}$.*

Proof. By changing the activation function $\hat{B}(\cdot)$ in (5) as the basis function $B_1(\cdot)$, or by examining that the weight matrix in (6) is exactly $\mathbf{W}$, we have the conclusion. □

In summary, through reformulation of a KAN layer, we have the following important observations:

- KAN can be reformulated in a similar way as MLP, i.e., the learnable weights and activation functions can be disentangled.
- The activation function of KAN is defined by repeating the input vector multiple times and performing multiple activation functions on the inputs after repetition.
- MLP is a special case of KAN when only considering one basis function.

B. Generalized KAN for INR

Inspired by the reformulation of KAN, we introduce a generalized KAN that employs multiple sinusoidal activation functions with different frequencies for INR. It was shown that the sinusoidal activation function, i.e., $\sin(\omega\ \cdot)$ where ω is the frequency hyperparameter, is more effective than the conventional ReLU activation function for continuous representation [4]. However, the classical INR method [4] employs only one sinusoidal function with the same frequency ω for all layers of the MLP. In this work, we propose to use multiple sinusoidal functions with different frequencies in the KAN to continuously represent signals. Specifically, the proposed GKAN layer $\Phi(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is formulated by setting the basis functions $b(\cdot), B_1(\cdot), B_K(\cdot)$ as sinusoidal functions with different frequencies:

$$\Phi_l(\mathbf{x}) = \mathbf{W}_l\left[\sin(\omega_{(1)}\mathbf{x}), \sin(\omega_{(2)}\mathbf{x}), \cdots, \sin(\omega_{(K)}\mathbf{x})\right]^T, \tag{7}$$

where $\mathbf{W}_l \in \mathbb{R}^{n\times nK}$ is the learnable weight matrix and $[\omega_{(1)}, \cdots, \omega_{(K)}]$ are different frequencies. The proposed GKAN is formulated by stacking multiple GKAN layers:

$$\mathrm{GKAN}(\mathbf{x}) = \Phi_L(\Phi_{L-1}(\cdots\Phi_1(\mathbf{W}_0\mathbf{x}))) : \mathbb{R}^n \to \mathbb{R}, \tag{8}$$

[1]We have changed the order of weight matrix and activation function which makes it slightly different from what an MLP layer is defined in mainstream research. However, this does not change the overall structure of MLP since MLP is composed of alternate weight matrices and activation functions.

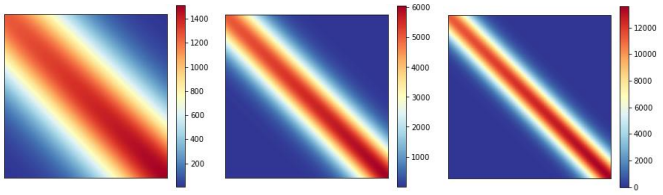

Fig. 1. The NTK matrix of a GKAN in Proposition 2 with different frequencies. The NTK matrix has a strong diagonal property with tunable bandwidth, which makes GKAN suitable for continuously representing complicated signals.

where we have added an extra weight matrix $\mathbf{W}_0$ before the first layer. To apply the proposed GKAN for INR, we just need to view the input vector $\mathbf{x}$ as a coordinate vector and expect the output to be the value at this coordinate. For example, suppose that we are given an image $\mathbf{O} \in \mathbb{R}^{n_1 \times n_2}$, the optimization model for continuously representing this image by using the proposed GKAN is formulated as

$$\min_{\theta} \sum_{i,j} \|\mathbf{O}_{(i,j)} - \mathrm{GKAN}([i; j])\|_F^2 + \psi(\theta), \tag{9}$$

where $\theta := \{\mathbf{W}_l\}_{l=0}^{L}$ denote the learnable parameters, $\mathbf{O}_{(i,j)}$ denotes the (i, j)-th element of $\mathbf{O}$, and $\psi(\theta)$ denotes a regularization term. Such model is also applicable for non-meshgrid data such as point cloud. Besides, further modifications can easily apply the GKAN for tasks such as surface representation and scene representation using neural radiance fields.

The differences between our GKAN as compared with previous networks are as follows. Compared to classical MLPs, the proposed GKAN employs multiple activation functions (i.e., multiple sin functions with different frequencies), which inclines to bring higher flexibility for characterizing complicated signals under the INR framework (see related justifications in Secs. II-C & II-D). Compared to KAN [1], the proposed GKAN differs that we employ sinusoidal functions with different frequencies instead of splines as activation functions, which inclines to hold stronger abilities for INR as shown in previous work [4]. Meanwhile, the GKAN simplifies the intrinsic formulation of the network layers as compared with KAN (i.e., from learnable activation functions to disentangled learnable weights and activation functions), which tends to better fit the calculating and optimization platforms used in modern deep learning methods and hence GKAN is expected to be more efficient than KAN.

C. Neural Tangent Kernel Perspective

In this subsection, we reveal the effectiveness of the proposed GKAN from the neural tangent kernel (NTK) [5], [10] perspective. The NTK theory [10] shows that the regression process using a neural network convergent to a kernel regression under some conditions, where the kernel that measures the similarity between samples is defined as the NTK:

$$k_{\mathrm{NTK}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbb{E}_{\theta} \langle \nabla_{\theta} f_{\theta}(\mathbf{x}_i), \nabla_{\theta} f_{\theta}(\mathbf{x}_j) \rangle, \tag{10}$$

where $f_{\theta}(\cdot)$ is the neural network with parameters θ. The NTK matrix $\mathbf{K}$ is further defined by $\mathbf{K}_{(i,j)} := k_{\mathrm{NTK}}(\mathbf{x}_i, \mathbf{x}_j)$. If the NTK matrix has strong diagonal property, then the NTK

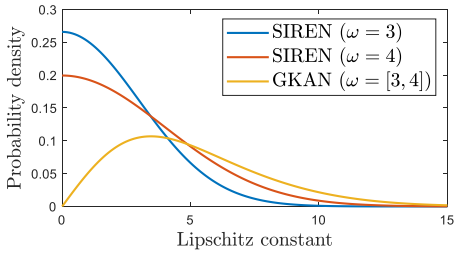

Fig. 2. The p.d.f. of the Lipschitz constant variable of a SIREN layer [4] and a GKAN layer with different frequencies ω. The p.d.f. of the Lipschitz constant of a GKAN layer distributes more gently, suggesting that GKAN tends to more easily represent functions with different smooth degrees.

$k_{\mathrm{NTK}}(\mathbf{x}_i, \mathbf{x}_j)$ is a good measure of similarity between two inputs, which is then suited for kernel regression (i.e., suited for representing complicated signals). Such diagonal property can also be interpreted as shift-invariance [5], which refers to that if the NTK $k_{\mathrm{NTK}}(\mathbf{x}_i, \mathbf{x}_j)$ only depends on the difference between inputs points (i.e., shift-invariance), then it is a good measure of similarity between two inputs. Now, we consider the NTK derived by the proposed GKAN. For simplicity, we consider an univariate GKAN.

Proposition 2. *Consider a GKAN with weights* $\theta := [w_1, w_2, h]$ *and two frequencies* $\omega = [\omega_{(1)}, \omega_{(2)}]^T$*:*

$$f_{\theta}(x) = w_1 \sin(\omega_{(1)} h x) + w_2 \sin(\omega_{(2)} h x) : \mathbb{R} \to \mathbb{R},$$

then the NTK of this GKAN is

$$\begin{aligned} &k_{\mathrm{NTK}}(x_i, x_j) = \mathbb{E}_{\theta} \langle \nabla_{\theta} f_{\theta}(x_i), \nabla_{\theta} f_{\theta}(x_j) \rangle \\ &= \mathbb{E}_{\theta} \big[x_i x_j \big(\sum_{k=1,2} \underbrace{w_k^2 \omega_{(k)}^2}_{\text{Scale term}} \underbrace{\cos(\omega_{(k)} h x_i) \cos(\omega_{(k)} h x_j)}_{\text{Sign term}} + \\ &\quad w_1 w_2 \omega_{(1)} \omega_{(2)} (\cos(\omega_{(1)} h x_i) \cos(\omega_{(2)} h x_j) + \cos(\omega_{(2)} h x_i) \\ &\quad \cos(\omega_{(1)} h x_j)) \big) + \sum_{k=1,2} \sin(\omega_{(k)} h x_i) \sin(\omega_{(k)} h x_j) \big]. \end{aligned} \tag{11}$$

Proof. Direct calculate yields the result. □

In (11), the scale term is proportional to $[\omega_{(1)}, \omega_{(2)}]$, suggesting that tuning $[\omega_{(1)}, \omega_{(2)}]$ is able to adjust the ability of GKAN for representing high-frequency components. When $x_i = x_j$, the sign term is always positive, while the non-diagonal elements of the NTK matrix $\mathbf{K}$ (i.e., when $x_i \neq x_j$) can be either positive or negative. Taken the above facts, the NTK matrix $\mathbf{K}$ of the GKAN has a strong diagonal property (as shown in Fig. 1), which makes it suitable for kernel regression using the NTK, i.e., suitable for continuously representing complicated signals [5], [7].

D. Lipschitz Smooth Perspective

Next, we show the advantage of GKAN as compared with traditional MLP structure from the Lipschitz smooth perspective. We focus on comparing GKAN with the popular SIREN [4] structure, an MLP with sinusoidal activation function. Equipped with the sinusoidal activation function, both the GKAN and SIREN have the Lipschitz smooth property and we are interested in the distribution of the Lipschitz constant

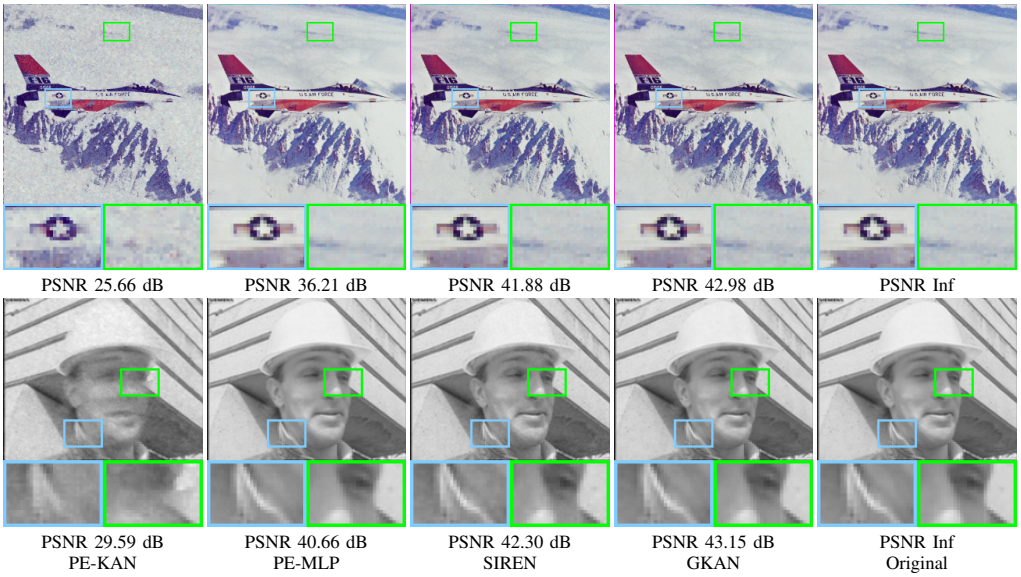

Fig. 3. The image and video fitting results using different methods on the image "House" and the video "Foreman".

TABLE I
QUANTITATIVE RESULTS OF DIFFERENT METHODS FOR FITTING IMAGES AND VIDEOS. (PSNR ↑ AND SSIM ↑)

Method	"Plane"		"Peppers"		"House"		"Sailboat"		"Earth"		"Foreman"		"Carphone"	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
PE-KAN	25.66	0.714	23.20	0.574	21.96	0.599	22.56	0.617	26.48	0.730	29.59	0.837	28.57	0.852
PE-MLP	36.21	0.980	38.25	0.988	36.98	0.983	37.29	0.987	39.38	0.990	40.66	0.991	41.00	0.992
SIREN	41.88	0.983	39.94	0.972	41.14	0.978	39.44	0.979	42.16	0.988	42.30	0.984	41.84	0.978
GKAN	**42.98**	**0.990**	**44.30**	**0.993**	**43.76**	**0.991**	**44.20**	**0.994**	**44.94**	**0.997**	**43.15**	**0.992**	**45.71**	**0.995**

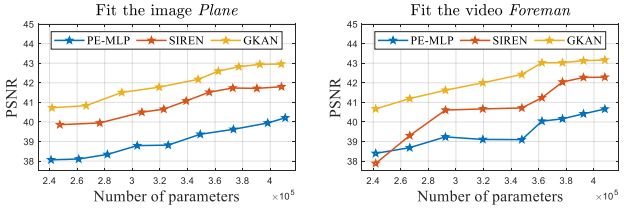

Fig. 4. The PSNR trajectory w.r.t. number of parameters using different methods for fitting the image and video.

TABLE II
QUANTITATIVE RESULTS OF DIFFERENT METHODS FOR FITTING POINT CLOUDS. (MSE ↓ AND R^2 ↑)

Method	"Rabbit"		"Mario"		"Duck"		"Cola"	
	MSE	R^2	MSE	R^2	MSE	R^2	MSE	R^2
PE-KAN	0.098	0.918	0.145	0.931	0.081	0.940	0.052	0.992
PE-MLP	0.175	0.739	0.179	0.894	0.128	0.847	0.060	0.990
SIREN	0.058	0.971	0.068	0.984	0.062	0.964	0.051	0.992
GKAN	**0.048**	**0.980**	**0.063**	**0.987**	**0.052**	**0.975**	**0.036**	**0.999**

in the representation. We give an example by using a single SIREN or GKAN layer.

Proposition 3. *Consider an univariate GKAN layer with weights* $w = [w_1, w_2]^T$ *and frequencies* $\omega = [\omega_{(1)}, \omega_{(2)}]^T$*:*

$$\Phi(x) = w_1 \sin(\omega_{(1)} x) + w_2 \sin(\omega_{(2)} x) : \mathbb{R} \to \mathbb{R}$$

and a SIREN layer: $\Phi'(x) = w \sin(\omega x) : \mathbb{R} \to \mathbb{R}$*. Assume that* $w, w_1, w_2 \sim i.i.d.\ \mathcal{N}(0,1)$*. Then we have*

$$\begin{aligned} |\Phi(x_1) - \Phi(x_2)| &\le L|x_1 - x_2|, \\ |\Phi'(x_1) - \Phi'(x_2)| &\le L'|x_1 - x_2|, \end{aligned} \tag{12}$$

where the Lipschitz constants L *and* L' *follow the p.d.f.*

$$\begin{aligned} p_L(x) &= \frac{4e^{\frac{-x^2}{\omega_{(1)}^2 + \omega_{(2)}^2}}}{\sqrt{2\pi(\omega_{(1)}^2 + \omega_{(2)}^2)}} \left(\int_{-\infty}^{x} \hat{p}(t)dt - \frac{1}{2}\right), \\ p_{L'}(x) &= \frac{2\exp\frac{-x^2}{\omega^2}}{\sqrt{2\pi\omega^2}},\ \hat{p}(t) = \frac{1}{\sqrt{2\pi\hat{\omega}}} e^{\frac{-t^2}{\hat{\omega}}}, \end{aligned} \tag{13}$$

where $\hat{\omega} = \omega_{(1)}^2 \omega_{(2)}^2 / (\omega_{(1)}^2 + \omega_{(2)}^2)$.

In Fig. 2, we visualize the p.d.f. of the Lipschitz constant of a SIREN or a GKAN layer. The p.d.f. of GKAN Lipschitz constant distributes more gently, which suggests that GKAN has a higher probability for characterizing a wider range of functions with different smooth degrees. In practice, the GKAN or SIREN are stacked with multiple layers, which results in severe non-convexity and hence the initialization is important for INR methods [4], [7]. According to Fig. 2, the initialized Lipschitz constants of different layers of GKAN may vary significantly due to the relatively flat distributions of the Lipschitz constant, while the initialized Lipschitz constants of different layers of SIREN may be more concentrate to zero. This gives GKAN the potential ability to more easily represent functions with different smooth degrees as compared with SIREN, leading to better representation abilities.

III. EXPERIMENTS

In experiments, we compare our GKAN with different INR methods. For fairness, all the models are optimized with the same optimizer under the same hyperparameters setting. Also,

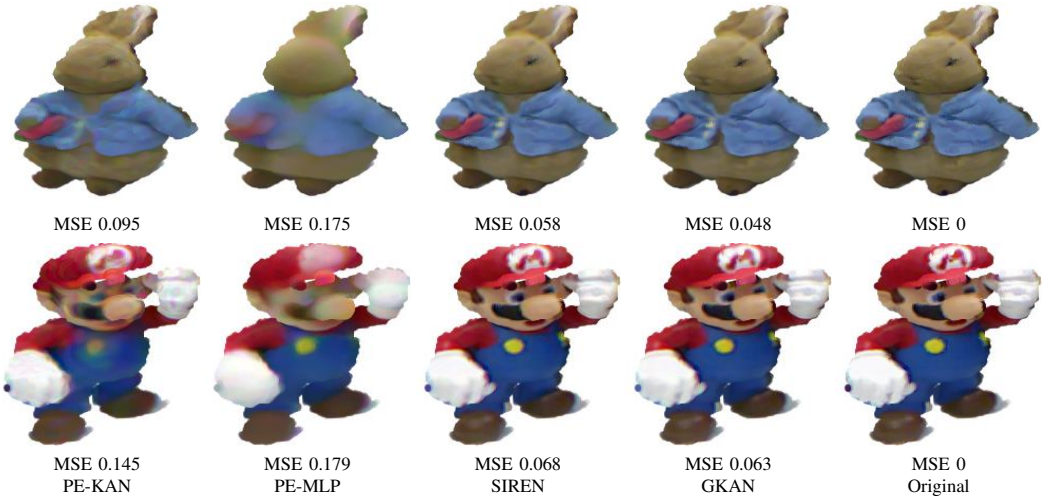

Fig. 5. The point cloud fitting results using different methods on "Rabbit" and "Mario".

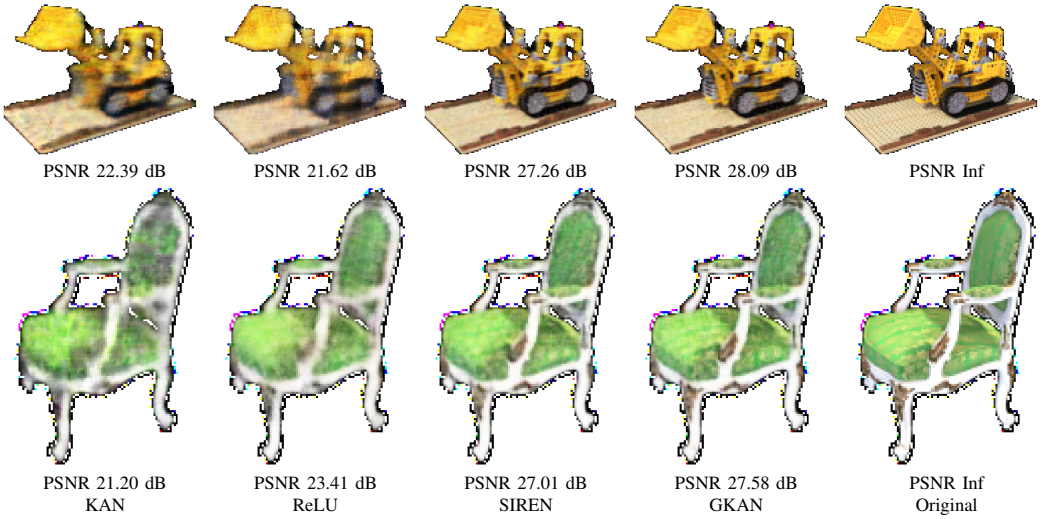

Fig. 6. The NeRF representation results using different methods on "Lego" and "Chair".

TABLE III
QUANTITATIVE RESULTS OF DIFFERENT METHODS FOR NERF REPRESENTATION. (PSNR ↑ AND SSIM ↑)

Method	"Lego"		"Chair"		"Drum"		"Ficus"		"Hotdog"		"Materials"		"Mic"		"Ship"	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
PE-KAN	22.39	0.909	21.20	0.849	21.57	0.908	27.23	0.961	27.51	0.954	25.05	0.898	28.61	0.953	27.09	0.846
PE-MLP	21.62	0.895	23.41	0.895	24.21	0.933	30.46	0.977	30.84	0.975	28.18	0.931	29.04	0.965	29.59	0.887
SIREN	27.26	0.953	27.01	0.937	27.23	0.957	34.58	0.989	33.29	0.981	31.34	0.959	38.03	0.985	31.19	0.913
GKAN	**28.09**	**0.959**	**27.58**	**0.948**	**27.60**	**0.959**	**35.61**	**0.991**	**33.50**	**0.982**	**31.88**	**0.963**	**38.82**	**0.989**	**32.05**	**0.923**

the number of parameters of different networks is set to be similar to ensure fair comparisons.

A. Fitting Image and Video Using GKAN

First, we apply the proposed GKAN for image and video fitting tasks. We consider the following baselines: the original KAN [1] with positional encoding [5] before the first layer (PE-KAN), the ReLU-based MLP with positional encoding before the first layer (PE-MLP) [5], and the sinusoidal activation function-based method SIREN [4]. We use the image datasets "Plane", "Peppers", "House", "Sailboat", "Earth", and video datasets "Foreman", "Carphone" which are online available[2]. We use the selected methods to continuously represent these images and videos by feeding coordinates into networks and calculating loss between outputs and pixel values. The results are evaluated by PSNR and SSIM. The results are given in Fig. 3 and Table I. GKAN obtains better performances than PE-KAN and SIREN, which validates the effectiveness of GKAN for INR. In Fig. 4, we show the results w.r.t. different number of parameters. We can see that GKAN obtains a better result than baselines across different number of parameters.

B. Fitting Point Cloud Using GKAN

Next, we use different INR methods for representing point clouds. We use the dataset containing color point clouds "Rabbit", "Mario", "Duck", and "Cola", which are online

[2]https://sipi.usc.edu/database/database.php & http://trace.eas.asu.edu/yuv/

available[3]. We use different INR methods to represent these point clouds by feeding coordinates into the networks and optimizing the loss between outputs and the color values at those coordinates. The results are evaluated by normalized root MSE and R-square (R^2). Fig. 5 and Table II report the results. It can be seen that GKAN obtain better performances than other INR methods for color point cloud representation.

C. NeRF Representation Using GKAN

Next, we consider the NeRF representation [6]. We use eight scenes in the Blender dataset, which is online available[4]. We use a relatively lightweight setting to test all the methods under the same hyperparameters configuration. The results are shown in Fig. 6 and Table III. It can be seen that GKAN obtains better accuracy for representing different scenes using the NeRF representation.

IV. Conclusions

We have reformulated KAN as a special type of MLP with multiple activation functions and proposed the GKAN by using sinusoidal activation functions with multiple frequencies, which showed considerable performance improvements over traditional MLP for continuously representing signals. We revealed the advantages of GKAN from the neural tangent kernel and Lipschitz smooth perspectives. Experiments on image, video, point cloud, and NeRF showed the effectiveness of the proposed method for continuous signal representation.

References

[1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljai, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-arnold networks," *arXiv: 2404.19756*, 2024.

[2] Z. Li, "Kolmogorov-Arnold networks are radial basis function networks," *arXiv: 2405.06721*, 2024.

[3] S. SS and G. R, "Chebyshev polynomial-based Kolmogorov-Arnold networks: An efficient architecture for nonlinear function approximation," *arXiv: 2405.07200*, 2024.

[4] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 7462–7473.

[5] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, "Fourier features let networks learn high frequency functions in low dimensional domains," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 7537–7547, 2020.

[6] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "NeRF: Representing scenes as neural radiance fields for view synthesis," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 405–421.

[7] Z. Liu, H. Zhu, Q. Zhang, J. Fu, W. Deng, Z. Ma, Y. Guo, and X. Cao, "FINER: Flexible spectral-bias tuning in implicit neural representation by variable-periodic activation functions," in *IEEE/CVF Computer Vision and Pattern Recognition Conference (CVPR)*, 2024.

[8] V. Saragadam, D. LeJeune, J. Tan, G. Balakrishnan, A. Veeraraghavan, and R. G. Baraniuk, "WIRE: Wavelet implicit neural representations," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 18 507–18 516.

[9] R. Liang, H. Sun, and N. Vijaykumar, "CoordX: Accelerating implicit neural representation with a split MLP architecture," in *International Conference on Learning Representations*, 2022.

[10] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, p. 85808589.

[3]http://www.vision.deis.unibo.it/research/80-shot

[4]https://github.com/bmild/nerf