DAREK - Distance Aware Error for Kolmogorov Networks

Masoud Ataei
Electrical and Computer Engg.
University of Maine
Orono, ME, USA
masoud.ataei@maine.edu

Mohammad Javad Khojasteh
Electrical and Microelectronic Engg.
Rochester Institute of Technology
Rochester, NY, USA
mjkeme@rit.edu

Vikas Dhiman
Electrical and Computer Engg.
University of Maine
Orono, ME, USA
vikas.dhiman@maine.edu

***Abstract*—In this paper, we provide distance-aware error bounds for Kolmogorov Arnold Networks (KANs). We call our new error bounds estimator DAREK—Distance Aware Error for Kolmogorov networks. Z. Liu et al. [1] provide error bounds, which may be loose, lack distance-awareness, and are defined only up to an unknown constant of proportionality. We review the error bounds for Newton's polynomial, which is then generalized to an arbitrary spline, under Lipschitz continuity assumptions. We then extend these bounds to nested compositions of splines, arriving at error bounds for KANs. We evaluate our method by estimating an object's shape from sparse laser scan points. We use KAN to fit a smooth function to the scans and provide error bounds for the fit. We find that our method is faster than Monte Carlo approaches, and that our error bounds enclose the true obstacle shape reliably.**

***Index Terms*—Error bounds, neural networks, splines.**

I. Introduction

Kolmogorov-Arnold Networks (KANs) [1] and its extensions [2]–[7] use B-Spline basis as activation functions. Utilizing splines in neural networks opens the door to more interpretable and analyzable neural networks [8]–[10]. In this work, we analyze distance-aware uncertainty estimation on KANs.

Uncertainty estimation for machine learning models can be categorized into two classes: probabilistic methods and worst-case methods. Probabilistic methods estimate probability distribution of model outputs, while worst-case methods provide error bounds on the outputs. A popular class of probabilistic methods [11]–[14] uses Monte Carlo (MC) approaches to estimate uncertainty, which are architecture invariant and can be applied to any neural network, including KANs, regardless of the number of layers or activation functions. However, these methods ignore the inductive biases embedded into an architecture resulting from the choice of layers and activation functions. Additionally, MC uncertainty estimation depends on training wider networks [12] or training multiple models [11], which increases their computation cost.

Gaussian processes (GPs) [15] offer architecture-specific uncertainty quantification, unlike MC sampling methods. The covariance function [15] determines how the predictions and uncertainties of a GP model change with the input. In the last decade, Deep GPs [16]–[18] have blurred the boundaries between neural networks and GPs by facilitating the learning of covariance functions and interpreting neural networks as GPs. GPs' limitations are the high computational cost and the need to store a large amount of training data for inference.

Probabilistic uncertainty approaches are often avoided in safety-critical applications due to their high computational cost, large sample size requirements, and reliance on hyperparameters [19]. We instead focus on worst-case analysis [20], which is easier to understand, implement, and debug, while being more robust and particularly useful with limited data.

We focus on a *distance-aware* worst-case analysis. An uncertainty estimator is considered distance-aware if its uncertainty increases monotonically with the input's distance from the training data [21]–[23]. While Z. Liu et al. [1] do provide error bounds for KAN, these bounds are loose, lack distance-awareness, and are unknown up to a constant factor. In contrast, we propose tighter and *distance-aware* error bounds to overcome these limitations.

In this paper, we introduce DAREKs (distance aware error for Kolmogorov networks), which are more *interpretable* than traditional neural networks, more *efficient* than traditional GPs and provide *distance-aware* uncertainty bounds. We specifically make these contributions: 1) establish analytical worst-case *distance-aware* error bounds for spline approximation; 2) extend the proposed error bounds to multi-layer splines; 3) develop a modified KAN architecture that provides distance-aware error bounds at the test point alongside the prediction; and 4) validate the scalability and efficacy of our method on various examples including object shape estimation.

II. Background

Let $\tau_{i:j} \coloneqq \{\tau_i, \tau_{i+1}, \dots, \tau_j\}$ be a sorted sequence of distinct points, and $f : [a, b] \to \mathbb{R}$ be an arbitrary function defined on the closed finite interval, with open bounds at the endpoints when its value is undefined. The divided differences of any function f on $\tau_{i:j}$ are defined recursively as follows [24]:

$$[\tau_1]f \coloneqq f(\tau_1), \qquad [\tau_1, \tau_2]f \coloneqq \frac{[\tau_2]f - [\tau_1]f}{\tau_2 - \tau_1},$$
$$[\tau_{1:k}]f \coloneqq \frac{[\tau_{2:k}]f - [\tau_{1:k-1}]f}{\tau_k - \tau_1}. \tag{1}$$

This work was partially supported by the National Science Foundation under Grant No. 2218063.

Newton's polynomial of order k, on a set of $k+1$ input-output pairs $f(\tau_{n:n+k}) \coloneqq \{(\tau_j, f(\tau_j))\}_{j=n}^{n+k}$ is defined by [24, p7],

$$\mathcal{P}_{k,n}[f(\tau_{1:m})](x) \coloneqq [\tau_n]f + \sum_{i=1}^{k}[\tau_{n:n+i}]f \prod_{j=n}^{n+i-1}(x-\tau_j).$$

Newton's polynomial is an alternate representation of a polynomial. One can sample any $k+1$ unique points on any kth order polynomial to get one of the many Newton's polynomial representations of the given polynomial. As long as the points are unique, all representations lead to polynomials with the same coefficients. It means a polynomial $f(x)$ of order k is identical to its Newton's polynomial, $\mathcal{P}_{k,n}[f(\tau_{n:n+k})](x) = f(x)$ [24].

Definition 1 (Piecewise polynomial)**.** *A piecewise polynomial $\hat{f}(x)$ of order k on a **given set of knots** $\tau_{1:m}$ is a combination of $m-1$ polynomials $\hat{f}_{[j]}(x)$ of order k, as follows:*

$$\hat{f}(x) = \sum_{i=0}^{k} c_{i,j}x^k =: \hat{f}_{[j]}(x) \qquad \forall x \in [\tau_j, \tau_{j+1}),$$
$$\text{s. t.} \quad \hat{f}_{[j]}(\tau_{j+1}) = \hat{f}_{[j+1]}(\tau_{j+1}). \tag{2}$$

$\hat{f}_{[j]}(x)$ represents the jth polynomial piece of $\hat{f}(x)$. Piecewise-polynomials are continuous but not necessarily smooth. When we refer to $\mathcal{P}_{k,j}$ we also mean the $[j]$th piece of piecewise polynomial.

B-Splines: A spline of order k is a smooth piecewise-polynomials of the same order. Any spline of order k that is differentiable $k-1$ times can be written as a linear combination of basis functions known as B-splines [24, p97].

Kolmogorov Arnold Networks: Z. Liu et al. [1] propose Kolmogorov Arnold Network (KAN) as a composition of 1-D splines $\phi_{l,i,j} : \mathbb{R} \to \mathbb{R}$, so that a two-layer KAN can approximate a function $f : \mathbb{R}^n \to \mathbb{R}$ using a model, $\text{KAN}_2(\mathbf{x}) = \sum_{i=1}^{2n+1}\phi_{2,1,i}(\sum_{j=1}^{n}\phi_{1,i,j}(x_j))$, where $\mathbf{x} = [x_j]_{j=1}^{n}$.

Since the B-splines basis functions depend on the knots and the model weights are learned from the training data, KAN is a *semi-parametric* model. It combines the accuracy and computational efficiency of parametric models (e.g. neural networks) with the interpretability of non-parametric models (e.g. splines, k-nearest neighbors). We use this property to develop a distance-aware uncertainty estimator for KAN.

Distance Awareness: Distance-aware uncertainty is driven by the intuition that a model's prediction uncertainty increases with distance from the training data. GPs [15] are inherently distance-aware, as specified by the kernel function. This concept was introduced into deep neural networks via DDU [22] and DUQ [23] using kernel functions. J. Liu et al. [21] define distance-awareness as a property of probabilistic estimators. Here, we extend their definition to make it compatible with worst-case analysis.

Definition 2 (Input distance awareness)**.** *Consider a predictive model $\hat{y} = \hat{f}(\mathbf{x})$ that estimates a true function $y = f(\mathbf{x})$ using in-domain training data inputs $\mathcal{X}_{IND} \subset \mathcal{X}$. Here, $(\mathcal{X}, d(\mathbf{x}, \mathbf{x}'))$ is a Hilbert space with a corresponding distance function $d(.,.)$. Let $u_{\hat{f}}(\mathbf{x}) : \mathcal{X} \to [0, \infty)$ be an uncertainty estimator such that for all $\mathbf{x} \in \mathcal{X}$, the true output is bounded $y \in [\hat{f}(\mathbf{x}) - u_{\hat{f}}(\mathbf{x}), \hat{f}(\mathbf{x}) + u_{\hat{f}}(\mathbf{x})]$. The uncertainty estimator $u_{\hat{f}}(\mathbf{x})$ is considered distance-aware if it monotonically increases with the test point's distance from the training data $d(\mathbf{x}, \mathcal{X}_{IND}) = \min_{\mathbf{x}' \in \mathcal{X}_{IND}} d(\mathbf{x}, \mathbf{x}')$.*

The critical assumption in all distance-aware uncertainty estimators is the choice of a distance function that encodes how the function varies near the training data. Here, we adopt Lipschitz continuity as the central assumption and introduce an extended notion of kth-order Lipschitz continuity as follows:

Assumption 1 (kth-order Lipschitz continuity)**.** *For any $k-1$ times differentiable function $f : [a, b] \to \mathbb{R}$, the kth-order Lipschitz constant $\mathcal{L}_f^k$ is the ratio of the maximum change in the $(k-1)$th derivative of f to the change in the input,*

$$\frac{|f^{(k-1)}(x) - f^{(k-1)}(y)|}{d(x,y)} \leq \mathcal{L}_f^k \quad \forall x \neq y \tag{3}$$

A function is kth order Lipschitz continuous if $\mathcal{L}_f^k$ exists.

If we assume a function f and its approximator $\hat{f}$ to be first order Lipschitz continuous with constant $\mathcal{L}_f^1$, then a trivial distance-aware uncertainty estimator is $u_{\hat{f}}(\mathbf{x}) = \mathcal{L}_f^1 d(x, \mathcal{X}_{IND})$. In the next section, we extend this notion to higher order splines and multi-layer KAN networks.

III. Method

In this section, we derive the error bound for the KAN model in three steps. First, we introduce the error bound on Newton's polynomial approximation. Second, we extend this error bound to arbitrary splines where the error at knots may not be zero. Lastly, we extend the resulting error bound to multi-layer KANs through addition and function composition.

Theorem 1 (Newton's Polynomial error bound [24])**.** *Consider $f : [a, b] \to \mathbb{R}$ has $k+1$ continuous derivatives (that is $f \in C^{(k+1)}$) and is $k+1$th-order Lipschitz continuous with constant $\mathcal{L}_f^{k+1}$. Let $\mathcal{P}_{k,j}[f(\tau_{1:m})]$ be Newton's polynomial fit that passes through the knots $f(\tau_{1:m}) \coloneqq \{(\tau_i, f(\tau_i))\}_{i=1}^{m}$, for all $j \in \{1, \ldots, m-k\}$, then the error at test point $x \in [\tau_j, \tau_{j+1})$ is bounded as follows:*

$$|f(x) - \mathcal{P}_{k,j}[f(\tau_{1:m})](x)| \leq \underbrace{\frac{\mathcal{L}_f^{k+1}}{(k+1)!}\left|\prod_{i=j}^{j+k}(x-\tau_i)\right|}_{=:\bar{u}_f(x;\tau_{1:m})} \tag{4}$$

The above bound is tight when the $k+1$th derivative is constant at the maximum amount, $f^{(k+1)}(x) = \mathcal{L}_f^{k+1}$.

Due to space constraints, we have included all the proofs in the supplementary material [1].

We use the error bound for Newton's polynomial as the basis of our analysis because it gives tight distance-aware bounds.

[1] https://arxiv.org/abs/2501.04757

The interpolation error, as defined by Thm. 1, assumes zero error at the knots. Under various conditions, for example in the presence of noise, zero error at knots is not possible. The next Lemma provides an error bound for any spline approximation with non-zero error at the knots.

Lemma 1 (Piecewise polynomial interpolation at knots)**.** *With the same assumptions as Thm. 1, consider a piecewise polynomial approximation $\hat{f}(x)$ that does not necessarily pass through all the knots $f(\tau_{1:m})$. Let $\hat{f}_{[j]}(x)$ be the $[j]$th polynomial piece of $\hat{f}(x)$ (Def. 1). The error function is defined as $e^f_j(x) := f(x) - \hat{f}_{[j]}(x)$ for all $x \in [a, b]$ and has known values at the knots. Then the error for $x \in [\tau_j, \tau_{j+1})$ is bounded by,*

$$|f(x) - \hat{f}(x)| \leq \bar{u}_f(x; \mathcal{T}) + |\mathcal{P}_{k,j}[e^f_j](x)| =: u_f(x; \mathcal{T}), \quad (5)$$

where $\mathcal{T} = \tau_{1:m}$ and $\bar{u}_f(x)$ is the error bound from Thm. 1. The above bound is tight when the inequality in Thm. 1 is tight and $\text{sign}(f(x) - \mathcal{P}_{k,j}[f](x)) = \text{sign}(\mathcal{P}_{k,j}[e^f_j](x))$ when $x \in [\tau_1, \tau_{m-k})$. This Lemma holds for any kth order piecewise polynomial, regardless of how $\hat{f}$ is computed.

Next, we discuss how the error propagates as we build compositions of such spline functions. For ease of exposition, we focus on two-layer case, with the multi-layer and multiple inputs extension provided in the supplementary material [1].

To extend layer-wise error bound to the two-layer case, consider a function $f(x) : [a, b] \to \mathbb{R}$ that is being approximated by a two-layer architecture $\hat{f}(x) = \hat{h}(\sum_{i=1}^n \hat{g}_i(x)) = \hat{h}(\mathbf{1}_n^\top \hat{\mathbf{g}}(x))$, where $\mathbf{1}_n$ is a vector of n ones. We note that any true function can always be decomposed into the same network structure as the approximation, $f(x) = h(\mathbf{1}_n^\top \mathbf{g}(x))$, and has infinitely many solutions. For example, when $n = 2$, one such decomposition is $g_1(x) = f(x)$, $g_2(x) = 0$ and $h(x) = x$. Moreover, given any decomposition $f(x) = h(g_1(x) + g_2(x))$ and an invertible function $v(x)$, we can get a new decomposition $f(x) = h'(g'_1(x) + g_2(x))$ where $h'(y) := h(v^{-1}(y))$ and $g'_1(x) := v(g_1(x) + g_2(x)) - g_2(x)$. The tightest worst-case error can be achieved by picking a true decomposition that minimizes layer-wise error across all possible decompositions. We will revisit the discussion of picking a true decomposition in Assumption 2. The next theorem describes the error bounds with the knowledge of true decomposition values at the knots.

Theorem 2 (Two-layer KAN error bound)**.** *Consider a true two-layer function $f(x) : [a, b] \to \mathbb{R}$. We approximate the true function $f(x)$ using a composition of kth order piecewise polynomials $\hat{f}(x) = \hat{h}(\mathbf{1}_n^\top \hat{\mathbf{g}}(x))$. Let the true decomposition at the knots be given as $f(\tau_i) = h(\mathbf{1}_n^\top \mathbf{g}(\tau_i))$ for all $\tau_i \in \tau_{1:m}$. Define the errors at knots as $e^h(.) := h(.) - \hat{h}(.)$ and $e^{\mathbf{g}}(.) := \mathbf{g}(.) - \hat{\mathbf{g}}(.)$. Also, assume that the second layer h is first order Lipschitz continuous with constant $\mathcal{L}^1_h$. Let $f, h, \mathbf{g} \in C^{(k+1)}$, with known Lipschitz constants, and let $\hat{h}$, $\hat{\mathbf{g}}$ be piecewise polynomials. Then the error bound for the two-layer approximation is given by:*

$$|f(x) - \hat{f}(x)| \leq u_h(\mathbf{1}_n^\top \hat{\mathbf{g}}(x); \mathcal{T}_{\mathbf{g}_m}) + \mathcal{L}^1_h \mathbf{1}_n^\top \boldsymbol{u}_{\mathbf{g}}(x; \tau_{1:m}), \quad (6)$$

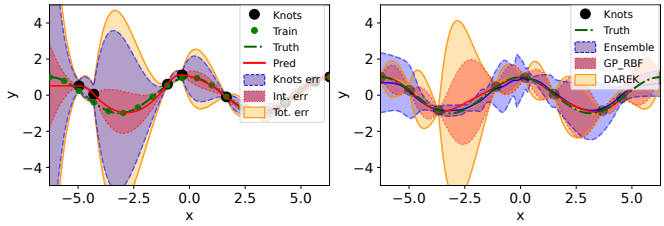

Fig. 1. The error bounds of KAN model on cos function. **left)** one-layer model, **right)** uncertainty estimation for a 2-layer DAREK, Ensemble, and GP on a cosine function. Ensemble and GP's uncertainty bounds are shown within the $\pm 3\sigma$ range.

where $\mathcal{T}_{\mathbf{g}_m} := \{\mathbf{1}_n^\top \hat{\mathbf{g}}(\tau_i)\}_{i=1}^m$ is input for $\hat{h}$ layer at knots, and $\boldsymbol{u}_{\mathbf{g}}(x) := (u_{g_i})_{i=1}^n$ is the vector error in $\hat{g}_i$ spline approximations as estimated in Lemma 1. The above bound is tight when $h^{(1)}(t) = \text{sign}(h(\mathbf{1}_n^\top \hat{\mathbf{g}}(x)) - \hat{h}(\mathbf{1}_n^\top \hat{\mathbf{g}}(x)))\mathcal{L}^1_h$, for all $t \in [\mathbf{1}_n^\top \hat{\mathbf{g}}(x), \mathbf{1}_n^\top \mathbf{g}(x)]$.

Thm. 2 shows that lower layer errors $\boldsymbol{u}_{\mathbf{g}}$ contribute more to the final bound than the errors of later layers u_h as long as the Lipschitz constant for later layers $\mathcal{L}^1_h$ is greater than 1, which is typical. In such cases, we prefer a true function decomposition with the smallest errors in earlier layers to reduce propagated error. Consider a decomposition where $\mathbf{g}(\tau_i) := \hat{\mathbf{g}}(\tau_i)$, making zero error at the knots in the early layer. Note that Thm. 2 needs the true function decomposition only at the knots not at all points, hence, we do not make any assumption about $\mathbf{g}(x)$ outside the knots $x \notin \tau_{1:m}$. For $h(y)$ to be a proper function, no single input to h can map to different outputs. Therefore, for all $x_1 \neq x_2$, wherever $f(x_1) \neq f(x_2)$, if different inputs do not map to the same intermediate values, $\mathbf{1}^\top \hat{\mathbf{g}}(x_1) \neq \mathbf{1}^\top \hat{\mathbf{g}}(x_2)$, then the mapping $h : \mathbf{1}^\top \hat{\mathbf{g}}(x) \mapsto f(x)$ is a valid function. In other words, a true function decomposition, where all error at knots is concentrated in the last layer, always exists, as long as earlier layers do not collapse different inputs to the same intermediate values. While this assumption is straightforward, it is not always satisfied. Instead, we adopt a more conservative assumption, that the error at knots is equally distributed among the layers, leaving the worst-case analysis of this assumption for future work.

Assumption 2 (Equal division of error)**.** *The total function error at the knots is equally divided among the intermediate layers. For a two-layer network approximation, $\hat{f}(x) = \hat{h}(\mathbf{1}_n^\top \hat{\mathbf{g}}(x))$ the errors at layers $\hat{h}, \hat{\mathbf{g}}$ can be divided based on their contribution to the full error as (using Theorem 2),*

$$e^h(\mathbf{1}^\top \hat{\mathbf{g}}(\tau_i)) = e^{g_1}(\tau_i) = \cdots = e^{g_n}(\tau_i) \geq \frac{e^f(\tau_i)}{1 + n\mathcal{L}^1_h}, \quad (7)$$

where $e^f(\tau_i) = f(\tau_i) - \hat{f}(\tau_i)$, is the error at knots for the function f. The corresponding true function decomposition can be obtained by adding errors to the approximations, $g_j(\tau_i) = \hat{g}_j(\tau_i) + \text{sign}(e^f(\tau_i))e^{g_j}(\tau_i)$ and $h(\mathbf{1}_n^\top \mathbf{g}(\tau_i)) = \hat{h}(\mathbf{1}_n^\top \hat{\mathbf{g}}(\tau_i)) + \text{sign}(e^f(\tau_i))e^h(\mathbf{1}_n^\top \hat{\mathbf{g}}(\tau_i))$, where g_j is jth element of vector $\mathbf{g}$.

Similar to [21], we also assume an equal division of Lipschitz constants among all layers. Once we have layer-wise

error at knots, and layer-wise Lipschitz constants, we can use Thm 2 to find the error bound on the full function. We left dividing the Lipschitz constant among layers for future work.

Algorithm 1: DAREK

Data: Input-output pairs $f(\tau_{1:m})$, trained model $\hat{f} = \hat{h}_L(\dots \hat{\mathbf{h}}_2(\hat{\mathbf{h}}_1(\mathbf{x})))$, test point $\mathbf{x}^*$, order k, Lipschitz constants $\mathcal{L}_f^{(k+1)}$, $\mathcal{L}_f^{(1)}$.

Precompute errors $e^f = |f(\boldsymbol{\tau}_i) - \hat{f}(\boldsymbol{\tau}_i)|$.
Divide e^f among layers, $e^{\mathbf{h}_1}, \dots, e^{h_L}$ at knots. [Assum. 2]
Divide $\mathcal{L}_f^1, \mathcal{L}_f^{k+1}$ equally among layers ($\mathcal{L}_{h_l} = \sqrt[L]{\mathcal{L}_f}$).
for $l, n \in (\{0, \dots, L-1\}, \{1, \dots, N_{h_l}\})$ **do**
  /* $\mathcal{O}(\log_2(m))$ `binary search`. */
  Find j such that $\hat{y}_{l,n}(\boldsymbol{\tau}_j) \leq \hat{y}_{l,n}(\mathbf{x}^*) < \hat{y}_{l,n}(\boldsymbol{\tau}_{j+1})$.
  Fit Newton's Polynomials $\mathcal{P}_{k,j}[e^{h_{l,n}}]$ on the knots $\{(\hat{y}_{l,n}(\boldsymbol{\tau}_i), e^{h_{l+1,n}}(\hat{y}_{l,n}(\boldsymbol{\tau}_i))\}_{i=j}^{j+k}$
  Find $u_{h_{l+1,n}}(\hat{h}_{l,n}(\mathbf{x}^*); \hat{h}_{l,n}(\tau_{1:m}))$ [Lemma 1].
Compute error bound over f [Thm. 2]

DAREK Algorithm: We summarize DAREK in Algo. 1 to compute error bounds for a L-layer spline network with N_{h_l} hidden units in layer l. The intermediate layer values of the network are denoted as $\hat{\mathbf{y}}_0(\mathbf{x}) := \mathbf{x}$, $\hat{\mathbf{y}}_l(\mathbf{x}) := \hat{\mathbf{h}}_l(\dots \hat{\mathbf{h}}_2(\hat{\mathbf{h}}_1(\mathbf{x})))$ and $\hat{f}(\mathbf{x}) = \hat{y}_L(\mathbf{x})$. Note that subscript n denotes an element of a vector, and j denote jth piece of a piecewise-polynomial. To understand the algorithm consider two-layer approximation $\hat{f}(x) = \hat{h}_{2,1}(\hat{h}_{1,1}(x) + \hat{h}_{1,2}(x))$. The algorithm after dividing the prediction error e^f and Lipschitz constant budgets, $\mathcal{L}_f^{(1)}$ and $\mathcal{L}_f^{(k+1)}$, among splines ($\hat{h}_{2,1}$, $\hat{h}_{1,1}$, and $\hat{h}_{1,2}$), it calculates the error of spline using (4) and (5) for each spline in the network by substituting the knots of the particular spline in equations, then calculates the error of each layer ($u_{h_2} = u_{h_{2,1}}$ and $u_{h_1} = u_{h_{1,1}} + u_{h_{1,2}}$). Doing that, the total error can be calculated using (6), $u_f = u_{h_2} + \mathcal{L}_{h_2}^{(1)} u_{h_1}$.

IV. Experiments

In this section, we evaluate our proposed error estimation for KAN model, DAREK (6), through various experiments. We start by analyzing a simple model equivalent to a single spline, followed by comparison of DAREK's error bound with the uncertainty estimation of probabilistic models. Finally, we demonstrate the effectiveness of our approach by estimating an object's shape from a single laser scan. All experiments were conducted using cubic ($k = 3$) splines, with the first and $(k+1)$th derivative Lipschitz constants set to one.

Spline error bound: We trained a one-layer KAN model on a cos function using 20 equally spaced points from $[-2\pi, 2\pi]$ as training data, with 9 knots randomly selected from these samples. The model was trained for 200 epochs with a learning rate of one. As Fig. 1 (left) depicted, while the fitted model does not perfectly follow the actual function, the overall error bound completely covered the true function. The error bounds between knots grow from one knot and shrink when meeting another knot.

Comparison with baselines: To the best of our knowledge, our paper is the first to introduce distance-aware worst-case error bounds under higher-order Lipschitz smoothness. Hence a comparison with typical baselines is *not entirely fair*. In any case, Fig 1 compares 2-layer DAREK with Ensemble and GP. Ensembles is a Monte Carlo method consisting of $q = 10$ randomly initialized KAN models. For GP, we use a Radial Basis Function (RBF) kernel with hyperparameters length scale and variance equal to one. The ensemble's uncertainty does not follow clear patterns, while GP inherently captures the distance awareness through interpolation. The comparison with GP is *unfair* without explicitly relating Lipschitz constants to RBF kernel parameters, which is left for future work. Ensembles have a test-time computational complexity of $\mathcal{O}(qLN_h^2)$, where L is the number of layers and N_h is the number of hidden units per layer. GP has a computational complexity of $\mathcal{O}(m^3)$. In comparison, DAREK has a computational complexity of $\mathcal{O}(\log_2(m)LN_h) + \mathcal{O}(LN_h^2)$ (see Alg. 1), which is faster than Ensembles as long as $m < 2^n N_h$.

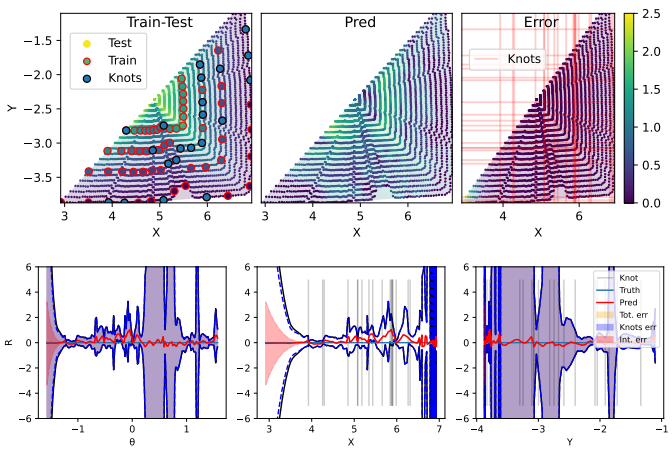

Fig. 2. In the first row, the plots from left to right show the train-test dataset, the trained model's predictions, and the interpolation error of each test point. The second row depicts the prediction of distance from the object boundary (R) versus the laser scanner angle (θ), the x position of the boundary point, and the y location of the boundary point.

Object shape estimation using KAN: In this experiment, a two-layer KAN with units [2,20,1] and 20 knots was trained on laser scans to learn the (Signed distance function) SDF profile of an object whose shape is being estimated. We took laser scan data from [25]. Fig. 2 top left shows the training, test samples, and selected knots, while the top middle plot presents the predicted values at the test points, and the top right plot displays the prediction error. The red lines indicate the positions of knots in the input layer. The lower plots, from left to right, show the prediction error at the boundary with respect to angle θ of the laser scan, x, and y respectively.

V. Conclusion

We derive distance-aware error bounds for KAN models and validate the error bounds on an object shape estimation problem. In future work, we plan to address the assumption of layer-wise error in more detail and evaluate our method on high-dimensional problems.

REFERENCES

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

[2] Z. Bozorgasl and H. Chen, "Wav-kan: Wavelet kolmogorov-arnold networks," *arXiv preprint arXiv:2405.12832*, 2024.

[3] R. Bresson, G. Nikolentzos, G. Panagopoulos, M. Chatzianastasis, J. Pang, and M. Vazirgiannis, "Kagnns: Kolmogorov-arnold networks meet graph learning," *arXiv preprint arXiv:2406.18380*, 2024.

[4] D. W. Abueidda, P. Pantidis, and M. E. Mobasher, "Deepokan: Deep operator network based on kolmogorov arnold networks for mechanics problems," *arXiv preprint arXiv:2405.19143*, 2024.

[5] C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus, "Kolmogorov-arnold networks (kans) for time series analysis," *arXiv preprint arXiv:2405.08790*, 2024.

[6] R. Genet and H. Inzirillo, "Tkan: Temporal kolmogorov-arnold networks," *arXiv preprint arXiv:2405.07344*, 2024.

[7] B. C. Koenig, S. Kim, and S. Deng, "Kan-odes: Kolmogorov–arnold network ordinary differential equations for learning dynamical systems and hidden physics," *Computer Methods in Applied Mechanics and Engineering*, vol. 432, p. 117397, 2024.

[8] B. Igelnik and N. Parikh, "Kolmogorov's spline network," *IEEE transactions on neural networks*, vol. 14, no. 4, pp. 725–733, 2003.

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

[10] S. Aziznejad, H. Gupta, J. Campos, and M. Unser, "Deep neural networks with trainable activations and controlled lipschitz constant," *IEEE Transactions on Signal Processing*, vol. 68, pp. 4688–4699, 2020.

[11] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *NeuRIPS*, vol. 30, 2017.

[12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, vol. 48. PMLR, 20–22 Jun 2016, pp. 1050–1059.

[13] W. J. Maddox, P. Izmailov, T. Garipov, D. P. Vetrov, and A. G. Wilson, "A simple baseline for bayesian uncertainty in deep learning," *NeuRIPS*, vol. 32, 2019.

[14] S. Jantre, N. M. Urban, X. Qian, and B.-J. Yoon, "Learning active subspaces for effective and scalable uncertainty quantification in deep neural networks," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 5330–5334.

[15] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. The MIT Press, 2006, vol. 1, no. 1.

[16] A. Damianou and N. D. Lawrence, "Deep gaussian processes," in *Artificial intelligence and statistics*. PMLR, 2013, pp. 207–215.

[17] Y. Song, Y. Liu, and P. M. Djurić, "Novel architecture of deep feature-based gaussian processes with an ensemble of kernels," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 6750–6754.

[18] A. Chakraborty and A. Chakraborty, "Scalable model-based gaussian process clustering," in *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2024, pp. 5730–5734.

[19] R. Cheng, R. M. Murray, and J. W. Burdick, "Limits of probabilistic safety guarantees when considering human uncertainty," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3182–3189.

[20] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Interval analysis*. Springer, 2001.

[21] J. Liu, Z. Lin, S. Padhy, D. Tran, T. Bedrax Weiss, and B. Lakshminarayanan, "Simple and principled uncertainty estimation with deterministic deep learning via distance awareness," *Advances in neural information processing systems*, vol. 33, pp. 7498–7512, 2020.

[22] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal, "Deep deterministic uncertainty: A new simple baseline," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 24 384–24 394.

[23] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," in *International conference on machine learning*. PMLR, 2020, pp. 9690–9700.

[24] C. De Boor and C. De Boor, *A practical guide to splines*. springer New York, 1978, vol. 27.

[25] A. Howard, "The robotics data set repository (radish)," *http://radish. sourceforge. net/*, 2003.