K-DAREK: Distance Aware Error for Kurkova Kolmogorov Networks

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

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

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

***Abstract*—Neural networks are powerful parametric function approximators, while Gaussian processes (GPs) are nonparametric probabilistic models that place distributions over functions via kernel-defined correlations but become computationally expensive for large-scale problems. Kolmogorov-Arnold networks (KANs), semi-parametric neural architectures, model complex functions efficiently using spline layers. Kurkova Kolmogorov-Arnold networks (KKANs) extend KANs by replacing the early spline layers with multi-layer perceptrons that map inputs into higher-dimensional spaces before applying spline-based transformations, which yield more stable training and provide robust architectures for system modeling. By enhancing the KKAN architecture, we develop a novel learning algorithm, distance-aware error for Kurkova-Kolmogorov networks (K-DAREK), for efficient and interpretable function approximation with uncertainty quantification. Our approach establishes robust error bounds that are distance-aware; this means they reflect the proximity of a test point to its nearest training points. In safe control case studies, we demonstrate that K-DAREK is about four times faster and ten times more computationally efficient than Ensemble of KANs, 8.6 times more scalable than GP as data size increases, and 7.2% safer than our previous work distance-aware error for Kolmogorov networks (DAREK). Moreover, on real data (e.g., Real Estate Valuation), K-DAREK's error bound achieves zero coverage violations.**

***Index Terms*—Error bounds, neural networks, worst-case analysis, KKAN.**

I. Introduction

Neural networks (NNs) are recognized as powerful function approximators; however, their predictions can be unreliable, potentially leading to catastrophic consequences in safety-critical applications [1]. One promising direction for improving both reliability and interpretability is to incorporate splines into neural networks. Splines provide smooth, piecewise polynomial mappings that ensure local control and continuous derivatives, making them well-suited for modeling complex functions, particularly in control engineering and signal processing [2]–[6]. Spline-based neural networks can leverage these properties to produce smoother function approximations than conventional Multi-layer perceptrons (MLPs). Furthermore, spline-based neural networks offer improved interpretability and thus greater reliability [7]. However, producing robust and trustworthy uncertainty estimates remains difficult. Two predominant strategies for uncertainty estimation are probabilistic methods and worst-case (e.g., adversarial or robust) formulations.

Probabilistic methods are generally accurate in data-rich domains, particularly when the goal is to model stochastic behavior. Common methods for probabilistic uncertainty estimation include Gaussian processes (GPs) [8] and Bayesian neural network approaches such as Monte Carlo dropout [9], variational inference [10], and Ensembles [11]. Interpreting the output of these methods often requires calibration, which presents challenges, including that their distributions are typically unbounded, unscaled, and valid only for the observed portion of the input space [12]–[15]. Additionally, GPs, which are popular in uncertainty estimation, are computationally expensive and also rely on kernel hyperparameters [8]. On the other hand, worst-case or robust approaches, which are particularly relevant for safety-critical applications, offer certainty guarantees under specified constraints or perturbations [16]–[18]. These models are often easier to understand, debug, and implement. While they can sometimes be conservative and face challenges in scaling to high-dimensional or complex distributions, they remain a practical and reliable choice when system constraints are known as hard constraints instead of prior distributions [17], [19].

Intuitively, uncertainty estimators must be *distance-aware*, meaning their confidence reflects the proximity to training data [20]. An interpretable uncertainty estimator is expected to demonstrate higher predictive confidence in regions near the training data, while exhibiting increased uncertainty as the input diverges from the observed distribution [11], [17], [20]–[22]. Uncertainty estimation in nonparametric models—such as GPs, splines, k-Nearest Neighbors, kernel density estimation [23]—can be easily made distance-aware because they depend on stored training data, whereas typical parametric models—such as Ensembles [11], and support vector machines—do not estimate distance-aware uncertainty.

More recently, Kolmogorov-Arnold network (KAN) [7] introduced a novel approach that combines nonparametric models (splines) with parametric models [17]. KAN, inspired by Kolmogorov-Arnold Theorem [24], proposed to model complex functions with a deep neural network using spline basis as activation functions. Building on this success,

This work is supported by the National Science Foundation under Grant No. 2218063 and the Gleason Endowment at RIT.

Kurkova Kolmogorov-Arnold networks (KKANs) [25] reduced the complexity of the KAN architecture by replacing its arbitrary-depth spline network structure with a simple two-block architecture, combining an MLP-based inner block with flexible basis functions as the outer block. This results in faster, more stable training and improved function approximation.

However, neither KAN nor KKAN provides distance-aware uncertainty estimates, even though both rely heavily on nonparametric components in their architectures. In prior work [17], we resolved this issue for KANs; here, we further generalize the approach by equipping KKANs with distance-aware uncertainty. Our modified KKAN architecture consists of two blocks: an MLP Block and a Spline Block (see Figure 1). In the MLP Block, each input dimension is transformed through a spectrally normalized MLP layer [20], [26] to produce higher-dimensional representations. The resulting representations are summed dimension-wise to form a unified feature vector. Finally, the Spline Block maps this feature vector to the output via a linear combination of spline functions. This unique structure enables learning the MLP Block with a desired Lipschitz smoothness. We then apply the spline error analysis from our previous work [17] to the Spline Block to compute the error bound for the joint KKAN architecture.

The term "spline" originates from the flexible wooden strips once used by shipbuilders, which were shaped by anchoring them to fixed control points (or knots). In spline error analysis, this idea translates into constraining spline knots to a subset of the training data and estimating the approximation error at a test point as a function of its distance to the nearest knots. Within this framework, our prior work [17], Distance-aware uncertainty for Kolmogorov networks (DAREK), offers a structured, bottom-up framework for uncertainty quantification, allowing error diagnosis at the unit level of a neural network. However, it lacks a principled mechanism for Lipschitz division and error sharing; therefore, its performance is sensitive to heuristic design choices. Moreover, the composition of spline functions in DAREK, and in SNNs more broadly, can yield highly nonlinear behavior, often resulting in non-smooth approximations (which may be mitigated through regularization [7]) The hybrid architecture of Distance-aware uncertainty for Kurkova Kolmogorov networks (K-DAREK), which combines MLP and spline components, produces smoother function approximations by adopting simpler nonlinear transformations. Also, incorporating scaled spectral normalization introduces explicit Lipschitz control and confines the feature range of each MLP block. Together, these enhancements reduce the challenges of Lipschitz sharing and improve the stability and reliability of uncertainty estimates.

In summary, our goal is to estimate distance-aware worst-case uncertainty for neural networks efficiently. This is made possible by the integration of splines into deep neural networks. We extend our prior analysis [17] to KKAN, leading to the following contributions: 1) We propose *K-DAREK*, a novel framework for deriving worst-case error bounds that leverages the expressive power of MLPs together with spline-based structures to provide interpretable, distance-aware uncertainty estimates. 2) We compare K-DAREK against GPs, Ensembles and our previous work [17] across synthetic datasets, real-world regression datasets (e.g., Real Estate Valuation), and a safe multi-agent control task. We find that K-DAREK is about four times **faster** and ten times more computationally **efficient** than Ensembles, 8.6 times more **scalable** than GP by growing the data size, and achieves a 7.2% **improvement in safety** over our previous method, DAREK. Moreover, on real data, K-DAREK's error bound achieves zero coverage violations.

Additional simulations, extended related literature, a table of acronyms, and algorithmic complexity are provided in the **Appendix** [1].

II. Background

To understand our approach, it is important to understand a few background concepts in: a) KANs [7], b) distance-awareness [20], c) spectral normalization [26], and d) DAREK [17].

KANs [7]: Kolmogorov-Arnold theorem (KAT) states that any continuous multivariate function can be represented as a finite superposition of univariate functions and addition [24]. Formally, a continuous multi-variate function $f : \mathcal{X} \to \mathbb{R}$, $(\mathcal{X} \subset \mathbb{R}^d)$ can be represented as:

$$f(\mathbf{x}) = \sum_{q=1}^{2d+1} \Phi_q\Big(\sum_{p=1}^{d} \psi_{p,q}(x_p)\Big), \tag{1}$$

where $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^d$ is a d-dimensional input vector and x_p is its p-th component. In this formulation, Φ_q and $\psi_{p,q}$ are univariate functions corresponding to the nodes of outer and inner layers, respectively. KAN [7] relaxes the two-layer structure by allowing arbitrary depth and a flexible number of univariate functions. These functions are replaced with B-splines, leading to a hierarchical representation of a vector-value function $\mathbf{f} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_m(\mathbf{x}))$ where each component $f_r : \mathcal{X} \to \mathbb{R}$ for $r \in \{1, 2, \cdots, m\}$, as follows:

$$f_r(\mathbf{x}) = \sum_{i_L=1}^{N_L} s_{L,r,i_L}\Big(\sum_{i_{L-1}=1}^{N_{L-1}} \mathcal{S}_{L-1,i_L,i_{L-1}}(\mathbf{x})\Big), \text{where,}$$

$$\mathcal{S}_{l,j,i}(\mathbf{x}) = s_{l,j,i}\Big(\cdots \sum_{i_2=1}^{N_2} s_{2,i_3,i_2}\Big(\sum_{i_1=1}^{N_1} s_{1,i_2,i_1}(\mathbf{x}_{i_1})\Big)\Big). \tag{2}$$

Here, $s_{l,j,i}$ denotes a k-th order B-spline defined by its knots, located in layer l, projecting input i to contribute to output j, and N_l for $l \in \{1, 2, \cdots, L\}$ represent the input dimension and hidden layer dimensions up to the last layer of the network.

Distance-awareness: The notion of distance-awareness as introduced by J. Liu et al. [20] and extended to nearest neighbor distance by Ataei et al. [17], is defined as follows:

Definition 1 (Distance-awareness [17]): An approximate function $\hat{f}(\mathbf{x})$, defined over $\mathbf{x} \in \mathcal{X}_{\text{IND}}$ is considered **distance-aware** if its associated error or uncertainty $u_{\hat{f}}(\mathbf{x})$ increases monotonically with the distance of a test point from the training dataset $\mathcal{X}_{\text{IND}}$. The distance is quantified by a function

[1] https://arxiv.org/abs/2510.22021

$d(\mathbf{x}, \mathcal{X}_{\text{IND}}) = \min_{\mathbf{x}' \in \mathcal{X}_{\text{IND}}} d(\mathbf{x}, \mathbf{x}')$, which computes the minimum distance between the test point and all training samples.

Spectral normalization: Spectral normalization [20] enforces a desired Lipschitz continuity for a single fully-connected neural network layer by normalizing its weight matrix W using its spectral norm (largest singular value). Spectral normalization is most straightforward to implement with the ReLU activation function, which has a Lipschitz constant of 1. Therefore, for a linear layer followed by a ReLU activation, defined as $h_l(\mathbf{x}) = \text{ReLU}(W_l\mathbf{x} + \mathbf{b}_l)$, the Lipschitz constant of h_l can be bounded as follows:

$$\begin{aligned}\mathcal{L}_{h_l} &= \frac{\|h_l(\mathbf{x}) - h_l(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \|(W_l\mathbf{x} + \mathbf{b}_l) - (W_l\mathbf{x}' + \mathbf{b}_l)\| \\ &= \frac{\|W_l(\mathbf{x} - \mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \sigma_{\max}(W_l). \end{aligned} \tag{3}$$

where $\sigma_{\max}(W_l)$ is the largest singular value of the matrix W_l. By normalizing the weight matrix, which constrains $\sigma_{\max}(W_l) \leq \mathcal{L}_{h_l}$, we ensure that each layer is Lipschitz continuous with the desired Lipschitz constant $\mathcal{L}_{h_l}$. Specifically, during training, we normalize the weights of the neural network after every training iteration as described in [20]

$$\bar{W}_l^{(t+1)} = \begin{cases} \frac{\mathcal{L}_{h_l} W_l^{(t)}}{\sigma_{\max}(W_l^{(t)})} & \text{if } \mathcal{L}_{h_l} < \sigma_{\max}(W_l^{(t)}) \\ W_l^{(t)} & \text{otherwise .} \end{cases} \tag{4}$$

In this paper, we refer to an MLP with ReLU activation functions in the hidden layers, a linear activation in the output layer, and scaled, spectrally normalized weights as a Spectral normalized ReLU-MLP (SNR-MLP).

DAREK [17]: This subsection reviews the background needed for spline error analysis. We denote the *Newton's polynomial operator* of order k as defined by [27, p7] on a sorted set of knots of $\boldsymbol{\tau}$ with m_k elements by $\mathcal{P}_{k,n}[\boldsymbol{\tau}, f(\boldsymbol{\tau})](x)$. We use $k+1$ nearest knots to $\boldsymbol{\tau}[n]$ from $\boldsymbol{\tau}$, where $\boldsymbol{\tau}[n]$ denotes the n-th element of set or vector $\boldsymbol{\tau}$, and the true output values are determined by $f(\boldsymbol{\tau})$. We define *k-th-order Lipschitz constant*, $\mathcal{L}_f^k$, of a $k-1$ times differentiable function $f : [a, b] \to \mathbb{R}$ as the ratio of the maximum change in the $(k-1)$-th derivative of f to the change in the input,

$$\frac{|f^{(k-1)}(x) - f^{(k-1)}(y)|}{d(x, y)} \leq \mathcal{L}_f^k \quad \forall x \neq y. \tag{5}$$

A *piecewise polynomial* $\hat{f}(x)$ of order k on $\boldsymbol{\tau}$ is constructed by dividing domain into $m_k - 1$ intervals, each associated with a distinct polynomial segment $\hat{f}_{[j]}(x)$ of order k, as follows:

$$\begin{aligned} &\hat{f}(x) = \sum_{i=0}^{k} c_{i,j} x^k =: \hat{f}_{[j]}(x) \qquad \forall x \in [\tau_j, \tau_{j+1}), \\ &\text{s. t.} \quad \hat{f}_{[j]}(\tau_{j+1}) = \hat{f}_{[j+1]}(\tau_{j+1}). \end{aligned} \tag{6}$$

Here, $\hat{f}_{[j]}(x)$ denotes the j-th polynomial component. Piecewise polynomials are continuous but not necessarily smooth.

Theorem 1 (Interpolation error [17]): For a function f with $k+1$ continuous derivatives (that is $f \in C^{(k+1)}$) and its $k+1$-th-order Lipschitz constant is $\mathcal{L}_f^{k+1}$, the error at test point $x \in [\boldsymbol{\tau}[j], \boldsymbol{\tau}[j+1])$ is bounded as follows:

$$|f(x) - \mathcal{P}_{k,j}[\boldsymbol{\tau}, f(\boldsymbol{\tau})](x)| \leq \underbrace{\frac{\mathcal{L}_f^{k+1}}{(k+1)!} \left| \prod_{i=j}^{j+k} (x - \boldsymbol{\tau}[i]) \right|}_{=: \bar{u}_f(x; \boldsymbol{\tau})}. \tag{7}$$

This result assumes the approximation exactly matches the function at the knot points, $\boldsymbol{\tau}[j]$. However, in practical scenarios-such as when noise is present-exact matching at knots is generally not achievable. The following result extends the error bound to account for spline approximations that incur non-zero error at the knots.

Theorem 2 (Spline error [17]): Under the assumptions of Theorem 1, consider a piecewise polynomial approximation $\hat{f}(x)$ and the error function is defined as $e_j^f(x) := f(x) - \hat{f}_{[j]}(x)$ 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,

$$|f(x) - \hat{f}(x)| \leq \bar{u}_f(x; \boldsymbol{\tau}) + |\mathcal{P}_{k,j}[e_j^f](x)| =: u_f(x; \boldsymbol{\tau}), \tag{8}$$

The conditions under which these error bounds are tight are detailed in [17].

The error of one output of a spline layer, where N_s splines contribute to the output, is computed as $u_{\text{sp}} = \sum_{i=1}^{N_s} u_{s_i}(\mathbf{x}, \mathcal{T})$. We also revisit the two-layer error bound presented in the DAREK paper [17], as our K-DAREK architecture also adopts a two-block structure, comprising an inner MLP Block followed by a Spline Block. The propagated error of a two-layer network, $\hat{f}(\mathbf{x}) = h_2(\mathbf{h}_1(\mathbf{x}))$, can be calculated using the following equation [17, Theorem 2]:

$$|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq u_{h_2}\big(\mathbf{h}_1(\mathbf{x}); \mathbf{h}_1(\mathcal{T})\big) + \mathcal{L}_{h_2}^1 \mathbf{1}_{\mathbf{h}_1}^\top \mathbf{u}_{\mathbf{h}_1}(\mathbf{x}; \mathcal{T}), \tag{9}$$

which shows that the error from earlier layers is scaled by the Lipschitz constant of the current layer, $\mathcal{L}_{h_2}^1$, and then added to the current layer's error. Here, $\mathbf{1}_{\mathbf{h}_1}$ is a vector of all ones with the same size as $\mathbf{h}_1$. In (9) $\mathcal{T}$ is a matrix of knots of size $N_s \times m_k$, where m_k defines knots for each of the N_s splines in layer h_1. This two-layer error propagation has been extended to arbitrarily deep spline networks [17].

Next, we introduce the hybrid K-DAREK architecture, which integrates MLP and spline components, and drives its error bound using Lipschitz continuity.

III. Architecture and Method

This section details the K-DAREK architecture and presents our approach for computing its worst-case error bound.

To develop our error analysis, we adopt the KKAN structure with a Spline Block. Building upon KAN, KKAN [25] insists on the two-layer structure of KAT (1) as a composition of two functional blocks, where ψ represents the inner block and Φ the outer block. The spectrally normalized MLPs in the inner block expand each input dimension into a higher-dimensional space through basis functions, such as Chebyshev basis functions, project this representation into a hidden space using the MLP,

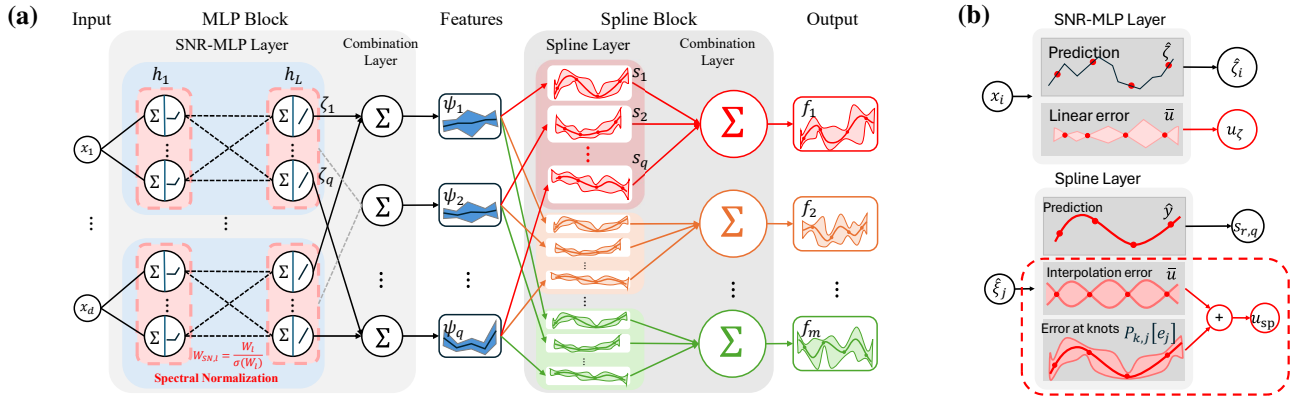

Fig. 1. **(a)** K-DAREK architecture, consists of two blocks: MLP and Spline. The MLP Block contains one SNR-MLP for each input dimension. A combination layer unifies the output of SNR-MLPs into a feature vector. The Spline Block is a Spline Layer followed by a combination layer that constructs the model's final output. **(b)** The error analysis of K-DAREK, has two main components. The SNR-MLP Layer receives inputs $\mathbf{x}$ and computes the feature map of SNR-MLPs, ζs, along with the linear error (13). The Spline Layer employs features $\boldsymbol{\psi}$ to calculate the spline values $s_{r,q}$ and spline error using interpolation error and error at the knot as described in (9).

and finally map it to the output space via a combined output layer. The structure of KKAN can be written as:

$$f_r(\mathbf{x}) = \sum_{i_s=1}^{q} s_{r,i_s}\Bigg(\sum_{p=1}^{d} \underbrace{\mathbf{1}_{hn}^{\top}\mathbf{T}_n\big(\mathrm{MLP}_{p,i_s,h}(\mathbf{T}_n(x_p))\big)}_{\psi_{p,i_s}(x_p)}\Bigg), \tag{10}$$

where $\mathbf{T}_n(\mathbf{y}) := (T_0(\mathbf{y}), \ldots, T_{n-1}(\mathbf{y})) \in \mathbb{R}^{hn}$ is the Chebyshev basis operator that expands each dimension of an input vector $\mathbf{y} \in \mathbb{R}^h$ to the first n Chebyshev polynomials of the first kind, $T_n(\mathbf{y}) := \cos(n \arccos(\mathbf{y})) \in \mathbb{R}^h$. We assume that $T_n(\mathbf{y})$ applies element-wise when operating on a vector. Also, s_{r,i_s} represents the spline that maps the i_s-th dimension of the intermediate layer to the output dimension f_r. $\mathrm{MLP}_{p,i_s,h} : \mathbb{R}^n \to \mathbb{R}^h$ is an MLP that produces a vector output of a desired dimension h. In K-DAREK, we introduce two modifications to KKAN. First, we restrict the Chebyshev operator to the first two degrees of Chebyshev functions, $T_0(\mathbf{y}) = 1$ and $T_1(\mathbf{y}) = \mathbf{y}$. This simplification makes error analysis more efficient, while trading off the model's expressiveness. Second, we use a spectrally normalized MLP instead of $\mathrm{MLP}_{p,i_s,h}$ in (10), which allows us to control the Lipschitz constant of the MLP.

The proposed architecture, as illustrated in Fig. 1, consists of two components: **a)** the main model architecture and **b)** the error analysis framework.

A. Main model architecture

The model architecture consists of two building blocks: MLP Block and Spline Block, as shown in Fig. 1.

MLP Block: There are d spectral-normalized ReLU activated MLPs (SNR-MLPs) followed by a combination layer. Each SNR-MLP processes a single input component x_p independently and maps it to a feature map $\boldsymbol{\psi}_p(x_p) = [\psi_{p,i_s}(x_p)]_{i_s=1}^{q} \in \mathbb{R}^q$ (see (10)). The combination layer then aggregates $\boldsymbol{\psi}_p(x_p)$ across the input dimensions to produce the final feature vector $\boldsymbol{\xi} = \sum_{p=1}^{d} \boldsymbol{\psi}_p(x_p) \in \mathbb{R}^q$.

Spline Block: As shown in Fig. 1 **(a)** and (10), the Spline Layer in this block consists of m groups, each containing q univariate splines that produce outputs $\mathbf{s}_r = (s_{r,1}, s_{r,2}, \cdots, s_{r,q})$. The combination layer then aggregates each group by summing its spline outputs, resulting in a vector $\mathbf{f} = \sum_{i_s=1}^{q} s_{r,i_s} \in \mathbb{R}^m$.

B. Error Analysis framework for K-DAREK

We divide the error analysis framework into five components described in this subsection. Alg. 1 provides a brief outline of the overall error framework.

Algorithm 1: K-DAREK

Data: Input-feature-output tuple $\{\mathcal{T}, K, Y\}$, trained model $\hat{f}$, test point $\mathbf{x}^*$, order k, Lipschitz constants $\mathcal{L}_f^{(k+1)}$, $\mathcal{L}_f^{(1)}$.

Precompute errors $E^f = |Y - \hat{f}(\mathcal{T})|$
Divide $\mathcal{L}_f^1$ equally among layers ($\mathcal{L}_h = \sqrt[L]{\mathcal{L}_f / d}$).
Find feature of test input $\boldsymbol{\psi}^* = \Psi(\mathbf{x}^*)$
Find $\mathcal{L}_{\mathrm{sp}} = \mathcal{L}_h / N_{\mathrm{sp}}$ and $\mathcal{L}_{\mathrm{MLP}} = (\mathcal{L}_h)^D$
Find u_{MLP} using [Eq. 13]
for $n \in \{1, \ldots, N_{sp}\}$ **do**
    /* $O(\log_2(m_k))$ `binary search`. */
    Find j such that $\boldsymbol{\kappa}_n[j] \le \boldsymbol{\psi}_n < \boldsymbol{\kappa}_n[j+1]$.
    Fit Newton's Polynomials $\mathcal{P}_{k,j}[\mathbf{e}_n^f]$ on the knots $\{(\boldsymbol{\kappa}_n[i], E_i^f)\}_{i=j}^{j+k}$
    Find $u_{\mathrm{sp}_n}(x_n^*; \boldsymbol{\tau}_n)$ [Eq. 8].
Compute error bound over f [Eq. 16]

E is matrix of size Y and $\mathbf{e}_i$ is i-th column of E.
$\boldsymbol{\kappa}_n[j]$ is j-th element of n-th column of K.

Knot Selection: Each spline in Spline Layer requires m_k knots, and each group contains q univariate splines, forming a knot matrix $K \in \mathbb{R}^{m_k \times q}$, where column c corresponds to the knots of the c-th spline in each group, $\boldsymbol{\kappa}_c$. To construct this matrix, we select m_k samples from the training data during network training, forming a data matrix $\mathcal{T} \in \mathbb{R}^{m_k \times d}$, where column c represents the c-th dimension of the selected samples, denoted by $\boldsymbol{\tau}_c$. Corresponding to these input samples, we define a label matrix $Y \in \mathbb{R}^{m_k \times m}$, where column c contains the true target values for the c-th output dimension, $\mathbf{y}_c$. The knots serve as representatives of the training data, and we use the actual error at the knot locations to compute the *error at knots*, the error bound. Note that matrices $\mathcal{T}$ and Y are paired samples selected

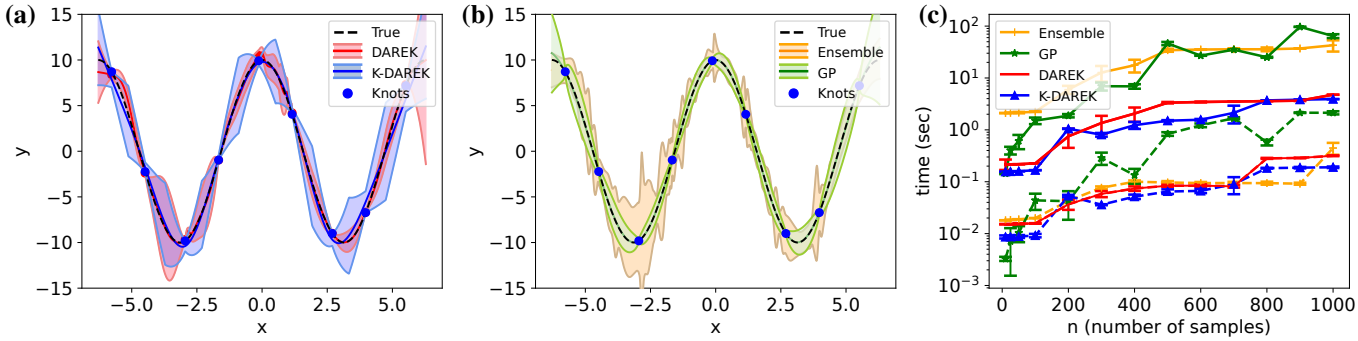

Fig. 2. Cosine function approximation experiment. **(a)** Error estimation comparison of DAREK and K-DAREK, showing that K-DAREK achieves smoother function fitting without requiring an explicit regularizer. In both models, the error bounds increase as test points move further from the nearest knot, reflecting distance-awareness in the error behaviour. **(b)** 3σ uncertainty bounds comparison between an Ensemble of KANs and a GP. The Ensemble's uncertainty bounds lack distance-awareness and do not follow a consistent spatial pattern, in contrast to the DAREK, K-DAREK, and GP. **(c)** Process time comparison of K-DAREK (green), DAREK (orange), GP (red), and Ensemble (blue) models. Training time (solid line) and inference time (dashed). The error bars indicate the variability observed across repeated runs of the experiment.

from training data, and to calculate the knot matrix K, we pass $\mathcal{T}$ through the network and collect the resulting features.

Error of MLP Block: Each SNR-MLP has D layers, where the l-th layer has an associated Lipschitz constant $\mathcal{L}_l = \sqrt[L]{\mathcal{L}_f/d}$. Although this uniform Lipschitz factorization is one possible factorization, it may not be the tightest. The overall Lipschitz constant of a single SNR-MLP is the product of the layer-wise Lipschitz constants: $\mathcal{L}_{\text{MLP}} \coloneqq \mathcal{L}_1\,\mathcal{L}_2\,\cdots\,\mathcal{L}_D$. Given a test point $\mathbf{x}^*$, the error of the i-th SNR-MLP is:

$$u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) = \mathcal{L}_{\text{MLP}_i}\,\|x_i^* - \tau_i^*\|,$$
$$\text{where, } \tau_i^* = \underset{\tau^\dagger \in \boldsymbol{\tau}_i}{\text{argmin}}\,\|\tau^\dagger - x_i^*\|. \tag{11}$$

Here, x_i^* is the i-th component of the test input $\mathbf{x}^*$, and τ_i^* is the i-th component of the closest sampled data to x_i^*. We obtain the worst error bound for the MLP Block by summing the individual bounds across all input dimensions:

$$u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) = \sum_{i=1}^{d} u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) = \sum_{i=1}^{d} \mathcal{L}_{\text{MLP}_i}\,\|x_i^* - \tau_i^*\|. \tag{12}$$

If all SNR-MLPs have equal Lipschitz constant $\mathcal{L}_{\text{MLP}}$, this expression simplifies to:

$$u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) = \mathcal{L}_{\text{MLP}} \sum_{i=1}^{d} \|x_i^* - \tau_i^*\|. \tag{13}$$

Note that each τ_i^* may originate from a different training sample data for different input dimensions, so further simplification beyond this equation is not possible.

Error of Spline Block: Denote a spline in the Spline Layer by $\text{sp}_{r,i}$, where r is the output index and i is the feature index. Then the set of knots of this spline is $\boldsymbol{\tau}_i$ and the corresponding output values required for error calculation are $\mathbf{y}_r$. Given this, for a test point $\mathbf{x}^*$ and its corresponding feature $\boldsymbol{\xi}^*$, the spline error in K-DAREK framework is defined as:

$$u_{\text{sp}_{r,i}}(\xi_i^*, \boldsymbol{\kappa}_i) \coloneqq \bar{u}_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i) + \left\|\mathcal{P}[(\hat{f}(\boldsymbol{\tau}_r) - \mathbf{y}_r)^{\text{sp}}_{[j]}](\xi_i^*)\right\|. \tag{14}$$

The worst error bound of the r-th group of splines is given by:

$$u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) \coloneqq \sum_{i=1}^{q} u_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i). \tag{15}$$

Total Error: Having established the error for both the MLP and Spline Blocks in Equations (13) and (15), we can now compute the overall error of K-DAREK. The total error of r-th component of f at a test point $\mathbf{x}_*$ is given by:

$$u_{f_r}(\mathbf{x}_*) = u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) + \mathcal{L}_{\text{sp}}^1 u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}). \tag{16}$$

Here, $\mathcal{L}_{\text{sp}}^1$ denotes the Lipschitz constant of the Spline Block, which propagates the uncertainty from the MLP Block into the Spline Block. This formulation accounts for both interpolation and transformation errors within the model architecture.

Lipschitz and Error Sharing: To use the provided error analysis and ensure that the neural network preserves the stability and bounded sensitivity of the true system, the approximated model must exhibit Lipschitz continuity consistent with the true dynamics $\mathcal{L}_f$. To achieve this, we divide the Lipschitz constant equally across all L layers, both MLP and the spline layers, by setting each layer a Lipschitz constant of $\mathcal{L}_h = \sqrt[L]{\mathcal{L}_f/d}$, where d is the input dimension, inspired by [20]. The division by d indicates SNR-MLPs are independent (one per input dimension). Additionally, we assumed the error at the knots arises solely from the Spline Block. Under this assumption, the total error is distributed equally among splines within each group. Various approaches, including random sharing and heuristic sharing, can be employed for Lipschitz and error sharing. We defer the study of these options to future research.

IV. Experiments and Results

We evaluated K-DAREK and compared it with other baselines in three experiments.

Function approximation: The first experiment is learning the function $f(x) = 10\cos(x)$ over the range $[-2\pi, 2\pi]$ using 50 training data. DAREK model has two layers: the first maps the input to a 5-dimensional latent space, and the second maps

it back to one output dimension using splines. Similarly, K-DAREK uses an inner block with a single SNR-MLP layer with five units and an outer block that linearly combines spline outputs to produce the final output. For the Ensemble, we use ten KAN models with the same architecture as DAREK. All models use cubic splines with 9 knots and are trained for 500 epochs with a learning rate of 0.1, which is decayed by a factor of 0.9 every 50 epochs. We select an RBF kernel for the GP and learn the kernel hyperparameters. Fig. 2 **(a)** compares the error estimation of DAREK and K-DAREK. The results suggest that K-DAREK achieves a more generalized fit, reducing error at knots. While the interpolation error might slightly increase due to the inner block's linear approximation (as opposed to higher-order polynomials), the overall fit is more robust. Fig. 2 **(b)** compares the uncertainty bounds produced by an Ensemble of KANs and GPs. We used the same input knots for the first-layer splines in DAREK, K-DAREK, and Ensemble, and trained the GP on these selected knots to ensure a fair comparison. In DAREK, the lack of architectural constraints can lead to erratic behavior, and the error bounds from earlier layers propagate through the model, making their control crucial. The Lipschitz constant helps justify and manage this propagation, and distributing this effect effectively is essential. Table (I) shows that K-DAREK achieves a *zero* violation rate, defined as the percentage of test points where the true value lies outside the predicted error bounds, while using the fewest learnable parameters. The mean square error (MSE) is calculated as $MSE = \sum_{i=1}^{n} |\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)|^2/n$, where n is the number of test points. The "Size" column shows the total number of learnable parameters. Fig. 2 **(c)** illustrates the process time complexity of DAREK and K-DAREK compared to GP and Ensemble. The results show that K-DAREK is slightly more efficient than DAREK across all dataset sizes and significantly outperforms both GP and Ensemble for datasets larger than 300 samples. These results are consistent with the algorithmic complexity analysis provided in Appendix C.

TABLE I
ERROR ESTIMATOR COMPARISON FOR THE FUNCTION APPROXIMATION EXPERIMENT. K-DAREK ACHIEVES BETTER FITTING WITH FEWER VIOLATIONS.

Model	MSE loss	Violations(%)	Width	Size
Ensemble	0.378	0.0	10×[1,5,1]	700
GP	**0.195**	0.0	N.A.	N.A.
DAREK	0.406	7.2	[1,5,1]	70
K-DAREK	0.623	**0.0**	[1,5]+[5,1]	**45**

Real Data Experiment: We evaluate model performance on the UCI Real Estate Evaluation dataset [28], and summarize the results in Table II. Training is performed on the standardized dataset with the Adam optimizer and a decaying learning rate, and each experiment is repeated 20 times. We report the mean RMSE and the percentage of violations for GP, two-layer DAREK (DK2), K-DAREK (K-DK2), and Ensemble of KANs (ENS). Among these models, K-DK2 achieves the lowest violation rate. Full experimental settings and complete results for more datasets are provided in Appendix E.

TABLE II
RESULTS OF TEST ERRORS AND PERCENTAGE OF VIOLATIONS ON THE REAL ESTATE EVALUATION DATASET.

Metric	GP	DK2	K-DK2	ENS
RMSE	**0.55 ± 0.13**	0.87 ± 0.16	0.68 ± 0.14	1.21 ± 0.62
Violations (%)	1.33 ± 1.14	**0.00 ± 0.00**	**0.00 ± 0.00**	4.22 ± 2.17

Multiagent safe control: Learning-based control integrates data-driven modeling techniques, such as neural networks, with traditional control theory to handle complex or partially unknown dynamical systems. These methods enhance adaptability and robustness, especially in environments where analytical models are difficult to derive, or the system must respond to uncertainty [29], [30].

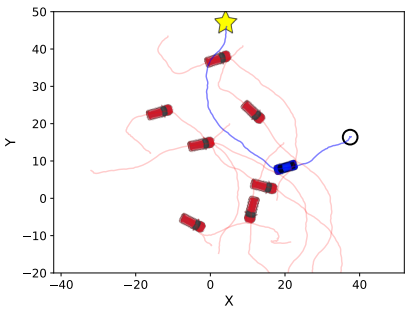

Fig. 3. Multiagent safe control setup. This is one of the successful trajectories using K-DAREK bounds, where the controlled agent safely navigates from the start position to the goal location. The controlled agent and its path are shown in blue, and the surrounding red vehicles represent other agents.

In this experiment, we evaluated both DAREK and K-DAREK in a multiagent safe control setup adopted from [31]. As shown in Fig. 3, the controlled agent (blue car) must safely navigate from its initial position (circle) to the goal (star) while avoiding collisions with other agents (red cars) that follow unknown control policies. The blue car observes only the states of other agents. Its dynamics are discrete control-affine, given by:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) + g(\mathbf{x}_t)\mathbf{u} + d(\mathbf{x}_t), \tag{17}$$

where state $\mathbf{x} \in \mathbb{R}^4$ contains position and velocity in x and y directions. Here, f denotes the nominal drift dynamics, g is the control effectiveness matrix, $\mathbf{u}$ is the control input (accelerations in x and y directions), and d is the disturbance. The control policy first computes an optimal but potentially unsafe trajectory using a model predictive control (MPC) controller that ignores obstacles. Then, following [31, eq. 17], a control barrier functions (CBF)-based controller is applied to compute the closest safe control input that respects the obstacle constraints. A quadratic program is formulated for the CBF-based optimization problem, where the disturbance d is bounded inside a polytope. We use the worst-case error bounds predicted by the two-layer DAREK (DK2), two-layer K-DAREK (K-DK2), and three-layer K-DAREK (K-DK3) to define this polytope. DK2 has a hidden unit of 5, K-DK2 has

TABLE III
RESULTS FOR MULTIAGENT SAFE CONTROL EXPERIMENT. OVERALL, K-DAREK MODELS ACHIEVE LOWER COLLISION RATES COMPARED WITH DAREK.

	$\bar{d}_a = \bar{d}_v = \bar{d}_p = 15\%$			Average: $\bar{d} \in \{5\%, 10\%, 15\%\}$		
Model	DK2	K-DK2	K-DK3	DK2	K-DK2	K-DK3
Success	96.0	96.0	92.0	95.1	95.4	95.6
Collision	0.0	0.0	0.0	1.1	0.7	0.7
Stuck	4.0	4.0	8.0	3.7	3.9	3.7

the SNR-MLP of size [2,5] and a spline layer of [5,4], and K-DK3 has an SNR-MLP of size [2,5,5] and a spline layer of size [5,4]. We evaluate performance over 50 trials, recording the number of **success** runs-reaching the goal without colliding with any other car, **collision**-crashes into another car, and cases where the agent becomes **stuck**-fails to reach the goal, either due to being blocked (e.g., deadlock) or overly conservative behavior resulting from excessive uncertainty estimation. All the models are trained offline for 300 epochs on 5000 noise-free data points collected. We report the experiment with 15% noise level (uniform) added to control and observations, and the average over all experiments with different noise levels in Table III. The comprehensive results are provided in Appendix E. Results show that, on average, K-DAREK achieves a slightly higher success rate and fewer collisions, consistent with the 1D experiment, which demonstrated that K-DAREK is more conservative and has fewer violations.

V. CONCLUSION

In this work, we presented K-DAREK, a principled approach for quantifying distance-aware worst-case error in hybrid models that combine deep dense neural networks and spline-based networks. Similar to KAT, we focus on component-wise distance-awareness, while the analysis for the higher-dimensional distance-awareness is left for future work. We derived an explicit error bound for the MLP block using its Lipschitz constant, linking test point uncertainty to proximity with training data. For the spline block, we redefined the DAREK formulation to align with this purpose based on the selection of knots derived from the training set and the accuracy of predicted values at those knots. By combining these components, we obtained an overall model error bound that captures both local interpolation uncertainty and propagated transformation error through the model. This framework provides a theoretically grounded understanding of prediction and error estimation reliability and computational efficiency, paving the way for robust model design in settings that demand certified error bounds or risk-aware predictions.

REFERENCES

[1] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *NeurIPS*, vol. 32, 2019.

[2] Y. Wang and G. Wahba, "Bootstrap confidence intervals for smoothing splines and their comparison to bayesian confidence intervals," *Journal of Statistical Computation and Simulation*, vol. 51, pp. 263–279, 1995.

[3] M. Egerstedt and C. Martin, *Control theoretic splines: optimal control, statistics, and path planning*. Princeton University Press, 2009.

[4] M. Unser, "Splines: A perfect fit for signal and image processing," *IEEE Signal processing magazine*, vol. 16, no. 6, pp. 22–38, 2002.

[5] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *IEEE/CVF CVPR*, 2018, pp. 869–877.

[6] X. Wang, J. Shen, and D. Ruppert, "On the asymptotics of penalized spline smoothing," *Elec. Journal of Statistics*, vol. 5, 2011.

[7] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov–arnold networks," in *ICLR*, 2025.

[8] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. The MIT Press, 2006.

[9] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.

[10] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *ICML*, 2015, pp. 1613–1622.

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

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML*, 2017, pp. 1321–1330.

[13] X. Jiang and X. Deng, "Knowledge reverse distillation based confidence calibration for deep neural networks," *Neural Processing Letters*, vol. 55, no. 1, pp. 345–360, 2023.

[14] R. Cheng, R. M. Murray, and J. W. Burdick, "Limits of probabilistic safety guarantees when considering human uncertainty," in *IEEE ICRA*, 2021, pp. 3182–3189.

[15] M. Ataei and V. Dhiman, "DADEE: Well-calibrated uncertainty quantification in neural networks for barriers-based robot safety," *arXiv:2407.00616*, 2024.

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

[17] M. Ataei, M. J. Khojasteh, and V. Dhiman, "DAREK-Distance aware error for Kolmogorov networks," in *IEEE ICASSP*, 2025, pp. 1–5.

[18] J. A. Lopez and V. Kreinovich, "Towards decision making under interval uncertainty," in *NAFIPS*. Springer, 2023, pp. 335–337.

[19] K. Long, V. Dhiman, M. Leok, J. Cortés, and N. Atanasov, "Safe control synthesis with uncertain dynamics and constraints," *IEEE RAL*, vol. 7, no. 3, pp. 7295–7302, 2022.

[20] 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," *NeurIPS*, vol. 33, pp. 7498–7512, 2020.

[21] J. Mukhoti, A. Kirsch, J. van Amersfoort, P. H. Torr, and Y. Gal, "Deep deterministic uncertainty: A new simple baseline," in *IEEE/CVF CVPR*, 2023, pp. 24 384–24 394.

[22] J. Van Amersfoort, L. Smith, Y. W. Teh, and Y. Gal, "Uncertainty estimation using a single deep deterministic neural network," in *ICML*, 2020, pp. 9690–9700.

[23] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[24] J. Schmidt-Hieber, "The Kolmogorov–Arnold representation theorem revisited," *Neural networks*, vol. 137, pp. 119–126, 2021.

[25] J. D. Toscano, L.-L. Wang, and G. E. Karniadakis, "KKANs: Kurkova-kolmogorov-arnold networks and their learning dynamics," *Neural Networks*, vol. 191, p. 107831, 2025.

[26] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv:1802.05957*, 2018.

[27] C. De Boor and C. De Boor, *A practical guide to splines*. Springer, 1978, vol. 27.

[28] A. Frank. (2010) UCI machine learning repository.

[29] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, "Control barriers in bayesian learning of system dynamics," *IEEE TAC*, vol. 68, no. 1, pp. 214–229, 2021.

[30] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[31] R. Cheng, M. J. Khojasteh, A. D. Ames, and J. W. Burdick, "Safe multi-agent interaction through robust control barrier functions with learned uncertainties," in *IEEE CDC*, 2020, pp. 777–783.