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 parametric and powerful tools for function approximation, and the choice of architecture heavily influences their interpretability, efficiency, and generalization. In contrast, Gaussian processes (GPs) are nonparametric probabilistic models that define distributions over functions using a kernel to capture correlations among data points. However, these models become computationally expensive for large-scale problems, as they require inverting a large covariance matrix. Kolmogorov-Arnold networks (KANs), semi-parametric neural architectures, have emerged as a prominent approach for modeling complex functions with structured and efficient representations through spline layers. Kůrkovà Kolmogorov-Arnold networks (KKANs) extend this idea by reducing the number of spline layers in KAN and replacing them with Chebyshev layers and multi-layer perceptrons, thereby mapping inputs into higher-dimensional spaces before applying spline-based transformations. Compared to KANs, KKANs perform more stable convergence during training, making them a strong architecture for estimating operators and system modeling in dynamical systems.**

By enhancing the KKAN architecture, we develop a novel learning algorithm, distance-aware error for Kůrkovà-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. Through case studies on a safe control task, we demonstrate that K-DAREK is about four times faster and ten times higher computationally efficiency than Ensemble of KANs, 8.6 times more scalable than GP by increasing the data size, and 50% safer than our previous work distance-aware error for Kolmogorov networks (DAREK). [1]

***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. 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 [1]–[3]. Spline-based neural networks can leverage these properties to produce smoother function approximations than conventional Multi-layer perceptrons (MLPs). However, despite these advances, ensuring reliable uncertainty estimates remains challenging. Two predominant approaches have been developed to address this issue: 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 average behavior. Common approaches for probabilistic uncertainty estimation include Gaussian processes (GPs) [4] and Bayesian neural networks such as Ensembles [5]. Interpreting the output of these methods often requires calibration, which presents challenges, including the distributions being typically unbounded, unscaled, and valid only for the observed portion of the input space [6]–[9]. Additionally, GPs, which are popular in modeling regression tasks and uncertainty estimation, are computationally expensive and also rely on kernel hyperparameters [4]. On the other hand, worst-case or robust approaches—particularly relevant for safety-critical applications—offer certainty guarantees under specified constraints or perturbations [10]–[12]. Distance-aware uncertainty for Kolmogorov networks (DAREK) [11] is a neural-level approach to uncertainty estimation in spline neural networks (SNNs) that estimates the uncertainty of each neuron and then builds up the uncertainty on the full network. These models are often easier to understand, debug, and implement. While they can sometimes be conservative and face challenges in scaling to very high-dimensional or highly complex distributions, they remain a practical and reliable choice when the constraints on the systems are known as hard constraints instead of prior distributions [11], [13].

Intuitively, uncertainty estimators must be *distance-aware*, meaning their confidence reflects the proximity to training data [14]. 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 [5], [11], [14]–[16]. Uncertainty estimation in non-parametric models—such as GPs, splines, k-Nearest Neighbors, kernel density estimation [17]—can be easily made distance-aware because

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

[1]Code available at https://github.com/Masoud-Ataei/KDAREK/

TABLE I
EXPANSIONS OF IMPORTANT ACRONYMS.

Acronym	Expansion
NN	Neural network
KAT	Kolmogorov-Arnold theorem
KAN	Kolmogorov-Arnold network
KKAN	Kůrkovà Kolmogorov-Arnold network
MLP	Multi-layer perceptron
ReLU	Rectified linear unit
MSE	Mean square error
SNR-MLP	Spectral-normalized ReLU activated MLP
GP	Gaussian process
DKL	Deep kernel learning
DAREK	Distance-aware uncertainty for Kolmogorov networks
K-DAREK	Distance-aware uncertainty for Kůrkovà Kolmogorov networks
SM	Spectral mixer
MPC	Model predictive control
CBF	Control barrier functions

they depend on stored training data. For the same reason, typical parametric models—such as Bayesian neural networks (BNNs), Ensembles [5], and support vector machines—do not estimate distance-aware uncertainty.

More recently, Kolmogorov-Arnold network (KAN) [18] combined non-parametric models (splines) with parametric models in a novel way. KAN [18], inspired by Kolmogorov-Arnold Theorem [19], proposed to model complex functions with a deep neural network using spline basis as activation functions. Building on KANs success, Kůrkovà Kolmogorov-Arnold networks (KKANs) [20] reduced the complexity of KAN architecture by replacing the arbitrary depth spline network structure with a simple two-block architecture. This results in faster and stable training with more accurate function approximation. The two-block KKAN architecture integrates MLP-based inner block with flexible basis functions as the outer block.

However, neither KAN nor KKAN estimates distance-aware uncertainty in their predictions, despite extensively using non-parametric models in their architecture. We addressed this limitation with KAN in our previous work [11]. We extend KKANs in this work by introducing 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 [14], [21] to produce higher-dimensional representations. Then the resulting representations are dimension-wise summed to form a unified feature vector. 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 [11] to the Spline Block to compute the error bound for the joint KKAN architecture. The main idea for spline error analysis is to restrict the spline-knots to a subset of the training dataset and to estimate the error at any given test point as a function of its distance from the nearest knots[2].

DAREK offers a structured, bottom-up framework for uncertainty quantification, enabling the diagnosis of errors at the unit level of a neural network [11]. 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 spline neural networks, including DAREK, produces highly nonlinear functions, often leading to a non-smooth approximated function, which can be addressed by regularization [18]. The hybrid architecture of Distance-aware uncertainty for Kůrkovà Kolmogorov networks (K-DAREK), which combines MLP and spline components, produces smoother function approximations by adopting simpler nonlinear transformations. Additionally, incorporating scaled spectral normalization introduces explicit Lipschitz control and confines the feature range of each MLP block. These enhancements together mitigate 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 network models. This has been made possible by the recently introduced KAN and then KKAN architectures that enable the integration of splines in deep neural networks. In our previous work [11], we introduced uncertainty bounds for KAN. In this work, we extend our analysis to KKAN, leading to the following contributions: 1) We propose *K-DAREK*, a novel framework for worst-case error bounds, leveraging the inherent structure of KKAN model to provide interpretable distance-aware uncertainty estimates. 2) We evaluate and compare K-DAREK against GPs, Ensembles and our previous work [11]. We perform these evaluations on synthetic datasets and a safe control task. We find that K-DAREK is about 4 times faster and 10 times more computationally efficient than Ensembles, 8.6 times more scalable than GP by growing the data size, and 50% safer than our previous work DAREK.

II. RELATED WORK

A lot of work has been done on uncertainty estimation in neural networks. However, most of these models are probabilistic and not distance-aware. For a comprehensive survey, we refer the interested reader to [22], [23]. In this section, we focus on relevant discussions about closely related works, along with a historical perspective.

a) KAN and KKAN: Kolmogorov-Arnold theorem states that any multivariate continuous function can be represented as a finite decomposition of univariate continuous functions and addition [19]. There has been a long debate about whether or not the KAT is relevant to the design of neural networks. This controversy is exemplified by two influential papers: Girosi and Poggio argue that KAT lacks practical implications

[2]"Spline" originates from flexible wooden strips used by shipbuilders, constrained by control points (or knots).

for modern network architectures, as it does not account for the smoothness and generalization properties that are critical in practical learning systems [24]. Whereas Kůrkovà contends that the theorem provides foundational insight into the representational capabilities of neural networks [19], [25], [26]. Z. Liu et al. [18] address this issue by focusing on Spline NN with many layers. KANs [18] replace traditional fixed activation functions with learnable, spline-parametrized functions on edges, enhancing interpretability and adaptability. Kůrkovà introduced a different remedy for this three decades ago [25]. He suggested that the component function within the decomposition could be effectively approximated using a Multi-Layer Perceptrons (MLP) with small error, rather than KAT, where the aim was to achieve zero error.

b) Deep kernel learning: There are some similarities between Deep Kernel Learning (DKL), developed in the neural networks community, and Kernel methods with our design based on KKAN. Wilson et al. [27] introduced DKL by replacing the last layer of MLP with a kernel layer such as spectral mixture (SM) base kernels and learn the hyperparameters of kernels jointly with weights of the NN. This change combined the expressiveness of deep networks with the probabilistic properties of GPs and made the network behave similarly to a GP with the same kernel function. Similarly, KKAN can be viewed as an MLP where the final layer is replaced with a linear combination of splines. Architecturally, both DKL and KKAN share the idea of projecting features into the output space via a learned functional layer. However, their underlying mechanisms differ significantly. SM kernels in DKL architectures are composed of cosine functions that operate in the frequency domain, often exhibiting periodic behavior, as their behavior is governed by combinations of cosine functions in the frequency domain, lacking direct spatial or semantic correspondence. In contrast, spline-based representations are piecewise polynomials that are naturally suited for interpreting time-domain inputs. They offer higher interpretability than frequency domain representations and are particularly effective for modeling smooth and structured data.

c) Distance-awareness and interpretability: Distance-awareness of an uncertainty estimator is the property that the uncertainty of a test point increases with an increase in distance from the training data [11], [14]–[16]. As mentioned in the introduction, distance-aware uncertainty is more intuitive and interpretable than distance-unaware uncertainty. Having said that, interpretability is a subjective term which is hard to define precisely and concretely [28], [29]. Our usage of interpretability here is connected with "explanation by examples" [29]. A distance-aware uncertainty estimator can identify the nearest training data and the corresponding distance as an interpretable explanation of why the uncertainty is high or low. J. Liu et al. [14] introduced the concept of distance-awareness and showed that a dense layer with residual connections satisfies the double-sided Lipschitz continuity. They estimate uncertainty by replacing the last layer of a deep neural network with a Gaussian process and ensuring the Lipschitz continuity of the deep neural network. Their method inherits all the strengths and weaknesses of Gaussian processes, which include poor scalability to large datasets.

A recurring theme in this section has been about interpretability. Interpretability is a widely valued and inconsistently defined concept in machine learning, driven by diverse and sometimes conflicting motivations [28]. Despite this ambiguity, it is common for models to be described as "interpretable" without precise criteria or justification [29]. Different scientific disciplines bring varying intuitions about what makes a function simple or understandable [18], further complicating a unified definition. Mechanistic interpretability seeks to reverse-engineer trained models by identifying the internal components or circuits responsible for specific behaviors [30], while symbolic regression focuses on discovering human-readable mathematical expressions that approximate a model's behavior or underlying data-generating process [31], [32]. In this paper, we define *interpretability* by intuition of human-understandable mapping between model inputs and outputs, particularly through closeness of predictions to training data, which intuitively supports robustness and safety [28], [29], [33].

III. Background

To understand our approach, it is important to understand a few concepts in detail: a) Kolmogorov-Arnold networks (KANs), b) distance-awareness, c) Spectral normalization, and d) DAREK [11]. A recurring theme in this section is the discussion of interpretability, a concept for which there is no universally agreed-upon definition; for further clarification, see Section II-C.

a) KANs: Kolmogorov-Arnold theorem (KAT) states that any continuous multivariate function can be represented as a finite superposition of univariate functions and addition [19]. 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 representing the outer and inner layers' nodes, respectively. KAN [18] 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:

$$f_r(\mathbf{x}) = \sum_{i_L=1}^{N_L} s_{L,r,i_L}\Big(\sum_{i_{L-1}=1}^{N_{L-1}} \mathcal{S}_{L-1,i_L,i_{L-1}}(\mathbf{x})\Big), \tag{2}$$

where $\mathcal{S}$ is defined as follows:

$$\mathcal{S}_{l,j,i}(\mathbf{x}) = s_{l,j,i}\Big(\cdots \sum_{i_2=1}^{N_2} s_{2,i_3,i_2}\Big(\sum_{i_1=1}^{N_1} s_{1,i_2,i_1}(\mathbf{x}_{i_1})\Big)\Big). \tag{3}$$

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.

b) Distance-awareness : The notion of distance-awareness as introduced by J. Liu et al. [14] and extended to nearest neighbor distance by Ataei et al. [11], is defined as follows:

Definition 1 (Distance-awareness [11]): 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 $d(\mathbf{x}, \mathcal{X}_{\text{IND}}) = \min_{\mathbf{x}' \in \mathcal{X}_{\text{IND}}} d(\mathbf{x}, \mathbf{x}')$, which computes the minimum distance between the test point and all training samples.

c) Spectral normalization: Spectral normalization [14] normalizes a single neural network layer to a desired Lipschitz continuity by normalizing its weight matrix W using its spectral norm (largest singular value). Spectral normalization is easiest 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:

$$\begin{aligned}\mathcal{L}_{h_l} &= \frac{\|h_l(\mathbf{x}) - h_l(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \|(W_l\mathbf{x} + \mathbf{b}_l) - (W_l\mathbf{x}' + \mathbf{b}_l)\| \\ &= \frac{\|W_l(\mathbf{x} - \mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \sigma_{\max}(W_l). \end{aligned} \tag{4}$$

where $\sigma_{\max}(W_l)$ is maximum singular value decomposition of 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 [14]

$$\bar{W}_l = \begin{cases} \mathcal{L}_{h_l} \frac{W_l}{\sigma_{\max}(W_l)} & \text{if } \mathcal{L}_{h_l} < \sigma_{\max}(W_l) \\ W_l & \text{otherwise .} \end{cases} \tag{5}$$

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

d) DAREK: This subsection reviews the background needed for spline error analysis. We denote the *Newton's polynomial operator* of order k as defined by [34, 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 $f : [a, b] \to \mathbb{R}$ is the ratio of the maximum change in the $(k-1)$-th derivative of f to the change in the input,

$$\frac{|f^{(k-1)}(x) - f^{(k-1)}(y)|}{d(x, y)} \leq \mathcal{L}_f^k \quad \forall x \neq y. \tag{6}$$

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

$$\begin{aligned}\hat{f}(x) &= \sum_{i=0}^{k} c_{i,j}x^k =: \hat{f}_{[j]}(x) \qquad \forall x \in [\tau_j, \tau_{j+1}), \\ \text{s. t.} \quad & \hat{f}_{[j]}(\tau_{j+1}) = \hat{f}_{[j+1]}(\tau_{j+1}). \end{aligned} \tag{7}$$

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

Interpolation error [11, Theorem 1]: 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:

$$|f(x) - \mathcal{P}_{k,j}[\boldsymbol{\tau}, f(\boldsymbol{\tau})](x)| \leq \underbrace{\frac{\mathcal{L}_f^{k+1}}{(k+1)!} \left| \prod_{i=j}^{j+k} (x - \boldsymbol{\tau}[i]) \right|}_{=:\bar{u}_f(x;\boldsymbol{\tau})} \tag{8}$$

This result assumes perfect interpolation, meaning the approximation exactly matches the function at the knot points. 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.

Spline error [11, Lemma 1]: With the same assumptions as *interpolation error*, 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,

$$|f(x) - \hat{f}(x)| \leq \bar{u}_f(x; \boldsymbol{\tau}) + |\mathcal{P}_{k,j}[e_j^f](x)| =: u_f(x; \boldsymbol{\tau}), \tag{9}$$

The conditions under which these error bounds are tight are detailed in [11].

The error of one output of a spline layer, where N_s splines contribute to the output, is computed as $u_{\text{sp}} = \sum_{i=1}^{N_s} u_{s_i}(\mathbf{x}, \mathcal{T})$. Next, we revisit the two-layer error bound presented in the DAREK paper [11], 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, $\hat{f}(\mathbf{x}) = h_2(\mathbf{h}_1(\mathbf{x}))$, also can be calculated using the following equation [11, Theorem 2]:

$$|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq u_{h_2}\big(\mathbf{h}_1(\mathbf{x}); \mathbf{h}_1(\mathcal{T})\big) + \mathcal{L}_{h_2}^1 \mathbf{1}_{\mathbf{h}_1}^\top \mathbf{u}_{\mathbf{h}_1}(\mathbf{x}; \mathcal{T}) \tag{10}$$

which suggests that the error of the 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 (10) $\mathcal{T}$ is a matrix of size $N_s \times m_k$, where define m_k knots for N_s splines in layer h_1.

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

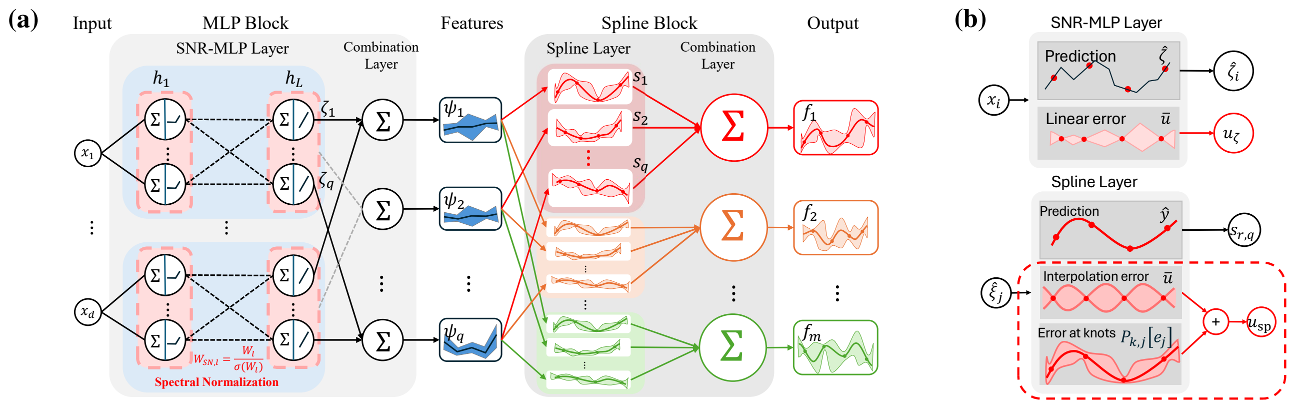

Fig. 1. **(a)** K-DAREK architecture. It has two blocks: MLP and Spline Blocks. The MLP Block has one SNR-MLP for each dimension. The combination layer unifies the output of SNR-MLPs into a feature vector. The Spline Block is a Spline Layer with a combination layer that constructs the model's output. **(b)** The error analysis of K-DAREK, has two main components. The SNR-MLP Layer gets the inputs $\mathbf{x}$ and computes the feature map of SNR-MLPs, ζs, and the linear error (15). The Spline Layer from features $\boldsymbol{\psi}$ calculates the spline values $s_{r,q}$ and error of the spline using interpolation error and error at the knot as explained (10).

IV. Architecture and Method

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

To develop our error analysis, we adopt the KKAN structure with a Spline Block. Building upon KAN, Kúrkovà Kolmogorov-Arnold network (KKAN) [20] 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 inner block is approximated using spectrally normalized Multi-layer perceptron (MLP)s, that expands each input dimension into a higher dimensional space using basis functions, such as Chebyshev basis functions, and project this representation into a hidden space using the MLP, and then expands it through combined output layer. The structure of KKAN can be written as:

$$f_r(\mathbf{x}) = \sum_{i_s=1}^{q} s_{r,i_s}\Bigg(\sum_{p=1}^{d} \underbrace{\mathbf{1}_{hn}^\top \mathbf{T}_n\big(\mathrm{MLP}_{p,i_s,h}(\mathbf{T}_n(x_p))\big)}_{\psi_{p,i_s}(x_p)}\Bigg), \qquad (11)$$

where $\mathbf{T}_n(\mathbf{y}) := (T_0(\mathbf{y}), \dots, T_n(\mathbf{y})) \in \mathbb{R}^{hn}$ 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 only the first two degrees of Chebyshev functions: $T_0(\mathbf{y}) = 1$ and $T_1(\mathbf{y}) = \mathbf{y}$. It allows an easier and more efficient error analysis, while trading off the expressiveness of the model. Secondly, we use spectrally normalized MLP instead of $\mathrm{MLP}_{p,i_s,h}$ in (11). This helps us control the Lipschitz continuity 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 main model architecture comprises two building blocks: MLP Block and Spline Block, as shown in Fig. 1 and equivalently in (11).

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 (11)). 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 (11), 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 over 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 subsections: (a) *Knot selection* to discuss the choice of retaining a subset of the training data for non-parametric error analysis on splines, (b) *Error of the MLP Block*, (c) *Error of the Spline Block*, (d) *Total Error*, and (e) *Lipschitz and Error sharing* to discuss how assumptions on the Lipschitz continuity of the true function can translate to the layer-wise Lipschitz continuity. Similarly, we discuss how errors made in the function approximation can translate to layer-wise errors. A brief outline of the overall error framework is provided in Alg. 1.

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 as denoted by $\boldsymbol{\tau}_c$. Corresponding to these input samples, we

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 $\mathcal{L}_f^{(k+1)}$,
      $\mathcal{L}_f^{(1)}$.
1  Precompute errors $E^f = |Y - \hat{f}(\mathcal{T})|$
2  Divide $\mathcal{L}_f^1$ equally among layers ($\mathcal{L}_h = \sqrt[L+1]{\mathcal{L}_f/d}$).
3  Find feature of test input $\boldsymbol{\psi}^* = \Psi(\mathbf{x}^*)$
4  Find $\mathcal{L}_{\text{sp}} = \mathcal{L}_h/N_{\text{sp}}$ and $\mathcal{L}_{\text{MLP}} = (\mathcal{L}_h)^L$ [Eq.12]
5  Find $u_{\text{MLP}}$ using [Eq. 15]
6  for $n \in \{1, \ldots, N_{sp}\}$ do
       /* $\mathcal{O}(\log_2(m_k))$ binary search. */
7      Find $j$ such that $\boldsymbol{\kappa}_n[j] \le \boldsymbol{\psi}_n < \boldsymbol{\kappa}_n[j+1]$.
8      Fit Newton's Polynomials $\mathcal{P}_{k,j}[\mathbf{e}_n^f]$ on the knots
         $\{(\boldsymbol{\kappa}_n[i], E_i^f)\}_{i=j}^{j+k}$
9      Find $u_{\text{sp}_n}(x_n^*; \boldsymbol{\tau}_n)$ [Eq. 9].
10 Compute error bound over $f$ [Eq. 18]
```

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.

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

Error of MLP Block: Each SNR-MLP consists of L layers, where the l-th layer has an associated Lipschitz constant $\mathcal{L}_l$ as previously discussed. The overall Lipschitz constant of a single SNR-MLP is given by the product of the layer-wise Lipschitz constants:

$$\mathcal{L}_{\text{MLP}} \coloneqq \mathcal{L}_1\, \mathcal{L}_2 \cdots \mathcal{L}_L, \tag{12}$$

although (12) provides a valid Lipschitz bound for the SNR-MLP, it may not be the tightest possible bound. Given a test point $\mathbf{x}^*$, we can estimate the error of the i-th SNR-MLP using the following bound:

$$\begin{aligned} u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) &= \mathcal{L}_{\text{MLP}_i}\, \|x_i^* - \tau_i^*\|, \text{where,} \\ \tau_i^* &= \operatorname*{argmin}_{\tau^\dagger \in \boldsymbol{\tau}_i} \|\tau^\dagger - x_i^*\|. \end{aligned} \tag{13}$$

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^*. The worst error bound for the MLP Block is then obtained by summing the individual bounds across all input dimensions:

$$\begin{aligned} u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) &= \sum_{i=1}^{d} u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) \\ &= \sum_{i=1}^{d} \mathcal{L}_{\text{MLP}_i}\, \|x_i^* - \tau_i^*\|. \end{aligned} \tag{14}$$

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

$$u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) = \mathcal{L}_{\text{MLP}} \sum_{i=1}^{d} \|x_i^* - \tau_i^*\|. \tag{15}$$

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: Let 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:

$$\begin{aligned} u_{\text{sp}_{r,i}}(\xi_i^*, \boldsymbol{\kappa}_i) \coloneqq\; & \bar{u}_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i) \\ & + \left\|\mathcal{P}[(\hat{f}(\boldsymbol{\tau}_r) - \mathbf{y}_r)_{[j]}^{\text{sp}}](\xi_i^*)\right\| \end{aligned} \tag{16}$$

And the worst error bound of the r-th group of splines can be written as:

$$u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) \coloneqq \sum_{i=1}^{q} u_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i) \tag{17}$$

Total Error: Having established the error for both the MLP and Spline Blocks in Equations (15) and (17), 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:

$$u_{f_r}(\mathbf{x}_*) = u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) + \mathcal{L}_{\text{sp}}^1 u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) \tag{18}$$

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 based on Lipschitz continuity and ensure that the neural network approximator preserves the stability and bounded sensitivity of the true system, we require the approximated model to exhibit a Lipschitz continuity behavior consistent with the known Lipschitz constant $\mathcal{L}_f$ of the true dynamics. To achieve this, we divide the overall Lipschitz constant $\mathcal{L}_f$ across the L MLP layers and the spline layer. Specifically, we divide $\mathcal{L}_f$ equally among layers by setting the Lipschitz constant of each layer to $\mathcal{L}_h = \sqrt[L+1]{\mathcal{L}_f/d}$, where d is the input dimension, inspired by [14]. The division by d accounts for the fact that we use d independent SNR-MLPs (one per input dimension). Additionally, we assumed the error at knots only arises from Spline Block. Under this assumption, we distribute the total error equally among splines within each group.

V. Experiments and Results

We evaluated K-DAREK and compared it with DAREK in two 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 cubic splines, each with 9 knots. In contrast, the K-DAREK model uses an inner block with a single hidden layer with 5 units and an outer block that linearly combines spline outputs to produce the final output. The output splines are also cubic with 9 knots. Both models

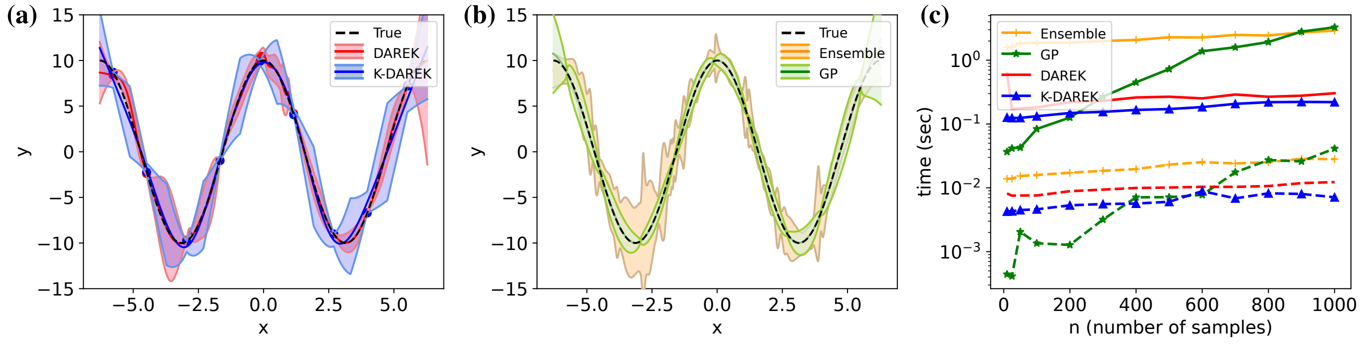

Fig. 2. Cosine function approximation experiment. **(a)** Error estimation comparison of DAREK and K-DAREK. **(b)** 3σ uncertainty bounds comparison of Ensemble and GP. **(c)** Time Comparison of K-DAREK (green), DAREK (orange), GP (red), and Ensemble (blue) models. Training time (solid line) and inference time (dashed).

are trained for 500 epochs with a learning rate of 0.1. Fig. 2 **(a)** compares the error estimation of DAREK and K-DAREK. The results suggest that K-DAREK has a better 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. The plot **(b)** compares the uncertainty bounds produced by an Ensemble of KANs and Gaussian Processes (GPs). To ensure a fair comparison, the same input knots were used for DAREK, K-DAREK, Ensemble, and GP. In DAREK, the lack of architectural constraints can lead to erratic behavior. 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 (II) shows that K-DAREK achieves the lowest 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 $loss = \sum_{i=1}^{n}(\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i))/n$, where n is the number of test points. The size indicates the total number of learnable parameters in the model. Fig. 2 **(c)** illustrates the time complexity of DAREK and K-DAREK compared to GPs and an Ensemble of 10 KAN models. The plot shows that K-DAREK is more efficient than DAREK across all dataset sizes and significantly outperforms both GP and Ensemble for datasets larger than 500 samples.

TABLE II
ERROR ESTIMATOR MODEL COMPARISON BETWEEN DAREK AND K-DAREK IN APPROXIMATION OF COSINE FUNCTION. K-DAREK HAS BETTER FITTING AND LESS VIOLATIONS.

Model	MSE loss	Violations(%)	width	size
Ensemble	0.378	0.0	10×[1,5,1]	700
GPs	**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**

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 [35], [36]. In this experiment, we tested

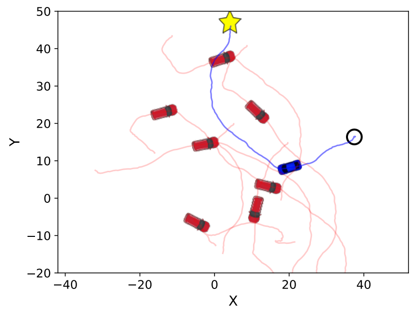

Fig. 3. Multi-agent safe control setup. This is one of the successful trajectories using K-DAREK bounds. The controlled agent and its path are shown in blue, and surrounding red vehicles represent other agents. Using K-DAREK bounds, the controller with K-DAREK bounds was able to navigate from the start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

both approaches (DAREK and K-DAREK) in a multi-agent safe control setup. The experiment is adopted from [37]. Fig. 3 shows the problem setup, where the controlled agent (blue car) is supposed to navigate safely from the circle location and reach the goal (the star) without running into other agents (red cars). The other agents have their own base controller that is unknown to the controlled agent, and we only control the blue car by observing the state of other cars. The system dynamics of the car is discrete control-affine with the form of:

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) + g(\mathbf{x}_t)\mathbf{u} + d(\mathbf{x}_t), \tag{19}$$

where state $\mathbf{x} \in \mathbb{R}^4$ contains position and velocity in x and y directions. Here, f is the nominal dynamic or drift term, g is the control effectiveness matrix, $\mathbf{u}$ is the control input, which are accelerations in x and y directions, and d is the disturbance. A model predictive control (MPC) controller without considering the obstacle was adopted to find the unsafe path of the next state to reach, and a control barrier functions (CBF) control based on [37, eq. 17] construct to find the closest safe control input considering the obstacles.

The paper constructs a quadratic program for CBF based optimization problem by bounding the disturbance d in a polytope. We used the predicted worst-case error bound of 2-layer DAREK (D2), 2-layer K-DAREK (K-D2), and 3-layer K-DAREK (K-D3) models as bounds on the polytope. D2 has two spline layers with a hidden unit of 5, K-D2 has the SNR-MLP of size [2,5] and a spline of [5,4], and K-D3 has an SNR-MLP of size [2,5,5] and a spline of size [5,4]. We evaluate performance over 100 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 on data collected for 5000 steps without noise, for 300 epochs. We used different uniform noises for position and velocity as reported in Table III by d_p and d_v. The total system Lipschitz has been considered as $\mathcal{L}_f = \bar{d_p} + \bar{d_v}\text{dt}$, and the simulation step time of 0.1 is chosen for dt, where $\bar{d_p}$ and $\bar{d_v}$ are position and velocity error bounds. Results show that while K-DAREK has a slightly lower success rate, however, it results in fewer collisions, which is in agreement with 1D experiment that K-DAREK is more conservative and has fewer violations. K3 has larger error bounds, which suggests sacrificing success rate for fewer collisions.

VI. Conclusion

In this work, we presented K-DAREK a principled approach for quantifying the distance-aware worst-case error in hybrid models composed of deep dense neural networks and spline-based networks. We derived an explicit error bound for the MLP block using its Lipschitz constant, which relates test point uncertainty to proximity with training data. For the spline block, we redefined the formulation from DAREK and matched it with this purpose based on the selection of knots derived from the training set and the accuracy of predicted values at those knots. Combining these components, we arrived at an overall model error bound that incorporates both local interpolation uncertainty and propagated transformation error through the model. This framework enables a more 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] M. Egerstedt and C. Martin, *Control theoretic splines: optimal control, statistics, and path planning*. Princeton University Press, 2009.

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

[3] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "Splinecnn: Fast geometric deep learning with continuous b-spline kernels," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 869–877.

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

[5] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[6] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330.

[7] 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.

[8] 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.

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

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

[11] M. Ataei, M. J. Khojasteh, and V. Dhiman, "DAREK-Distance aware error for Kolmogorov networks," in *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2025, pp. 1–5.

[12] J. A. Lopez and V. Kreinovich, "Towards decision making under interval uncertainty," in *North American Fuzzy Information Processing Society Annual Conference*. Springer, 2023, pp. 335–337.

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

[14] 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.

[15] 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.

[16] 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.

[17] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, January 2006. [Online]. Available: https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/

[18] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljacic, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov–arnold networks," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: https://openreview.net/forum?id=Ozo7qJ5vZi

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

[20] 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.

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

[22] M. Abdar, F. Pourpanah, S. Hussain, D. Rezazadegan, L. Liu, M. Ghavamzadeh, P. Fieguth, X. Cao, A. Khosravi, U. R. Acharya *et al.*, "A review of uncertainty quantification in deep learning: Techniques, applications and challenges," *Information fusion*, vol. 76, pp. 243–297, 2021.

[23] J. Gawlikowski, C. R. N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher *et al.*, "A survey of uncertainty in deep neural networks," *Artificial Intelligence Review*, pp. 1–77, 2023.

[24] F. Girosi and T. Poggio, "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, no. 4, pp. 465–469, 1989.

[25] V. Kůrkovà, "Kolmogorov's theorem is relevant," *Neural computation*, vol. 3, no. 4, pp. 617–622, 1991.

[26] V. Kůrkovà, "Kolmogorov's theorem and multilayer neural networks," *Neural networks*, vol. 5, no. 3, pp. 501–506, 1992.

[27] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial intelligence and statistics*. PMLR, 2016, pp. 370–378.

TABLE III
RESULTS FOR MULTIAGENT SAFE CONTROL EXPERIMENT, WHERE d_p AND d_v ARE UNIFORM DISTURBANCES ON THE POSITION AND THE VELOCITY. D2 IS DAREK WITH TWO LAYERS, AND K-D2 AND K-D3 ARE TWO-LAYER K-DAREK AND THREE-LAYER K-DAREK, AS EXPLAINED IN THE TEXT. THE EXPERIMENT GIVES A LOWER COLLISION RATE FOR K-DAREK MODELS COMPARED WITH THE DAREK MODEL.

		$\bar{d_v} = 0$			$\bar{d_v} = 1$			$\bar{d_v} = 2$			$\bar{d_v} = 3$		
$\bar{d_p}$	Model	D2	K-D2	K-D3	D2	K-D2	K-D3	D2	K-D2	K-D3	D2	K-D2	K-D3
0	Success	97	96	93	93	95	96	93	92	93	89	91	83
	Collision	1	1	1	3	2	0	4	2	4	9	6	5
	Stuck	2	3	6	4	3	4	3	6	3	2	3	12
1	Success	91	90	82	84	74	68	82	64	51	69	49	47
	Collision	9	10	2	15	3	2	13	3	1	14	3	6
	Stuck	0	0	16	1	23	30	5	33	48	17	48	47

[28] F. Doshi-Velez and B. Kim, “Towards a rigorous science of interpretable machine learning,” *arXiv preprint arXiv:1702.08608*, 2017.

[29] Z. C. Lipton, “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” *Queue*, vol. 16, no. 3, pp. 31–57, 2018.

[30] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, “Progress measures for grokking via mechanistic interpretability,” *arXiv preprint arXiv:2301.05217*, 2023.

[31] N. Makke and S. Chawla, “Interpretable scientific discovery with symbolic regression: a review,” *Artificial Intelligence Review*, vol. 57, no. 1, p. 2, 2024.

[32] T. Tohme, M. J. Khojasteh, M. Sadr, F. Meyer, and K. Youcef-Toumi, “ISR: Invertible symbolic regression,” *arXiv preprint arXiv:2405.06848*, 2024.

[33] R. Marcinkevičs and J. E. Vogt, “Interpretability and explainability: a machine learning zoo mini-tour. 2020, p,” *arXiv preprint arXiv:2012.01805*, 2012.

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

[35] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, “Control barriers in bayesian learning of system dynamics,” *IEEE transactions on automatic control*, vol. 68, no. 1, pp. 214–229, 2021.

[36] 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.

[37] 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 *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 777–783.