IEEE CSS

Worst-Case Distance-Aware Error Bounds for Neural Networks

Masoud Ataei[1], Vikas Dhiman[1] (Member, IEEE), Mohammad Javad Khojasteh[2] (Member, IEEE)

[1]Electrical and Computer Engineering, University of Maine, Orono, ME 04469 USA
[2]Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY 14623 USA

CORRESPONDING AUTHOR: M. Ataei (e-mail: masoud.ataei@maine.edu)

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

ABSTRACT Safety-critical applications of machine learning require uncertainty estimates that support reliable worst-case analysis. Neural networks (NNs) provide expressive function approximation, while Gaussian processes (GPs) offer principled probabilistic uncertainty, but both face limitations in this setting: most modern neural architectures lack tractable worst-case error bounds, and Gaussian processes become computationally expensive at scale. For uncertainty to be interpretable, a central requirement is distance-awareness: that is, uncertainty increases with the distance between a test input and the nearest relevant training data. We present a general framework for worst-case distance-aware error bounds for NNs that combine dense layers with spline-based components. Our approach establishes error bounds that are both distance-aware, reflecting proximity of a test point to its nearest training data, and worst-case, providing deterministic guarantees under known Lipschitz constraints rather than probabilistic assumptions. Our algorithm, K-DAREK (Distance-Aware Error for Kůrkovà-Kolmogorov Networks), provides efficient and interpretable uncertainty quantification for NNs. K-DAREK is about four times faster and ten times more computationally efficient than an ensemble of KANs, 8.6 times more scalable than GP, eliminates up to 8.2% error-bound violation rate observed in DAREK, and reduces the average collision rate from 1.8% to 1.1% in the multi-agent safe control experiment. On high-dimensional real-world regression tasks (e.g., Real Estate Valuation), K-DAREK preserves distance-aware error bounds and achieves zero coverage violations, addressing the overgeneralization and inducing-point-coverage limitations exhibited by SNGP and DUE, respectively.

INDEX TERMS Uncertain systems, safe learning for control, error bounds, neural networks, worst-case analysis, spline, Kurkova Kolmogorov-Arnold networks (KKAN).

I. INTRODUCTION

Deploying neural networks (NNs) in safety-critical applications requires provable error bounds, which most modern architectures lack [1]. NN predictions do not generally provide guarantees on the worst-case deviation from the true underlying function, making it difficult to reason about safe decisions or operations. A principled worst-case analysis for NNs at any test point would quantify the maximum possible deviation between the network's prediction and the true function. We present a general framework for a worst-case distance-aware error bound for NNs that combine dense layers with spline-based components. 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], [3], [4], [5], [6], [7], [8]. Spline-based NNs [9], [10], [11], [12] can leverage these properties to produce smoother function approximations than conventional multi-layer perceptrons (MLPs). Furthermore, spline-based NNs offer improved interpretability and thus greater reliability [13]. However, producing robust and trustworthy uncertainty estimates remains difficult. Two predominant strategies for uncertainty estimation are probabilistic methods and worst-case formulations (e.g., adversarial or robust).

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) [14] and Bayesian neural network approaches such as Monte Carlo

TABLE 1. Notation and definitions of important quantities.

Notation	Definition	Notation	Definition
f, $\hat{f}$	Target function and trained approximator	$\hat{h}_{\text{MLP}}$, $\hat{h}_{\text{sp}}$	MLP block and spline block of K-DAREK
d, $d_u(.)$	Input dimension $\mathbf{x} \in \mathbb{R}^d$, distance function candidate for uncertainty function u	q	Intermediate feature dimension
$\mathcal{D}$	Training dataset, $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$	L	Number of layers in each SNR-MLP
$\mathcal{X}$, $\mathcal{X}_\mathcal{D}$	Input space ($\mathcal{X} \subset \mathbb{R}^d$) and input portion of $\mathcal{D}$	$\boldsymbol{\tau}$, k	Set of knots and order of spline
$\mathbf{x}$, $\mathbf{x}^*$	Input vector and test input vector	$\boldsymbol{\tau}^*$	Nearest training input (knot) to test point $\mathbf{x}^*$
$\mathcal{L}_f$, $\mathcal{L}_{\text{sp}}$	Lipschitz constant of true function f and spline block	$\mathcal{L}_{\text{MLP}}$, $\mathcal{L}_{\text{MLP}_i}$	Lipschitz constants of MLP block and the i-th SNR-MLP
u_{MLP}, u_{sp}, u_f	Error bounds on MLP block, spline block, and target function f	$\mathcal{T}$, K	Input knot matrix ($\mathcal{T} \in \mathbb{R}^{m_k \times d}$) and feature-space knot matrix ($K \in \mathbb{R}^{m_k \times q}$).
$\boldsymbol{\xi}$, $\boldsymbol{\xi}_i$	Aggregated latent feature vector before spline mapping and its i-th component	m_k	Number of knots per spline

dropout [15], variational inference [16], and ensembles [17]. 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 [18], [19], [20], [21]. Additionally, GPs, which are popular in uncertainty estimation, are computationally expensive and also rely on kernel hyperparameters [14]. 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 [22], [23], [24], [25], [26]. 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 [23], [27]. In control theory, worst-case analysis can be particularly appealing because many problems are formulated in terms of bounded errors [28], [29].

An interpretable uncertainty estimator is expected to demonstrate distance-awareness, that is, to have higher predictive confidence in regions near the training data, while exhibiting increased uncertainty as the input deviates from the observed training data [17], [30], [23], [31], [32]. Uncertainty estimation in nonparametric models—such as GPs, splines, k-nearest neighbors, kernel density estimation [33]—can be easily made distance-aware because they depend on stored training data, whereas typical parametric models—such as ensembles [17] and support vector machines—do not estimate distance-aware uncertainty.

More recently, Kolmogorov-Arnold network (KAN) [13], [34], [35], [36], [37], [38] introduced a novel approach that combines nonparametric models (splines) with parametric models [23]. KAN, inspired by the Kolmogorov-Arnold theorem (KAT) [39], [40], was proposed to model complex functions with a deep neural network using spline bases as activation functions. Building on this success, Kùrkovà Kolmogorov-Arnold networks (KKANs) [41] 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 [23], 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 Fig. 1). In the MLP block, each input dimension is transformed through a spectrally normalized MLP layer [42], [30] 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 constant. We then apply the spline error analysis from our previous work [23] 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 [23], Distance-aware error 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 spline neural networks (SNNs) more broadly, can

TABLE 2. Expansions of important acronyms.

Acronym	Expansion
NN	Neural network
SNN	Spline neural network
KAT	Kolmogorov-Arnold theorem
KAN	Kolmogorov-Arnold network
KKAN	Kůrková Kolmogorov-Arnold network
ReLU	Rectified linear unit
MLP	Multi-layer perceptron
MLP-Spline	A network that uses MLPs and Splines in its architecture
SNR-MLP	Spectral-normalized ReLU activated MLP
RBF	Radial basis function
GP	Gaussian process
DKL	Deep kernel learning
DUE	Deterministic uncertainty estimation
SNGP	Spectral-normalized neural Gaussian process
DAREK	Distance-aware error for Kolmogorov networks
K-DAREK	distance-aware error for Kůrková Kolmogorov networks
CBF	Control barrier function
MPC	Model predictive control
MSE	Mean squared error

yield highly nonlinear behavior, often resulting in non-smooth approximations (which may be mitigated through regularization [13]). The hybrid architecture of distance-aware error for Kůrková Kolmogorov networks (K-DAREK), which combines MLP and spline components, produces smoother function approximations by using 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 efficiently estimate distance-aware worst-case error bounds for NNs. This is made possible by integrating splines into deep NNs. We propose a general framework for worst-case error bounds for MLP-Spline models, with K-DAREK as an instance of these models. The contributions of this work are as follows: 1) We propose a novel general framework for deriving worst-case error bounds that leverages the expressive power of MLPs together with spline-based structures to provide interpretable, distance-aware error bound estimates, under the assumption of known Lipschitz constraints. 2) We introduce K-DAREK as an instance of our framework, and compare K-DAREK against spectral-normalized neural Gaussian process (SNGP) [30], deterministic uncertainty estimation (DUE) [43], GPs, ensembles, and our previous work [23]. We evaluate K-DAREK on synthetic datasets, real-world regression datasets (e.g., Real Estate Valuation), as well as in regions with missing data across multiple datasets. We further test on a high-dimensional face bounding-box prediction and a safe multi-agent control task. We also conduct ablation studies to isolate the contributions of the MLP block, spectral normalization, and the knot-selection strategy, and a sensitivity analysis quantifying how the violation rate degrades when the Lipschitz constant is underestimated. 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, eliminates up to an 8.2% error-bound violation rate observed in DAREK [23], and reduces the average collision rate from 1.8% to 1.1% in the multi-agent safe control experiment. Furthermore, unlike SNGP, K-DAREK does not suffer from "feature collapse" (overgeneralization) [43] on large datasets, effectively captures uncertainty in regions with missing data, and remains distance-aware in high dimensions.

A preliminary version of this work appeared in [44]. The current paper substantially extends that work in the following ways: 1) we formalize the problem of worst-case distance-aware error bound of general NNs rigorously and exemplify it in the specific case of KKAN model; 2) we add a comprehensive related work section, a structured overview of the framework, and an computational-complexity analysis; 3) we significantly expand the experimental evaluation to cover feature collapse, missing-data regions across multiple function constructions, high-dimensional face bounding-box prediction (up to 200 dimensions), a fixed budget comparison, real-world regression datasets, an ablation study isolating the contribution of each architectural component, a sensitivity analysis of the assumed Lipschitz constant, and an extensive safe control experiment; 4) we release a software package for reproducing these experiments[1].

Important notations and definitions used in the paper are summarized in Table 1. Acronym expansions are provided in Table 2.

II. RELATED WORKS

Uncertainty estimation in NNs is a well-investigated research problem. However, most of these models are probabilistic and not distance-aware. For a comprehensive survey, we refer the interested reader to [45], [46]. In this section, we discuss closely related work and provide a historical perspective.

Spline-Based Neural Networks: KAT states that any multivariate continuous function can be represented as a finite decomposition of univariate continuous functions and addition [39]. The theorem does not hold if the continuity assumption is replaced with the continuously differentiable assumption [47], and there has been a long debate about whether the KAT is relevant to the design of NNs. This controversy is exemplified by two influential papers: Girosi and Poggio argue that KAT lacks practical implications for modern network architectures, as it does not account for the smoothness and generalization properties that are critical in practical learning systems [48]. Whereas Kůrková contends that the theorem provides foundational insight into the representational capabilities of NNs [49], [50], [39]. Z. Liu et al. [13] address this issue by focusing on SNN with many layers. KANs [13] replace traditional fixed activation functions with learnable, spline-parametrized functions on edges, enhancing interpretability and adaptability. Kůrková introduced a different remedy for this over three decades ago [49]. She suggested that the component function in the

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

decomposition could be effectively approximated using an MLP with small error, rather than KAT, where the aim was to achieve zero error. While KAN and KKAN improve interpretability, neither inherently provides distance-aware error bound estimates and worst-case error guarantees; our previous work addressed this for KAN [23], and this paper extends it to KKAN.

Deep kernel learning: There are similarities between Deep Kernel Learning (DKL) and our design, which is based on KKAN. Wilson et al. [51] introduced DKL by replacing the last layer of an MLP with a kernel layer, such as spectral mixture (SM) base kernels, and jointly learning the kernel hyperparameters along with the network weights. This modification combined the expressiveness of deep networks with the probabilistic properties of GPs, effectively making the network behave like a GP with the same kernel. Similarly, KKAN can be interpreted as an MLP in which the final layer is replaced by 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 operating in the frequency domain, often exhibiting periodic behavior and lacking direct spatial or semantic correspondence. In contrast, spline-based representations are piecewise polynomials that are well-suited to interpreting time-domain inputs. They offer higher interpretability than frequency domain representations and are particularly effective for modeling smooth, structured data. Moreover, unlike our design, DKL does not guarantee distance-awareness and may map inputs that are far apart to the same feature representation [43].

Probabilistic distance-aware methods: Existing distance-aware uncertainty estimators are predominantly probabilistic, with SNGP [30] being a seminal example. In [30], distance-awareness is used to quantify how far an input lies from training examples and to produce a graded measure of unfamiliarity that aids out-of-distribution (OOD) detection [52]. To study the distance-aware error bounds, we draw inspiration from the SNGP, which provides formal error guarantees for NNs by leveraging Lipschitz continuity. In the SNGP framework, weight normalization is applied to enforce a global Lipschitz constraint on the MLP. Moreover, the final layer of the MLP is replaced with a GP layer, akin to DKL. However, unlike standard DKL, the GP layer in SNGP is implemented via random Fourier features (RFF), and the underlying approximation for tractability may lead the uncertainty to converge to zero in the data limit [43]. In this work, we extend these ideas in K-DAREK by combining MLPs with a KAN-based architecture, yielding a scalable, computationally efficient method that provides worst-case distance-aware error bounds for this network. Unlike SNGP, which employs a single MLP with a GP approximation only at the final layer, our worst-case (adversarial) approach applies a separate MLP to each input, yielding a structurally more input-adaptive architecture.

DUE [43], similar to SNGP, uses spectral normalization on the feature extractor and applies an approximated GP on the final layer. Unlike SNGP, it employs an inducing-point GP instead of RFF, which yields better results; however, it may yield poor uncertainty estimates when the number of inducing points is limited or when the points do not cover parts of the feature space, especially in high dimensions.

Other probabilistic distance-aware methods include the following. Deterministic uncertainty quantification (DUQ) [32] is restricted to classification and models each class via a single centroid in feature space, which may be limiting. In this context, deep deterministic uncertainty (DDU) [31] fits one Gaussian mixture per class, so it assumes the features of a class follow a Gaussian distribution.

Probabilistic distance-awareness quantifies the expected distance of a test point from the training distribution, typically using measures such as likelihood or kernel density estimates. In contrast, our worst-case (robust) distance-awareness analysis relies on deterministic bounds, defined via Lipschitz constants, to guarantee how much the output can change under any admissible perturbation. We present a detailed empirical comparison of these probabilistic distance-aware methods with K-DAREK, and highlight their conceptual similarities and differences. Also, our method is a deterministic alternative to probabilistic approaches rather than a variant of them. In Section VI, we demonstrate how probabilistic models can produce misleading uncertainty estimates and overgeneralize in the absence of data.

Interpretability: A recurring theme in this paper is interpretability. Distance-awareness provides a form of interpretability. An estimator whose confidence reflects proximity to training data offers a human-understandable explanation for its own output, since the nearest training point and its distance explain why the output is high or low. Interpretability itself is a widely valued and inconsistently defined concept in machine learning, driven by diverse and sometimes conflicting motivations [53]. Despite this ambiguity, it is common for models to be described as “interpretable” without precise criteria or justification [54]. Different scientific disciplines offer varying intuitions about what makes a function simple or understandable [13], [55], [56], further complicating the pursuit of a unified definition. Mechanistic interpretability seeks to reverse-engineer trained models by identifying the internal components or circuits responsible for specific behaviors [57], while symbolic regression focuses on discovering human-readable mathematical expressions that approximate a model’s behavior or underlying data-generating process [58], [59], [60]. In this paper, we define ***interpretability*** as the human-understandable mapping between model inputs and outputs, particularly through closeness of predictions to training data, which intuitively supports robustness and safety [54], [53]. Interpretability is a subjective term which

is hard to define precisely and concretely [53], [54]; here, it is connected with "explanation by examples" [54].

System verification: Neural network verification methods such as CROWN, α-CROWN, β-CROWN, and auto-LiRPA [61], [62], [63], [64] certify output bounds for a fixed, already-trained network $\hat{f}$ under a specified input perturbation, by propagating bounds through its known weights. These methods have also been applied to control-theoretic guarantees directly, for instance, certifying Lyapunov stability of neural controllers [65]. Our work focuses on a different problem than system verification literature. System verification focuses on ensuring a constraint is satisfied for all inputs, $g(\hat{f}(x), x) \geq 0$ for all $x \in \mathcal{X}$, given a learned function $\hat{f}(x)$ [65], where g is a task-specific constraint function (for instance, a safety or stability condition). Typical approaches find upper and lower bounds of the learned function $\overline{f}(x) \geq \hat{f}(x) \geq \underline{f}(x)$, and use the bound to guarantee $\underline{g}(\hat{f}(x), x) \geq 0$. On the other hand, we aim to find the error between a learned function $|\hat{f}(x) - f(x)|$ and an unknown function that is only observable through data samples and structural assumptions, in our case, Lipschitz continuity. To make this comparison concrete, in Sec. VI (Adapted-CROWN comparison experiment), we adapt a CROWN-style linear relaxation to K-DAREK's architecture and compare the resulting output bound against K-DAREK's own error bound on the same task, anchored at the same training knots (Fig. 3 **(d)**). This distinction also matters directly for control applications: in our multi-agent CBF experiment (Sec. VI), we use K-DAREK's error bound to quantify uncertainty in a learned dynamics model, whereas system verification methods such as [65] instead certify an upper bound on a controller's own output; the two serve different roles even when both are used inside a control pipeline.

III. BACKGROUND

We begin by defining distance-awareness and then reviewing the key background concepts needed to understand our approach: a) measuring distance-awareness [66], b) KANs [13], c) spectral normalization [42], and d) DAREK [23].

Definition 1 (Distance-awareness [23])**.** *An approximate function $\hat{f}(\mathbf{x})$, defined over $\mathbf{x} \in \mathcal{X}$, is* ***distance-aware*** *if its associated error or uncertainty $u_{\hat{f}}(\mathbf{x})$ increases monotonically with the distance of a test point $\mathbf{x}$ from the training input $\mathcal{X}_{\mathcal{D}}$*[2]*.*

The distance is quantified by a function $d_u(\mathbf{x}, \boldsymbol{\tau}^*)$, which computes the minimum distance between the test point and all training samples for the uncertainty function u. We denote the nearest available training point by $\boldsymbol{\tau}^* \in \mathcal{X}_{\mathcal{D}}$. The uncertainty function u_f on the direction of the distance function d_u at test point $\mathbf{x}^*$ is distance-aware if

$$[\nabla_{\mathbf{x}} u_f(\mathbf{x}^*)]^\top \nabla_{\mathbf{x}} d_u(\mathbf{x}^*, \boldsymbol{\tau}^*) \geq 0 \quad \forall \mathbf{x}^* \in \mathcal{X}. \tag{1}$$

Measuring distance-awareness: Checking the condition in (1) over the entire continuous input space $\mathcal{X}$ is, in general, computationally intractable, as it requires establishing the global monotonicity of the uncertainty function. The function u_f is generally nonlinear. There is no closed-form way to check condition (1) at every point in $\mathcal{X}$. In practice, we evaluate this property using a sampling-based approach, sampled distance-awareness (SDA) (2). This provides a practical and consistent metric for evaluating this property across different uncertainty estimators. By considering $d_u = \|x - \tau^*\|^2/2$ as the distance function, taking N samples from the input space, and computing the frequency with which condition (1) is satisfied, we arrive at a metric with the following formulation. Since $\nabla_x d_u(x, \tau^*) = x - \tau^*$, condition (1) reduces to $[\nabla_x u_{\hat{f}}(x^*)]^\top (x^* - \tau^*) \geq 0$, which is exactly the sign condition evaluated inside the indicator below. The *sampled distance-awareness* (SDA) is defined as:

$$\text{SDA} = \frac{1}{N} \sum_{\mathbf{x}_t \sim \mathcal{X}_{\text{test}}} \mathbb{1}\left[\nabla_{\mathbf{x}} u_{\hat{f}}(\mathbf{x}_t)^\top (\mathbf{x}_t - \boldsymbol{\tau}^*) \geq 0\right], \tag{2}$$

where $\mathbb{1}[.]$ is the indicator function and $\mathbf{x}_t \sim \mathcal{X}_{\text{test}}$ are test samples drawn uniformly from test data. A higher SDA value indicates better distance-awareness, meaning that uncertainty is more likely to increase as the test point moves farther from the nearest training point. Our goal is therefore empirical distance-awareness, measured via SDA and reported comparatively against baselines, rather than a theoretical guarantee that Definition 1 holds globally.

KANs [13]: KAT states that any continuous multivariate function can be represented as a finite superposition of univariate functions and addition [39]. Formally, a continuous multi-variate function $f : \mathcal{X} \to \mathbb{R}$, $(\mathcal{X} \subset \mathbb{R}^d)$ can be represented as:

$$f(\mathbf{x}) = \sum_{q=1}^{2d+1} \Phi_q\Big(\sum_{p=1}^{d} \psi_{p,q}(x_p)\Big), \tag{3}$$

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 [13] 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:

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

$$\mathcal{S}_{l,j,i}(\mathbf{x}) = s_{l,j,i}\Big(\cdots \sum_{i_2=1}^{N_2} s_{2,i_3,i_2}\Big(\sum_{i_1=1}^{N_1} s_{1,i_2,i_1}(\mathbf{x}_{i_1})\Big)\Big). \tag{4}$$

[2]Throughout this paper, $u_{\hat{f}}(x)$ denotes a deterministic worst-case error bound on $|f(x) - \hat{f}(x)|$ under the Lipschitz assumptions, as distinct from the probabilistic uncertainty (e.g., posterior variance) reported by baselines such as GP, DUE, and SNGP.

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.

Spectral normalization: Spectral normalization [30] enforces a desired Lipschitz continuity for a single fully-connected neural network layer by normalizing its weight matrix W by 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:

$$\begin{aligned}\mathcal{L}_{h_l} &= \frac{\|h_l(\mathbf{x}) - h_l(\mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \|(W_l\mathbf{x} + \mathbf{b}_l) - (W_l\mathbf{x}' + \mathbf{b}_l)\| \\ &= \frac{\|W_l(\mathbf{x} - \mathbf{x}')\|}{\|\mathbf{x} - \mathbf{x}'\|} \leq \sigma_{\max}(W_l). \end{aligned} \tag{5}$$

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 [30]

$$\bar{W}_l^{(t+1)} = \begin{cases} \frac{\mathcal{L}_{h_l} W_l^{(t)}}{\sigma_{\max}(W_l^{(t)})} & \text{if } \mathcal{L}_{h_l} < \sigma_{\max}(W_l^{(t)}) \\ W_l^{(t)} & \text{otherwise .} \end{cases} \tag{6}$$

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 activated MLP (SNR-MLP).

DAREK [23]: Here, we review the background needed for the spline error analysis. We denote *Newton's polynomial operator* of order k as defined by [67, p.7] on a sorted set of knots $\boldsymbol{\tau}$ 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 the set or vector $\boldsymbol{\tau}$, and the true output values are determined by $f(\boldsymbol{\tau})$. We define the *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,

$$\frac{|f^{(k-1)}(x) - f^{(k-1)}(y)|}{d(x, y)} \leq \mathcal{L}_f^k \quad \forall x \neq y. \tag{7}$$

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

$$\begin{aligned} \hat{f}(x) &= \sum_{i=0}^{k} c_{i,j} x^k =: \hat{f}_{[j]}(x) \qquad \forall x \in [\tau_j, \tau_{j+1}), \\ \text{s. t.} \quad & \hat{f}_{[j]}(\tau_{j+1}) = \hat{f}_{[j+1]}(\tau_{j+1}). \end{aligned} \tag{8}$$

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

Theorem 1 (Interpolation error [23])**.** *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:*

$$|f(x) - \mathcal{P}_{k,j}[\boldsymbol{\tau}, f(\boldsymbol{\tau})](x)| \leq \underbrace{\frac{\mathcal{L}_f^{k+1}}{(k+1)!} \left| \prod_{i=j}^{j+k} (x - \boldsymbol{\tau}[i]) \right|}_{=: \bar{u}_f(x; \boldsymbol{\tau})}. \tag{9}$$

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 knots.

Remark 1. *The upper bound in (9) is tight and reduces to an equality when $(k+1)$-th derivative of function f is constant and equal to $\mathcal{L}_f^{k+1}$, $f^{(k+1)}(x) = \mathcal{L}_f^{k+1}$, $\forall x \in [\boldsymbol{\tau}[j], \boldsymbol{\tau}[j+1]]$ [66]. Hence, the bound is tighter than the trivial bound (15), for any $k+1$ Lipschitz function with constant $\mathcal{L}^{k+1}$.*

Theorem 2 (Spline error [23])**.** *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,*

$$|f(x) - \hat{f}(x)| \leq \bar{u}_f(x; \boldsymbol{\tau}) + |\mathcal{P}_{k,j}[e_j^f](x)| =: u_f(x; \boldsymbol{\tau}). \tag{10}$$

The conditions under which these error bounds are tight are detailed in [23].

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 [23], as our K-DAREK architecture 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 [23, Theorem 2]:

$$|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq u_{h_2}\big(\mathbf{h}_1(\mathbf{x}); \mathbf{h}_1(\mathcal{T})\big) + \mathcal{L}_{h_2}^1 \mathbf{1}_{\mathbf{h}_1}^\top \mathbf{u}_{\mathbf{h}_1}(\mathbf{x}; \mathcal{T}), \tag{11}$$

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 (11) $\mathcal{T}$ 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 [23].

Remark 2. *Theorem 1 follows from the classical Newton interpolation remainder formula. When a degree-k polynomial exactly interpolates the function at the knot points, the approximation error between knots is entirely determined by the function's $(k+1)$-th-order variation. Bounding this variation using the $(k+1)$-th-order Lipschitz constant yields*

the computable worst-case error bound in (9), which is zero at the knots and grows smoothly between them. Theorem 2 extends this result to the practical setting where the approximation does not exactly match the function at the knots. In this case, the total error naturally decomposes into two components: the interpolation error characterized by Theorem 1 and the error introduced by the residuals at the knot values. By the linearity of polynomial interpolation and the triangle inequality, these two contributions combine to produce the bound in (10). Equation (11) is obtained similarly: writing $|f(x) - \hat{f}(x)| = |h_2(h_1(x)) - \hat{h}_2(\hat{h}_1(x))|$*, we insert and subtract* $h_2(\hat{h}_1(x))$*, and bound the two resulting terms via the triangle inequality, so that propagation is governed by the Lipschitz constant of the outer function* h_2*.*

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

IV. PROBLEM FORMULATION AND OVERVIEW OF THE RESULTS

The applicability of NNs is limited in the absence of a measure of uncertainty or error, particularly in safety-critical settings. In this section, we formulate the problem of deriving distance-aware worst-case error bounds for NNs and discuss their role in safe control and other applications.

A desirable property of an uncertainty estimate is distance-awareness: it should increase monotonically as the input moves away from the training data. We formalize worst-case distance-aware uncertainty as follows.

To characterize worst-case uncertainty, let $f : \mathcal{X} \to \mathbb{R}$ ($\mathcal{X} \subset \mathbb{R}^d$) be an unknown target function, and let $\mathcal{F}$ be the admissible function class, which encodes the structural properties that we expect the target function $f \in \mathcal{F}$ to satisfy. A dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ is observed, where $y_i = f(\mathbf{x}_i) + \eta_i$, where η_i is observation noise that satisfies $|\eta_i| \leq \epsilon_{\max}$, where $\epsilon_{\max} \geq 0$[3].

Given this dataset and a tolerance parameter $\epsilon \geq 0$, we define the following ambiguity set

$$\mathcal{F}_\mathcal{D} := \{g \in \mathcal{F} | \, \ell(g(\mathbf{x}_i), y_i) \leq \epsilon, \forall (\mathbf{x}_i, y_i) \in \mathcal{D}\}, \tag{12}$$

where ℓ is a discrepancy measure. $\mathcal{F}_\mathcal{D}$ is a subset of admissible functions consistent with the data given the tolerance parameter ϵ. By choosing a larger ϵ, the ambiguity set $\mathcal{F}_\mathcal{D}$ enlarges, leading to more conservative bounds, as described next.

Consider a predictor $\hat{f}_{\Theta^*}(\mathbf{x})$ that can, for instance, be trained on $\mathcal{D}$ by minimizing

$$\Theta^* = \arg\min_\Theta \frac{1}{|\mathcal{D}|} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}} \ell(\hat{f}_\Theta(\mathbf{x}_i), y_i), \tag{13}$$

where Θ^* are optimized parameters of the model, and $|\mathcal{D}|$ denotes the size of the data set. Then the associated worst-case uncertainty bound (prediction error) at a test point $\mathbf{x}^*$ is

$$J_\epsilon(\mathbf{x}^*) := \sup_{g \in \mathcal{F}_\mathcal{D}} \ell(g(\mathbf{x}^*), \hat{f}_{\Theta^*}(\mathbf{x}^*)). \tag{14}$$

This quantity captures the worst-case deviation across all admissible target functions compatible with the observed data, up to the tolerance parameter ϵ. Determining $J_\epsilon(\mathbf{x}^*)$ is generally intractable, as it involves optimization over an infinite-dimensional function space, and in this work, we suitably restrict the class $\mathcal{F}$, as discussed next.

To formalize the problem, we use the definition of distance-awareness given in Sec. III (Definition 1).

Having established the worst-case error bound (14) and formalized the notion of distance-awareness, we now state the main problem.

Problem 1. *Given a dataset* $\mathcal{D}$ *and a trained model* $\hat{f}$*, construct an upper bound* $u_f(\mathbf{x}; \mathcal{D})$ *on the prediction error,*

$$|g(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq u_f(\mathbf{x}; \mathcal{D}), \quad \forall \mathbf{x} \in \mathcal{X} \;\; \& \;\; \forall g \in \mathcal{F}.$$

The bound needs to satisfy four properties:

(i) Validity—the bound holds for every function g *in the admissible class* $\mathcal{F}$ *defined in (12); (ii) Distance-awareness—*u_f *satisfies Definition 1; (iii) Tightness—*u_f *should neither substantially overestimate the true error nor underestimate it—defined as the functional* $\rho(u_f) := \int_\mathcal{X} [u_f(x; \mathcal{D}) - J_\epsilon(x)] dx$*, which we seek to keep small; (iv) Efficiency—*u_f *can be efficiently evaluated in terms of computational costs.*

Solving this problem is intractable, as it requires optimization over an infinite-dimensional function space. In this work, to take steps toward addressing the above problem, we restrict $\mathcal{F}$ to the class of Lipschitz continuous functions with known constant $\mathcal{L}_f$. If we assume both true and learned functions $\hat{f}$ have the same Lipschitz $\mathcal{L}_f$, then a trivial upper bound for (14) will be:

$$u_f^{triv}(\mathbf{x}^*; \mathcal{D}) = 2\mathcal{L}_f \|\mathbf{x}^* - \boldsymbol{\tau}^*\| + \epsilon_{\max} \geq J_\epsilon(\mathbf{x}^*), \tag{15}$$

where $\boldsymbol{\tau}^*$ is the nearest training point to the test point,

$$\boldsymbol{\tau}^* = \arg\min_{\mathbf{x} \in \mathcal{X}_\mathcal{D}} \|\mathbf{x}^* - \mathbf{x}\|, \tag{16}$$

where $\mathcal{X}_\mathcal{D} = \{\mathbf{x}_i : (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{D}\}$.

Using the distance function $d_u = \|\mathbf{x}^* - \boldsymbol{\tau}^*\|$, the trivial bound (15) is distance-aware because it depends on $\mathbf{x}^*$ and it is monotonically increasing in this distance. Particularly, the left-hand side of (1) is equal to $2\mathcal{L}_f \|\nabla_\mathbf{x} d_u\|^2$, which is non-negative and holds for any $\mathbf{x}^*$.

To obtain an efficient and tight bound in NNs, we leverage the function's Lipschitz continuity and propagate the error layer by layer through the model's structure. For simplicity, consider a two-stage composition $\hat{f}(\mathbf{x}) = \hat{h}_2 \circ \hat{h}_1(\mathbf{x})$, where the Lipschitz constants of $\hat{h}_1$ and $\hat{h}_2$ are $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, and $\mathcal{L}_f \leq \mathcal{L}_1 \mathcal{L}_2$. Also, let $u_{h_1}(\mathbf{x}) \geq \|h_1(\mathbf{x}) - \hat{h}_1(\mathbf{x})\|$

[3]Since our analysis is worst-case, we assume bounded noise. However, this framework naturally extends to the probabilistic setting: if the noise can be bounded with high probability, the deterministic ambiguity set (12) generalizes to a probabilistic feasible set [68], [69], $\{g \in \mathcal{F} \mid P(\ell(g(\mathbf{x}_i), y_i) \leq \epsilon_{\text{th}}) \geq 1 - \beta, \forall (\mathbf{x}_i, y_i) \in \mathcal{D}\}$, where the constraint is required to hold with probability at least $1 - \beta$ rather than deterministically.

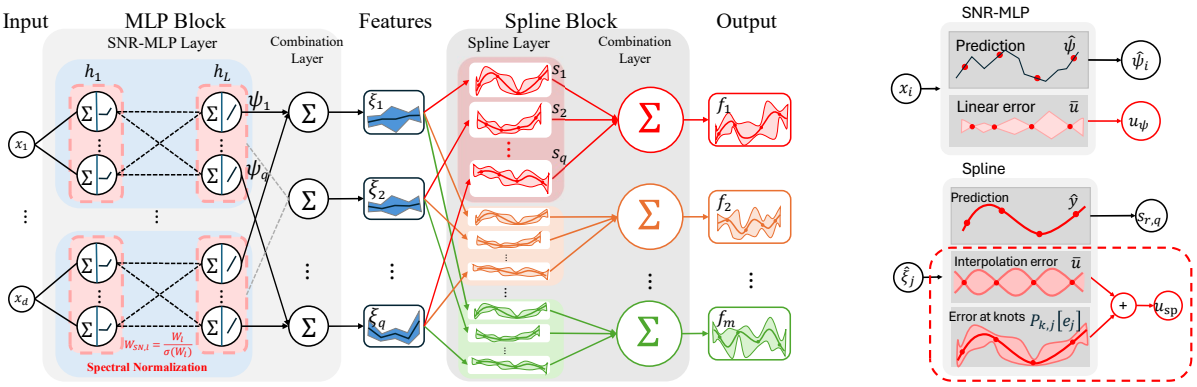

FIGURE 1. left) K-DAREK architecture consists of two blocks: MLP and spline. The MLP block contains one SNR-MLP for each input dimension. A combination layer unifies the output of SNR-MLPs into a feature vector. The spline block consists of a Spline Layer followed by a combination layer that constructs the model's final output. right) The error analysis of K-DAREK has two main components. A single SNR-MLP receives input $\mathbf{x}_i$ and computes a feature, $\hat{\psi}_i$, along with the linear error (21). A single Spline employs features $\hat{\xi}_j$ to calculate the spline values $s_{r,q}$ and spline error using interpolation error and error at the knot as described in (11).

and $u_{h_2}(\mathbf{z}) \geq \|h_2(\mathbf{z}) - \hat{h}_2(\mathbf{z})\|$ be the per-block error bounds. Then the total error can be decomposed into layer-wise errors:

$$|f(\mathbf{x}) - \hat{f}(\mathbf{x})| \leq u_{h_2}(\hat{h}_1(\mathbf{x})) + \mathcal{L}_2 u_{h_1}(\mathbf{x}). \tag{17}$$

This proposed error bound (17) is tighter than the trivial bound in (15) when the per-block error estimates u_{h_1}, u_{h_2} are tighter than their respective trivial bounds (see Remark 1 for more details). Note that in Equation (17), without loss of generality, $\epsilon_{\max}$ is implicitly considered inside u_h. In the proposed framework, its effect is recovered through the error-at-knots term in the spline error analysis.

Specifically, in this paper, we adopt a KKAN-style architecture,

$$\hat{f}(\mathbf{x}) = \hat{h}_{\text{sp}} \circ \hat{h}_{\text{MLP}}(\mathbf{x}),$$

where the MLP has Lipschitz constant $\mathcal{L}_{\text{MLP}}$, and the spline layer has Lipschitz constant $\mathcal{L}_{\text{sp}}$. The model architecture consists of two building blocks: MLP block and spline block, as shown in Fig. 1. In the MLP block, each input dimension is transformed by a spectrally normalized MLP layer that enforces a known Lipschitz constant. The resulting representations are summed to form a unified feature vector. The spline block then maps this feature vector to the output via splines, whose interpolation error can be bounded analytically.

As discussed next, while this approach does not fully solve Problem 1, our goal is a tractable worst-case bound that advances beyond every state-of-the-art baseline on at least one key criterion.

A. Overview of Applications

We validate our approach in both control-theoretic and machine learning applications, as described next. Safe navigation in unstructured environments: To validate K-DAREK in a safety-critical setting, we evaluate it on a multi-agent navigation task in an unstructured environment. In this scenario, a controlled agent must navigate from a start position to a goal while avoiding collisions with other agents that follow unknown and heterogeneous control policies [70]. The control architecture is two-layered: an model predictive control (MPC) controller first computes a goal-directed but potentially unsafe trajectory, and a control barrier function (CBF)-based safety filter then solves a quadratic program to find the closest safe control input that respects collision avoidance constraints, as depicted in Fig. 2. The key challenge is that the CBF requires bounds on the uncertain dynamics of agents. In [70], these bounds were obtained using a matrix-variate GP [71], which becomes computationally prohibitive as data grows. In this work, K-DAREK replaces the GP with distance-aware error bounds from the KKAN architecture, demonstrating that the framework preserves the structure of the original CBF safety condition, conditional on the validity of the assumed error bound and Lipschitz constant at a fraction of the computational cost.

Other machine learning applications: Beyond the safe control application, we evaluate K-DAREK on real-world regression tasks (e.g., Real Estate Valuation [72]) and Celebrity Attribute face detection dataset [73] and assess its uncertainty quantification against state-of-the-art approaches, including GP-based [14], [74], deep uncertainty methods [17], [30], [43], and worst-case analysis methods [23]. Across these benchmarks, we report two key metrics: the violation rate, which measures how often the true function value falls outside the predicted error bound, and sampled distance-awareness, which quantifies whether the model's uncertainty grows appropriately as test points move farther from training data. K-DAREK exhibits strong empirical distance-aware behavior, tight error bounds, low mean squared error (MSE), and computational efficiency compared to baselines.

Key takeaways: We compare the proposed K-DAREK method against state-of-the-art uncertainty methods, including SNGP, DUE, GP, and ensemble of KANs, across multiple metrics: training and inference process time, memory usage, error bound violation rate, MSE, and sampled distance-awareness. While K-DAREK does not dominate all baselines across every metric simultaneously, our results show that it

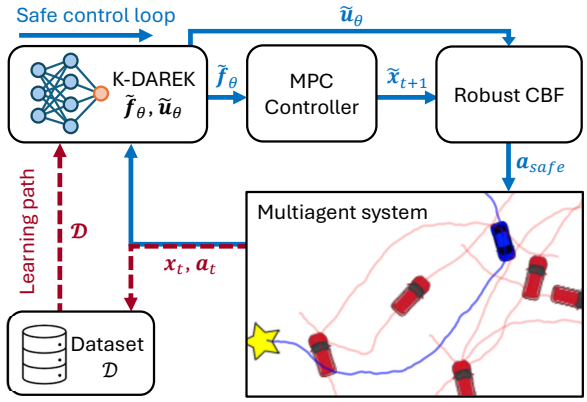

FIGURE 2. **Multi-agent safe control diagram. Red arrows → show the learning path. In this path, states $\mathbf{x}_t$ and controls $\mathbf{a}_t$ are collected as inputs and $\mathbf{x}_{t+1}$ as outputs in the dataset $\mathcal{D}$, and K-DAREK is trained to predict the system dynamics (26). Blue arrows → show the safe control loop. It passes learned dynamics $\tilde{\mathbf{f}}_\theta$ and estimated uncertainty $\tilde{\mathbf{u}}$ along with the nominal trajectory from the MPC controller $\tilde{x}_{t+1}$ to a robust CBF to produce the safe control $\mathbf{a}_{\text{safe}}$ and applies it to the controlled agent. The multi-agent plot in the diagram shows one of the successful trajectories using K-DAREK bounds, where the controlled agent (blue car) safely navigates to the goal location while avoiding the other agents (red cars), see Fig. 7.**

consistently **outperforms each baseline in at least one of these metrics** (and often in several) [cf. Tables 4, 8, and 10]. In particular, when NN approaches such as DUE and SNGP achieve competitive prediction accuracy, K-DAREK offers empirically stronger distance-aware behavior (higher SDA) and a lower error bound violation rate; conversely, when GP-based methods provide principled uncertainty bounds, K-DAREK scales significantly more favorably with dataset size. Taken together, these results demonstrate meaningful progress toward solving Problem 1, advancing beyond existing methods even if a complete solution remains open. In this sense, K-DAREK represents a tractable, efficiently computable estimate of the worst-case bound.

V. ARCHITECTURE AND METHOD

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

Building upon KAN, KKAN [41] insists on the two-layer structure of KAT (3) as a composition of two functional blocks, where ψ represents the inner block and Φ the outer block. The 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, and finally map it to the output space via a combined output layer. Intuitively, the KKAN inner block processes each scalar input coordinate independently in three steps: (1) lift the scalar x_p to a short vector of Chebyshev polynomial features $\mathbf{T}_n(x_p)$; (2) pass this feature vector through a small MLP to produce a hidden-space representation; (3) sum these per-dimension representations and map the result through the spline block to the output. For an input $\mathbf{x} \in \mathbb{R}^d$, intermediate feature dimension q, and an output dimension r, the structure of KKAN can be written as:

$$f_r(\mathbf{x}) = \sum_{i_s=1}^{q} s_{r,i_s}\Big(\sum_{p=1}^{d} \underbrace{\mathbf{1}_{on}^\top \mathbf{T}_n\big(\mathrm{MLP}_{p,i_s,o}(\mathbf{T}_n(x_p))\big)}_{\psi_{p,i_s}(x_p)}\Big), \quad (18)$$

where $\mathbf{T}_n(\mathbf{y}) := (T_0(\mathbf{y}), \ldots, T_{n-1}(\mathbf{y})) \in \mathbb{R}^{on}$ is the Chebyshev basis operator that expands each dimension of an input vector $\mathbf{y} \in \mathbb{R}^o$ to the first n Chebyshev polynomials of the first kind, $T_n(\mathbf{y}) := \cos(n \arccos(\mathbf{y})) \in \mathbb{R}^o$. 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. $\mathrm{MLP}_{p,i_s,o} : \mathbb{R}^n \to \mathbb{R}^o$ is an MLP that produces a vector output of a desired dimension o.

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 at the cost of the model's expressiveness, as it removes the nonlinear contribution of the higher-order Chebyshev terms. Second, we use a spectrally normalized MLP (SNR-MLP) instead of $\mathrm{MLP}_{p,i_s,o}$ in (18), which allows us to control the Lipschitz constant of the MLP.

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

A. 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 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 (18)). 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 **(left)** and (18), 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 $\mathbf{f} = (f_1, \ldots, f_m) = (\sum_{i_s=1}^q s_{r,i_s})_{r=1}^m \in \mathbb{R}^m$.

B. ERROR ANALYSIS FRAMEWORK FOR K-DAREK

We divide the error analysis framework into five components described in this subsection. At a high level, computing the error bound at a test point x^* proceeds in three stages, mirrored in Alg. 1: (1) pass x^* through the MLP block to get its feature vector and the MLP block's error via (21); (2) for each spline feature, locate its knot interval by binary search, fit a Newton polynomial to the nearest knots using the errors already observed there during training, and evaluate the local spline error via (16); (3) propagate the MLP error through the spline block's Lipschitz constant and add it to the spline error via (24).

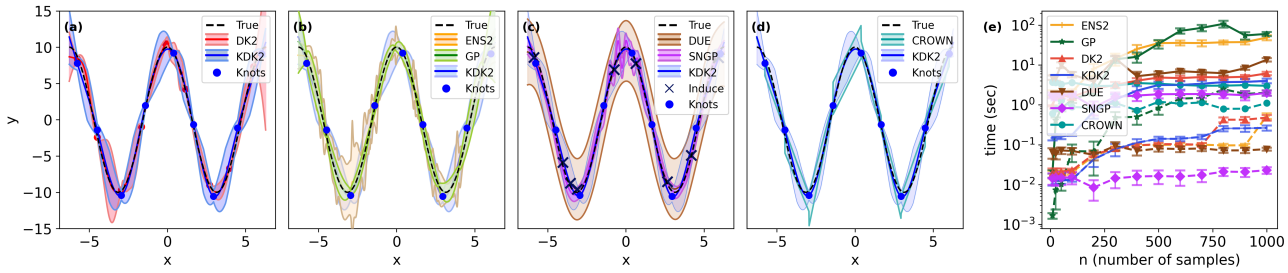

FIGURE 3. **Cosine function approximation experiment (qualitative comparison). (a) Error estimation comparison of DK2 and KDK2, showing that KDK2 achieves smoother function fitting without requiring an explicit regularizer. In both models, the error bounds increase as test points move farther from the nearest knot, reflecting distance-aware error behavior. (b) 3σ uncertainty bounds comparison between ENS2 and a GP. The ENS2 uncertainty bounds lack distance-awareness and do not follow a consistent spatial pattern, in contrast to DK2, KDK2, and GP, each shown alongside KDK2 for direct comparison. (c) 3σ uncertainty bounds comparison between DUE and SNGP, shown alongside KDK2 for direct comparison. Inducing points are projected back into the input domain and plotted as $\times$. Both uncertainties are almost uniformly distributed across the input domain in this sparse dataset. (d) Comparison of the KDK2 error bound (blue) with the certified output bounds computed using forward CROWN through `auto_LiRPA`. The learned model for CROWN is same KDK2 and intervals are defined between knots. CROWN certifies the model output over each specified perturbation region, whereas KDK2 estimates an error bound based on the distance from the nearest training knot. (e) Process time comparison of KDK2 (blue), DK2 (red), GP (darkgreen), ENS2 (orange), DUE (brown), SNGP (purple), and CROWN (cyan) models. Training time (solid line) and inference time (dashed). The error bars indicate the variability observed across repeated runs of the experiment.**

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 blocks ($\mathcal{L}_{\text{MLP}} = \mathcal{L}_{\text{sp}} = \sqrt{\mathcal{L}_f^{(1)}}$).

Find feature of test input $\boldsymbol{\psi}^* = \Psi(\mathbf{x}^*)$

Find $\mathcal{L}_{\overline{\text{sp}}} = \mathcal{L}_{\text{sp}}/q$, $\mathcal{L}_{\overline{\text{sp}}}^{(k+1)} = \mathcal{L}_f^{(k+1)}/q$, and $\mathcal{L}_{\overline{\text{MLP}}} = \mathcal{L}_{\text{MLP}}/d$

Find u_{MLP} using [Eq. (21)] /* $\mathcal{O}(\log_2(m_k)d)$ */

for $n \in \{1, \ldots, q\}$ **do**

  /* $\mathcal{O}(\log_2(m_k))$ `binary search`. */

  Find j such that $\boldsymbol{\kappa}_n[j] \leq \boldsymbol{\psi}_n < \boldsymbol{\kappa}_n[j+1]$.

  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 $u_{\text{sp}_n}(x_n^*; \boldsymbol{\tau}_n)$ [Eq. (10)].

Compute error bound over f [Eq. (24)] /* $\mathcal{O}(Lm_h^2) + \mathcal{O}(qm_k)$ */

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.

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, 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*, which contributes to the overall error bound. Note that matrices $\mathcal{T}$ and Y are paired samples selected from the training data, and to compute the knot matrix K, we pass $\mathcal{T}$ through the network and collect the resulting features.

Remark 3. *Choosing m_k samples from the training data influences the training and the downstream error bound, since the knots determine both the support of the spline and the reference points for the error-at-knots term. Several selection strategies are possible, including uniform random sampling, k-means, weighted k-means [75], Chebyshev points [76], and Latin hypercube sampling [77], all of which are implemented in the accompanying codebase. For the results reported in this paper, we use k-means because it places knots in regions where training data is present. We compare these strategies in the ablation study (Table 5); a more exhaustive comparison of their effect on bound tightness and distance-awareness across additional tasks and datasets is left for future work.*

2) Error of MLP Block: MLP block has d SNR-MLPs, one per input dimension, and each SNR-MLP has L layers. We denote the Lipschitz constant of the block by $\mathcal{L}_{\text{MLP}}$ and each SNR-MLP by $\mathcal{L}_{\text{MLP}_i}$, with $\mathcal{L}_{\text{MLP}} \leq \sum_{i=1}^{L} \mathcal{L}_{\text{MLP}_i}$. Given a test point $\mathbf{x}^*$, the error of the i-th SNR-MLP is

$$u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) = \mathcal{L}_{\text{MLP}_i} \|x_i^* - \tau_i^*\|, \quad \text{where} \quad \tau_i^* = \arg\min_{\tau^\dagger \in \boldsymbol{\tau}_i} \|\tau^\dagger - x_i^*\|. \tag{19}$$

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:

$$u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) = \sum_{i=1}^{d} u_{\text{MLP}_i}(x_i^*, \boldsymbol{\tau}_i) = \sum_{i=1}^{d} \mathcal{L}_{\text{MLP}_i} \|x_i^* - \tau_i^*\|. \tag{20}$$

If all SNR-MLPs have equal Lipschitz constant $\mathcal{L}_{\overline{\text{MLP}}}$ (so that $\mathcal{L}_{\text{MLP}} \leq d\,\mathcal{L}_{\overline{\text{MLP}}}$), this expression simplifies to

$$u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}) = \mathcal{L}_{\overline{\text{MLP}}} \sum_{i=1}^{d} \|x_i^* - \tau_i^*\|. \tag{21}$$

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

3) 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:

$$u_{\text{sp}_{r,i}}(\xi_i^*, \boldsymbol{\kappa}_i) \coloneqq \bar{u}_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i) + \left\|\mathcal{P}[(\frac{\hat{f}_r(\boldsymbol{\tau}) - \mathbf{y}_r}{q})^{\text{sp}}_{[j]}](\xi_i^*)\right\|. \tag{22}$$

The factor $1/q$ comes from the equal sharing of error across the q splines within each group, as discussed below under Lipschitz and error sharing. The worst error bound of the r-th group of splines is given by:

$$u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) \coloneqq \sum_{i=1}^{q} u_{\text{sp}_{r,i}}(\xi_i^*; \boldsymbol{\kappa}_i). \tag{23}$$

4) Total Error: Having established the error for both the MLP and spline blocks in Equations (21) and (23), 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:

$$u_{f_r}(\mathbf{x}^*) = u_{\text{sp}_r}(\boldsymbol{\xi}^*, K) + \mathcal{L}^1_{\text{sp}} u_{\text{MLP}}(\mathbf{x}^*, \mathcal{T}). \tag{24}$$

Here, $\mathcal{L}^1_{\text{sp}}$ 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.

5) Lipschitz and Error Sharing: To use the provided error analysis and ensure that the neural network preserves the stability and bounded sensitivity of the target function f, the approximated model must exhibit Lipschitz continuity consistent with the Lipschitz constant of $\mathcal{L}_f$. To achieve this, inspired by [30], we divide the Lipschitz constant of the true function equally between the MLP and spline block $\mathcal{L}_{\text{MLP}} = \mathcal{L}_{\text{sp}} = \sqrt{\mathcal{L}_f}$. We divide the Lipschitz constant between SNR-MLP components equally $\mathcal{L}_{\overline{\text{MLP}}} = \mathcal{L}_{\text{MLP}}/d$, and also, we apply the Lipschitz constant of $\mathcal{L}_l = \sqrt[L]{\mathcal{L}_{\overline{\text{MLP}}}}$ using spectral normalization on each layer of these components. Also, we assign an equal Lipschitz constant for each spline in the spline block as $\mathcal{L}_{\overline{\text{sp}}} = \mathcal{L}_{\text{sp}}/q$.

Additionally, we assume the error at knots arises solely from the spline block [66], so we divide it equally between splines $(\hat{f}_r(\boldsymbol{\tau}) - \mathbf{y}_r)/q$.

Remark 4. *The worst-case guarantee established above (24) holds under the assumption that $\mathcal{L}_f$ is a valid upper bound on the true Lipschitz constant of f. In practice, $\mathcal{L}_f$ is estimated numerically from the training data rather than known exactly.*

Remark 5. *Allocating Lipschitz constants and error divisions are challenging problems, and several approaches, including heuristic allocation and optimization-based methods, can be used for this purpose. However, these questions are not the focus of this paper and have been explored in the DAREK paper [66]. We defer a more detailed study of these options for K-DAREK to future work.*

The key theoretical difference between DAREK (built on KAN) and K-DAREK (built on KKAN) lies in the depth and compositional structure of the spline sub-network. DAREK stacks spline layers homogeneously to arbitrary depth, Eq. (4), whereas K-DAREK replaces most of that stack with a single dense (MLP) block. In DAREK, the worst-case error must be propagated recursively through L nested spline compositions, with each layer's Lipschitz constant scaling all upstream error terms. K-DAREK instead has only two blocks, so its error propagation requires applying the two-layer bound (24) only once. Additionally, it enforces a Lipschitz constant of the MLP block through spectral normalization. Consequently, K-DAREK's bound is less sensitive to Lipschitz division and error sharing.

C. ALGORITHM COMPLEXITY

The computational complexity of the proposed and baseline models varies based on their architectures. For an ensemble of n_e NNs with L layers and m_h hidden units per layer, the complexity of inference scales as $\mathcal{O}(n_e L m_h^2)$. In contrast, a GP depends on the number of training samples n and has a training computational complexity of $\mathcal{O}(n^3)$, and the inference complexity is $\mathcal{O}(n^2)$. DAREK model is composed of L layers with m_h hidden units per layer and m_k knots in each spline, and it achieves an inference complexity of $\mathcal{O}(\log_2(m_k) L m_h) + \mathcal{O}(L m_h^2)$. The proposed K-DAREK architecture has d MLPs with L layers and m_h hidden units per layer alongside one spline layer with m_k knots per spline and q splines (for a single-output network).

Regardless of the number of layers, K-DAREK's error computation involves two separate knot-search stages, one for each block; this results in a total inference complexity of $\mathcal{O}(\log_2(m_k)(d+q)) + \mathcal{O}(L m_h^2) + \mathcal{O}(q m_k)$, where the first term captures the two knot-search stages, and the second is the shared MLP dense-computation cost. The third term is the cost of evaluating the spline polynomials. For the cost of each step in K-DAREK, please refer to Algorithm 1. The computational cost is compared between different methods in Fig. 3 **(a)**.

VI. EXPERIMENTS AND RESULTS

We evaluated K-DAREK [4] against different models across multiple tasks, including function approximation, real-world regression datasets [72], feature collapse, overgeneralization in regions with missing data, face bounding-box prediction, and multi-agent safe control. We compare against Gaussian process (GP)-based models with an radial basis function (RBF) kernel and learnable hyperparameters, including the length scale and variance amplitude. These models include an exact GP (GP) [14] and approximate GP (APXGP) with learnable inducing point locations [74]. We also compare against deterministic uncertainty estimation (DUE) [43], and spectral-normalized neural Gaussian process (SNGP) [30]. For DUE and SNGP, we use a four-layer MLP with 128 hidden units [43] as the feature extractor and estimate the corresponding posteriors using Monte Carlo (MC) sampling with 64 posterior samples. We further compare against ensembles [17] of KAN networks, including one-layer (ENS1) and two-layer (ENS2) models. We use 3σ intervals as uncertainty bounds for probabilistic models (GP, APXGP, DUE, SNGP, ENS1, and ENS2). Additionally, we consider Lipschitz-based approaches DAREK and K-DAREK. For DAREK models built upon KAN (all-spline networks), we use DK1, which has one spline layer, and DK2, which has two stacked spline layers. We consider two variants of our proposed model: two-layer K-DAREK (KDK2), which consists of a one-layer SNR-MLP followed by a one-layer spline network, and the three-layer K-DAREK (KDK3), which consists of a two-layer SNR-MLP followed by a one-layer spline network. All spline layers have B-splines of degree $k = 3$ (cubic B-splines). The Lipschitz constants for the datasets are computed numerically from the training data.

We use several metrics, including the mean squared error (MSE) is defined as $MSE = \sum_{i=1}^{n} |\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)|^2/n$, and the root mean squared error (RMSE) is given by $\text{RMSE} = \sqrt{\sum_{i=1}^{n} |\hat{f}(\mathbf{x}_i) - f(\mathbf{x}_i)|^2/n}$, where n is the number of test points. We also use the violation rate, defined as the percentage of test points where the true value lies outside the predicted error bounds. In addition, we use the intersection over union (IoU), defined as $\text{IoU} = |\mathcal{A} \cap \mathcal{B}|/|\mathcal{A} \cup \mathcal{B}|$, where $\mathcal{A}$ and $\mathcal{B}$ denote the predicted and ground-truth regions, respectively [78]. We also use the sampled distance-awareness (SDA) metric defined in equation (2).

Function approximation: The goal of this experiment is to learn the function $f(x) = 10\cos(x)$ over the range $[-2\pi, 2\pi]$ using 50 training data points. Both DK2 and KDK2 models are two-layer models with 5 hidden units. For ENS2, we use ten KAN models with the same architecture as DK2. All spline-based models use 9 knots and are trained for 500 epochs with a learning rate of 0.1, decayed by a factor of 0.9 every 50 epochs. In this experiment, DUE and SNGP use 2 layers and 20 features. DUE uses 9 inducing points, and SNGP uses 9 random Fourier kernels (same as the number of knots in DK2 and KDK2). To ensure a fair comparison, we used the same input knots for the first layer in DK2, KDK2, and ENS2. The GP is trained on 9 data points selected using K-means from the training dataset. We trained DUE and SNGP for 5000 steps at a learning rate of 0.01 to ensure optimal performance, as they are very sensitive to this rate. Each experiment was repeated 10 times to compute error bars.

Fig. 3 **(a)** compares the error estimation of DK2 and KDK2. The results suggest that KDK2 achieves a more generalized fit, reducing error at knots. While the interpolation error may increase slightly due to the inner block's linear approximation (rather than higher-order polynomials), the overall fit is more robust. 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 [cf. Remark 5]. Fig. 3 **(b)** compares the uncertainty bounds produced by ENS2 and GPs. Fig. 3 **(c)** compares the uncertainty bounds produced by DUE and SNGP. Also, since DUE and SNGP use approximate GPs, their posteriors do not match the exact GP posterior. Table 3(a) shows that KDK2 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 "Size" column shows the total number of learnable parameters. The CROWN comparison shown in Fig. 3 **(d)** is discussed further in next experiment.

Fig. 3 **(e)** illustrates the process time complexity of DK2 and KDK2 compared to GP, ENS2, DUE, and SNGP. Process time measures the total CPU work performed by the process (summed across cores). For this plot, we train all the models for 30 epochs across all experiments. The results show that KDK2 is slightly more efficient than DK2 across all dataset sizes and significantly outperforms both GP and ENS2 for datasets larger than 300 samples. DUE requires higher training time and lower inference time than KDK2. SNGP has the lowest training and inference time among all tested models. These results are consistent with the algorithmic complexity analysis provided in subsection V-C.

We repeat the same experiment using **extended knots** for spline-based models. For DK2, MSE drops from 0.789 to 0.102, and for KDK2, from 2.151 to 0.060. KDK2 with extended knots achieves the lowest MSE and zero coverage violation while using fewer learnable parameters (Table 3 (b)). These improvements come at the cost of additional learnable parameters (size increased from 45 to 76). This

[4] The splines in the spline block are adopted from the KAN implementation [13] with learnable coefficients. Unlike KAN, which places knots on a uniform grid, K-DAREK selects knot locations from the training data (see Knot Selection in subsection V-B-1). KAN also uses *extended knots*, placing k extra knots before and after the spline domain to improve boundary behavior at the domain edges. Because the true function is unknown outside the training domain, K-DAREK uses extended knots for prediction, but excludes them from the core error bound analysis. Their effect on model performance is evaluated in the function approximation experiment.

TABLE 3. Error estimator comparison for the function approximation experiment, with and without extended knots. K-DAREK achieves the lowest violation with fewer learnable parameters. The exact GP attains the lowest MSE.

Model	MSE loss	Violations(%)	Size
a) without extended knots			
DUE	0.546 ± 0.075	**0.000 ± 0.000**	734
SNGP	0.396 ± 0.115	1.260 ± 2.179	726
ENS2	1.658 ± 0.655	4.310 ± 6.231	700
GP	**0.168 ± 0.024**	**0.000 ± 0.000**	N.A.
DK2	0.789 ± 0.349	8.480 ± 4.778	70
KDK2	2.151 ± 0.986	0.270 ± 0.359	**45**
b) with extended knots			
ENS2	0.167 ± 0.128	0.360 ± 0.424	1320
DK2	0.102 ± 0.062	1.380 ± 1.992	132
KDK2	**0.060 ± 0.008**	**0.000 ± 0.000**	**76**

indicates that extended knots improve smoothness, stability, and predictive accuracy.

Adapted-CROWN comparison experiment. To empirically ground the distinction between K-DAREK's error bound and system verification methods, we compare K-DAREK against forward CROWN (implemented using `auto_LiRPA`) [64] on the same cosine function approximation task used in the Function Approximation experiment above, $f(x) = 10\cos(x)$ over $[-2\pi, 2\pi]$ with 50 training samples. CROWN certifies a fixed, already-trained network, so we used the same trained KDK2 model in both roles: as the model whose error K-DAREK bounds, and as the network CROWN certifies. We constructed and implemented a tight affine (CROWN-style) bound using the same training knots of KDK2 as CROWN's perturbation anchors, so both methods are anchored at identical reference points. CROWN bounds the trained network $\hat{f}$: given a fixed input-perturbation set, it computes an affine relaxation from $\hat{f}$'s known architecture and weights, with no reference to the unknown target f. K-DAREK's bound (24), by contrast, bounds the deviation $|f(x^*)-\hat{f}(x^*)|$ at a single test point x^*, under the assumption that f is Lipschitz. These are bounds on different quantities — one on $\hat{f}$'s sensitivity over a region, the other on $\hat{f}$'s deviation from f at a point. The comparison here is meant to illustrate the two methods' differing design objectives, not to suggest either dominates the other.

We partition the input domain into intervals defined by consecutive knots. Within each intervals, splines of KDK2 are polynomial functions, $\hat{f}_{[j]}$. We find a tight local verification bound by solving the following optimization problem for each polynomial in every spline:

$$\begin{aligned} &\arg\min_{\alpha,\beta} \int_{\tau_j}^{\tau_{j+1}} \left(\alpha x + \beta - \hat{f}_{[j]}(x)\right) dx \\ &\text{s.t.} \quad \alpha x + \beta \geq \hat{f}_{[j]}(x), \end{aligned} \tag{25}$$

with the lower bound obtained from the mirrored problem (inequality reversed). We solve (25) numerically via sequential least-squares quadratic programming (SLSQP); this is a new numerical procedure introduced for this comparison and is not part of DAREK's original formulation. The affine constraints are checked on a fine grid and re-certified on a finer grid, so the reported bounds are numerically validated under the adopted discretization rather than exact continuous-domain certificates. We then propagate the bounds through the entire model using `auto_LiRPA`.

Fig. 3 **(d)** illustrates the resulting bounds, together with a process-time comparison (Fig. 3 **(e)**). Computing the adapted-CROWN bound is substantially more expensive at query time than K-DAREK, inference takes roughly four times as long at $n = 1000$ (1.10 s vs. 0.26 s), though this may reflect our unoptimized implementation of the adapted bound rather than an inherent limitation of the approach. Fitting the per-interval linear relaxations is a one-time cost, computed once and reused across all test queries, rather than repeated for every prediction. The CROWN bound is tight immediately at the anchor knots. Because it is built from an independent linear relaxation fit to each knot-to-knot interval, the bound necessarily widens away from the interval's fitted point and is discontinuous across interval boundaries, which is a general property of local piecewise-linear relaxation methods, not specific to this implementation. In the outermost intervals, bounded by padding knots outside the training domain, the bound widens substantially, since these regions extrapolate beyond where the underlying spline was fit to data. K-DAREK's error bound (KDK2), in contrast, remains smooth and continuous throughout, widening gracefully with distance from the nearest knot. This difference reflects the two methods' differing design objectives rather than a shortcoming of either approach.

Adapting CROWN itself to directly bound $|f(\mathbf{x}) - f^{(}\mathbf{x})|$ is left for future work; one possible direction is incorporating training-data proximity into CROWN's relaxation, analogous to how K-DAREK's bound depends on distance to the nearest knot.

Fixed budget experiment: In this experiment, we isolate the effect of architecture from model capacity by fixing the parameter count, training data, and the optimization budget across all methods. We constrain the number of learnable parameters to about 200 for NN models [5], and use a fixed training and test set of 1000 samples uniformly drawn from the 1D function $\cos(x)$ over $x \in [-2\pi, 2\pi]$. All models are trained for 1000 epochs with a learning rate of 0.01, which decays by a factor of 0.95 every 100 epochs. Within this budget, we chose the architectures as:

- DK2: Two layers, 5 hidden units, 12 knots, and cubic splines with extended knots.

[5]Table 4's Size column reports the exact parameter count for each model. Learnable parameter counts follow directly from each architecture's discrete design choices, such as layers, hidden units, knots, and inducing points, none of which can be tuned continuously to hit exactly 200. Within this constraint, architecture choices for each model family were made to preserve a representative, non-degenerate instance of that method, rather than minimizing complexity at the expense of a baseline's typical operating regime.

- KDK2: One dense layer and one spline layer with 10 hidden units, 10 knots, and cubic splines with extended knots.
- GP: RBF kernel and trained on all training samples to optimize kernel hyperparameters.
- APXGP: RBF kernel and 13 inducing points.
- DUE: One residual layer, 8 hidden units, 7 inducing points, and an RBF kernel.
- SNGP: One residual layer, 10 hidden units, 25 GP features, and 20 random features.
- ENS2: Three KAN models, each with two spline layers, 3 hidden units, 4 knots, and cubic splines with extended knots.

We report the MSE, violation rate, size, average bound width (Avr. u_f), memory usage, training process time, and inference process time in Table 4. Each experiment is repeated 10 times, and the mean and standard deviation across runs are computed. ENS2 incurs the highest memory usage (13.36 KB) and the highest MSE among all models because each KAN model in the ensemble is constrained to a small parameter budget, limiting the ensemble's overall expressiveness. The exact GP attains the lowest MSE (0.002) and the tightest bound on average (0.02) among the baselines but exhibits the highest violation rate (36%) because its 3σ credible interval relies on a posterior variance that collapses as the sample density increases, and it also incurs the highest inference time (912.39 ms). SNGP is the fastest in both training and inference times, but it requires large memory and yields high MSE and violation rates. KDK2 achieves the lowest MSE (0.001) and zero violations in the evaluated test samples with a wider average bound (2.07). DK2 and KDK2 are best in training time after SNGP; however, they incur higher inference time than the probabilistic baselines with fixed budget. The source of this overhead is architectural: unlike SNGP/DUE's single deterministic forward pass, K-DAREK additionally performs a per-feature knot search and local Newton polynomial fit to evaluate the error bound at each test point (Algorithm 1, lines 7–8), which our current implementation executes sequentially across splines. This can be reduced by parallel processing to 17.28 ± 1.76 ms and 22.93 ± 1.24 ms for inference time of DK2 and KDK2, respectively.

Ablation experiment: We conduct an ablation study on the same cosine function task used in the Function Approximation ($f(x) = 10\cos(x)$, 50 training points), repeated over 10 seeded trials, and report MSE, violation rate, and average bound width. We compare the full K-DAREK model (KDK2 with 5 hidden units, 9 knots, and cubic spline) against ablated variants that replace the MLP block with an additional spline layer (yielding an all-spline architecture), remove spectral normalization, vary the knot-selection strategy (random, Latin hypercube sampling, Chebyshev points, weighted k-means, or a fixed, equally spaced knots, in place of the plain k-means used elsewhere in this paper), and vary spline capacity (order $k = 1$, or number of knots $m_k = 5$ or $m_k = 15$, in place of the default cubic splines, k=3, with $m_k = 9$).

TABLE 4. Fixed-budget experiment on $\cos(x)$ over $[-2\pi, 2\pi]$ with 1000 training samples and $\sim$200 learnable parameters per model. Process time is the total CPU work performed (summed across cores) and averaged over 10 runs after 3 warm-up passes. Memory is the amount of memory required by the model's parameters. K-DAREK achieves the lowest MSE and zero violations, while ranking among the fastest in training. We highlight the best in bold, the *second-best in gray*, and the worst in red.

Model	MSE	Violation	Size	Avr. u_f	Memory (KB)	Train Time (S)	Inf. Time (ms)
APXGP	0.092	**0.000**	200	0.23	*0.84*	136.573 ± 3.278	72.45 ± 6.06
GP	*0.002*	0.364	**5**	**0.02**	**0.07**	115.021 ± 2.053	912.39 ± 104.57
DUE	0.036	**0.000**	204	1.52	0.93	185.972 ± 3.909	*63.34 ± 6.63*
SNGP	0.019	0.037	201	0.41	7.05	**44.150 ± 0.736**	**10.73 ± 2.52**
ENS2	0.099	0.028	216	0.89	13.36	129.783 ± 1.297	68.14 ± 7.25
DK2	0.009	*0.001*	200	4.09	5.39	*79.809 ± 0.906*	144.02 ± 12.86
KDK2	**0.001**	**0.000**	200	2.07	5.53	91.624 ± 3.009	220.15 ± 13.12

The results are reported in Table 5. Replacing the MLP block with an additional spline layer increases MSE (0.108 vs. 0.060) and introduces violations (1.680%) where the full model has none, supporting the MLP block's role in producing a more stable fit. Removing spectral normalization leaves MSE and violation rate largely unchanged, but the average bound width grows by roughly two orders of magnitude (2379.294 vs. 10.122), showing that without spectral normalization the MLP block's Lipschitz constant is effectively uncontrolled, producing a technically valid but highly conservative bound. Among knot-selection strategies, the fixed, equally spaced grid achieves the lowest MSE and zero violations in the evaluated test samples, and weighted k-means performs comparably to the plain k-means used elsewhere in the paper (0.059 vs. 0.060 MSE, both zero violations). Random knot selection and Chebyshev points perform worse than k-means strategies, and 5 knots is too few to constrain the bound tightly, producing the widest average bound width among all variants. The model with 15 knots achieves a low MSE, and its average bound width is smaller because the distance to the worst-case test point decreases as the number of knots increases.

TABLE 5. Ablation study (mean ± std over $N = 10$ trials) isolating the effects of the MLP block, spectral normalization, knot selection strategy, and spline order/knots size (m_k).

Arm	MSE	Violation (%)	Avr. u_f
K-DAREK (KDK2)	0.060 ± 0.008	0.000 ± 0.000	10.122 ± 1.525
No MLP block (all spline)	0.108 ± 0.062	1.680 ± 1.968	5.399 ± 5.606
No spectral norm	0.064 ± 0.016	0.000 ± 0.000	2379.294 ± 3116.110
Knots: random	0.336 ± 0.308	0.270 ± 0.447	94.485 ± 78.996
Knots: LHS	0.164 ± 0.046	0.000 ± 0.000	16.186 ± 7.727
Knots: Chebyshev	0.199 ± 0.020	0.000 ± 0.000	15.180 ± 2.247
Knots: weighted k-means	0.059 ± 0.015	0.000 ± 0.000	11.077 ± 2.438
Knots: fixed (equally spaced)	0.032 ± 0.008	0.000 ± 0.000	27.164 ± 12.930
Spline $k = 1$	0.631 ± 0.164	0.050 ± 0.071	6.977 ± 0.240
Spline knots $m_k = 5$ (small)	0.549 ± 0.259	0.010 ± 0.032	209.974 ± 56.002
Spline knots $m_k = 15$ (large)	0.053 ± 0.006	0.070 ± 0.082	2.614 ± 0.529

Sensitivity to Lipschitz misestimation: We evaluate how the violation rate and average bound width respond to underestimating the Lipschitz constant. Using the same cosine function task with 5% output noise (normal distribution) added to the training and test data, we scale the true

Lipschitz constant $\mathcal{L}_f$ by a factor ranging from 0.1 to 1.0 and recompute the resulting error bound. We train KDK2, a similar architecture introduced in Function approximation experiments and trained for 500 epochs with a learning rate of 0.1, decayed by a factor of 0.9 every 50 epochs, with the corresponding Lipschitz constant reported. Lipschitz constant of the noiseless true function is 10. Table 6 reports the results. RMSE remains essentially unchanged across the sweep (0.5815–0.5832). Violation rate rises sharply as the Lipschitz constant is underestimated, from 1.5% at the full estimated value to 44.5% at 10% of that value, while the average bound width shrinks correspondingly (8.976 to 0.518). This confirms the mechanism described in Remark 4: underestimating $\mathcal{L}_f$ produces a tighter but less reliable bound, with violation rate increasing as the degree of underestimation grows. We note that, even with the fully estimated Lipschitz constant, this experiment shows that the small residual violation rate (1.6%) is due to the added output noise rather than a limitation of the Lipschitz estimate itself.

TABLE 6. Sensitivity of K-DAREK's error bound to underestimating the Lipschitz constant $\mathcal{L}_f$, on the cosine function task with 5% output noise. $\mathcal{L}\%$ is the fraction of the numerically estimated $\mathcal{L}_f$ used to compute the bound.

$\mathcal{L}\%$	RMSE	Violation (%)	Avr. u_f
0.10	0.5815	44.5	0.518
0.50	0.5819	6.6	2.321
1.00	0.5832	1.5	8.976

Real data regression experiment: We evaluate model performance on six real-world regression datasets from the UCI repository [72]: Concrete Compressive Strength ($n = 1030$, 7 continuous features), which predicts the compressive strength of concrete from its ingredients and age; Airfoil Self-Noise ($n = 1503$, 3 continuous features), which predicts aerodynamic noise generated by airfoils from operating conditions; Combined Cycle Power Plant ($n = 9568$, 4 continuous features), which predicts the net electrical power output of a power plant from environmental variables; Red and White Wine Quality (Red Wine has $n = 1599$, 11 continuous features and White Wine has $n = 4898$, 11 continuous features), which predict wine quality scores from physicochemical measurements; and Real Estate Valuation ($n = 414$, 6 continuous features), which predicts residential property values from location- and property-related attributes. We report only the number of continuous input features, as categorical features are excluded from our experiments. For results in Table 7, models are trained using only the first two input features, whereas Table 8 reports results using all *continuous* features. Each experiment is repeated 20 times, and we report the mean and standard deviation of RMSE and violation rate across runs for all datasets.

Training is performed on the standardized dataset with the Adam optimizer and a decaying learning rate. For the Power Plant and White Wine datasets, we use a subset of 2000 samples to make running the GP computationally feasible. All models have five hidden units, and ENS2 consists of five models. The MLP model has two layers with 64 hidden units, and the KAN model has two layers with 5 hidden units. DUE uses 20 inducing points, and SNGP uses 1024 random Fourier kernels. MLP, KAN, GP, DK1, DK2, KDK2, KDK3, and ENS2 are trained for 1000 epochs using the Adam optimizer, starting with a learning rate of 0.1 and applying a decay rate of 0.9 every 100 epochs. DUE and SNGP were trained for 5000 steps with a learning rate of 0.01.

In most experiments, the proposed models (KDK2 and KDK3) achieve the best or the second-best performance in terms of RMSE or violation rate. The results demonstrate that Lipschitz-based error bounds achieve stronger empirical coverage than DUE, SNGP, and ENS2.

We note that a low violation rate alone does not establish that a bound is tight, since a sufficiently conservative bound trivially achieves zero violations. Remark 1 gives the condition under which our interpolation bound is tight (equality in (9)).

Feature collapse experiment. We repeat the experiment in Fig. 1 of [43], using the same reported setup for DUE and SNGP, to evaluate whether DAREK and K-DAREK suffer from feature collapse and whether they overgeneralize in uncertainty estimation. We additionally include GP and APXGP in the experiments. There are two datasets: a small dataset with 1k training samples and a large dataset with 1000k training samples. All spline-based models (DK2, DK3, KDK2, KDK3) use 15 knots per spline and 5 units for intermediate layers. APXGP uses 15 inducing points. As shown in Fig. 4, among the compared methods, SNGP tends to overgeneralize as the dataset grows, with its uncertainty converges to zero; GP/APXGP, DUE, and our method (DK/KDK) all avoid this failure mode at both dataset sizes.

Uncertainty of missing data (Lipschitz vs. Probabilistic perspective). Probabilistic models estimate uncertainty based on observed data and model assumptions (such as priors); however, Lipschitz-based methods provide worst-case guarantees independent of data distribution. This experiment explicitly shows this difference on datasets where the ground-truth function contains localized structure that is entirely unobserved during training. We construct one-dimensional synthetic datasets as:

$$f(x) = g(x) + 10\exp(-\frac{(x)^2}{8}),$$

which has a Gaussian perturbation centered in the middle of the domain, added to a base function $g(x)$. We consider five variants of $g(x)$: zero, linear ($x/2$), quadratic ($(x/2)^2$), a sinusoid ($3\sin(x)$), and a $\tanh$ function ($5\tanh(x/2)$). For each variant, we uniformly draw 50 samples from the interval $[-7.5, -3] \cap [3, 7.5]$. To simulate a realistic out-of-distribution (OOD) scenario, all training samples within a central interval $x \in [-3, 3]$ are removed. Models are trained

TABLE 7. Results of test errors and percentage of violations on real datasets *over the first two input features*. Bold and *gray bold italic* values indicate the best and second-best models, respectively. The proposed method achieves the fewest violations across datasets; RMSE is competitive with, but not uniformly lower than, the best baseline for each dataset.

	RMSE						Violations (%)					
Model	Concrete	Airfoil Self-Noise	Power Plant	Red Wine	White Wine	Real Estate	Concrete	Airfoil Self-Noise	Power Plant	Red Wine	White Wine	Real Estate
MLP	0.79 ± 0.05	0.79 ± 0.03	**0.27 ± 0.01**	0.90 ± 0.02	0.97 ± 0.04	0.95 ± 0.12	-	-	-	-	-	-
KAN	0.84 ± 0.05	0.85 ± 0.07	0.30 ± 0.04	0.93 ± 0.04	0.98 ± 0.05	1.04 ± 0.14	-	-	-	-	-	-
GP	*0.79 ± 0.04*	*0.78 ± 0.06*	**0.27 ± 0.01**	**0.88 ± 0.03**	**0.94 ± 0.05**	*0.90 ± 0.12*	*0.05 ± 0.15*	*0.53 ± 0.73*	*0.87 ± 0.26*	0.69 ± 0.27	*0.60 ± 0.39*	0.84 ± 0.94
DUE	0.98 ± 0.06	0.98 ± 0.05	0.85 ± 0.29	0.96 ± 0.03	0.97 ± 0.05	1.00 ± 0.13	33.16 ± 2.84	32.36 ± 3.28	37.33 ± 4.54	15.69 ± 1.14	27.10 ± 2.57	32.41 ± 5.74
SNGP	0.99 ± 0.07	1.00 ± 0.04	1.00 ± 0.03	0.96 ± 0.03	0.98 ± 0.05	1.00 ± 0.13	76.89 ± 5.28	79.77 ± 6.01	91.78 ± 1.83	94.50 ± 10.82	63.82 ± 15.19	62.77 ± 14.30
DK1	0.80 ± 0.06	0.85 ± 0.03	**0.27 ± 0.01**	0.89 ± 0.03	1.03 ± 0.19	**0.90 ± 0.11**	**0.00 ± 0.00**	1.03 ± 0.69	**0.00 ± 0.00**	*0.38 ± 0.27*	2.12 ± 0.99	*0.24 ± 0.48*
DK2	0.85 ± 0.11	0.86 ± 0.34	0.30 ± 0.05	0.96 ± 0.04	1.00 ± 0.06	1.16 ± 0.11	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
KDK2	**0.78 ± 0.05**	0.82 ± 0.03	**0.27 ± 0.01**	*0.89 ± 0.02*	0.97 ± 0.04	0.98 ± 0.12	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
KDK3	0.90 ± 0.18	0.84 ± 0.04	0.47 ± 0.04	0.89 ± 0.03	*0.95 ± 0.05*	0.94 ± 0.12	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
ENS2	*0.79 ± 0.04*	**0.77 ± 0.04**	*0.29 ± 0.02*	*0.89 ± 0.02*	0.97 ± 0.07	0.97 ± 0.11	39.85 ± 10.02	30.33 ± 7.04	30.87 ± 11.83	42.78 ± 7.58	44.90 ± 7.40	28.43 ± 8.07

TABLE 8. Results of test errors and percentage of violations on real datasets *over all continuous input features*. Bold and *gray bold italic* values indicate the best and second-best models, respectively. The proposed method achieves the fewest violations across datasets; RMSE is competitive with, but not uniformly lower than, the best baseline for each dataset.

	RMSE						Violations (%)					
Model	Concrete	Airfoil Self-Noise	Power Plant	Red Wine	White Wine	Real Estate	Concrete	Airfoil Self-Noise	Power Plant	Red Wine	White Wine	Real Estate
MLP	0.78 ± 0.06	*0.37 ± 0.04*	**0.25 ± 0.01**	1.00 ± 0.10	0.95 ± 0.04	0.85 ± 0.13	-	-	-	-	-	-
KAN	0.94 ± 0.12	0.69 ± 0.10	0.55 ± 0.53	2.06 ± 1.18	3.55 ± 3.20	3.99 ± 5.84	-	-	-	-	-	-
GP	**0.70 ± 0.05**	**0.32 ± 0.03**	**0.25 ± 0.01**	**0.74 ± 0.02**	*0.83 ± 0.07*	**0.55 ± 0.13**	*0.05 ± 0.15*	*1.66 ± 0.76*	*0.68 ± 0.20*	0.91 ± 0.41	*0.70 ± 0.33*	*1.33 ± 1.14*
DUE	0.75 ± 0.05	0.33 ± 0.02	**0.24 ± 0.01**	0.82 ± 0.04	0.77 ± 0.03	0.68 ± 0.14	40.24 ± 5.69	24.49 ± 2.00	31.95 ± 2.39	52.06 ± 2.68	51.28 ± 2.66	46.75 ± 8.85
SNGP	0.98 ± 0.06	1.00 ± 0.04	1.00 ± 0.02	0.96 ± 0.03	0.98 ± 0.05	1.00 ± 0.13	18.30 ± 3.12	64.55 ± 4.61	58.67 ± 3.69	6.34 ± 0.96	18.05 ± 2.41	1.33 ± 0.84
DK1	*0.71 ± 0.04*	0.69 ± 0.03	**0.25 ± 0.01**	*0.79 ± 0.03*	**0.81 ± 0.03**	*0.60 ± 0.11*	*0.05 ± 0.15*	**0.00 ± 0.00**	1.12 ± 0.70	25.00 ± 8.51	10.32 ± 3.14	2.05 ± 1.53
DK2	0.84 ± 0.07	*0.37 ± 0.04*	*0.26 ± 0.02*	0.90 ± 0.03	0.91 ± 0.04	0.87 ± 0.16	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
KDK2	0.79 ± 0.05	0.57 ± 0.03	0.27 ± 0.01	0.88 ± 0.05	0.91 ± 0.04	0.68 ± 0.14	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	*0.75 ± 0.45*	**0.00 ± 0.00**	**0.00 ± 0.00**
KDK3	0.82 ± 0.05	0.81 ± 0.26	0.51 ± 0.02	0.86 ± 0.03	0.92 ± 0.04	0.72 ± 0.14	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
ENS2	0.98 ± 0.24	0.55 ± 0.07	0.34 ± 0.07	2.20 ± 1.33	1.14 ± 0.27	1.21 ± 0.62	20.68 ± 3.81	9.10 ± 3.53	8.10 ± 4.15	11.41 ± 2.05	15.90 ± 2.71	4.22 ± 2.17

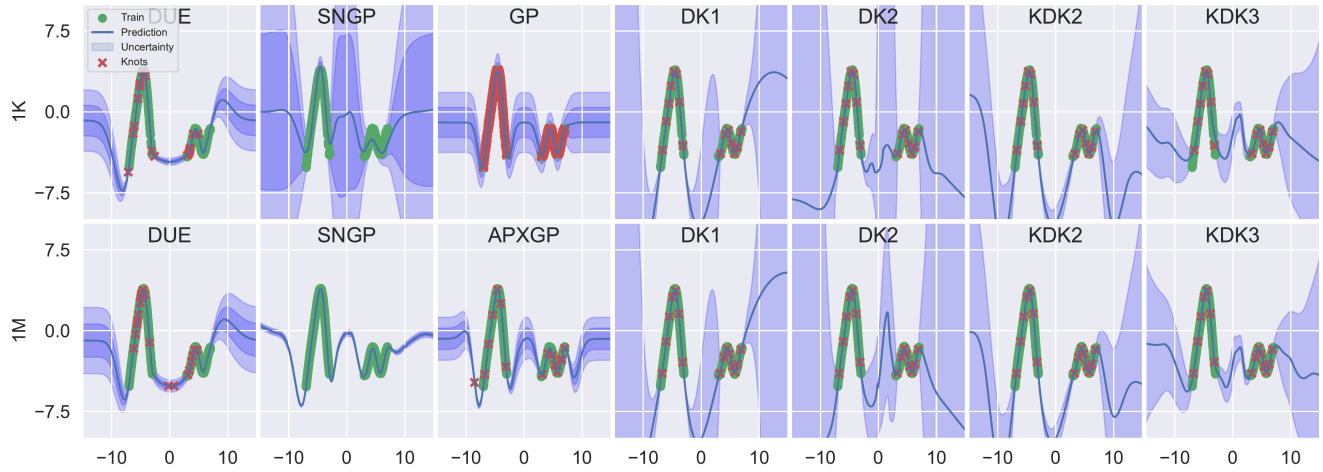

FIGURE 4. Mean of prediction and uncertainty estimates for a 1D regression task, used to evaluate feature collapse and over-generalization across different methods at two training set sizes (1K and 1M samples). Red cross × are projected inducing points in input space for DUE, inducing points for APXGP, and knots for DK1, DK2, KDK2, and KDK3; Note that some are placed in regions with no nearby training samples. GP is replaced by APXGP in the large dataset due to scalability constraints. All the models except SNGP estimate a large uncertainty in the gap between observed regions.

for 1000 epochs outside the gap and evaluated over the full domain, with the evaluation repeated across 10 seeded trials per g(x) variant.

Fig. 5 illustrates predictive means and uncertainty estimates across the entire input domain for the quadratic case ($g(x) = (x/2)^2$). None of the models recover the localized Gaussian perturbation, as expected given the absence of training data in that region, although predictive uncertainty increases within the gap for most models. GP and APXGP expand in the missing data region, whereas DUE and SNGP do not recover comparable expansions; the projections of some of the inducing points for both DUE and APXGP lie in the missing area. In contrast, DAREK and K-DAREK expand significantly in the missing data region, depending on the estimated Lipschitz constant and the distance to the training set.

Table 9 reports gap-region ([-3,3]) violation rate and average uncertainty for GP, SNGP, DUE, and KDK2 across all $g(x)$ variants. KDK2 achieves zero gap-region violations across all variants. SNGP and DUE show substantial violations throughout (54–98%), consistent with the overgeneralization visible in Fig. 5. GP achieves zero violations in four of the five variants, failing only on the tanh case. We note that KDK2's zero-violation coverage comes at the cost

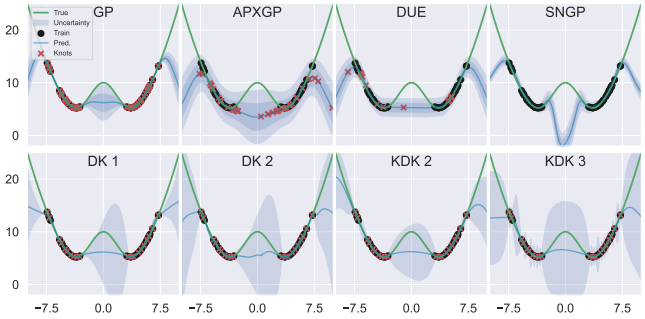

FIGURE 5. **Uncertainty quantification of probabilistic methods versus Lipschitz-based methods in the presence of missing data.**

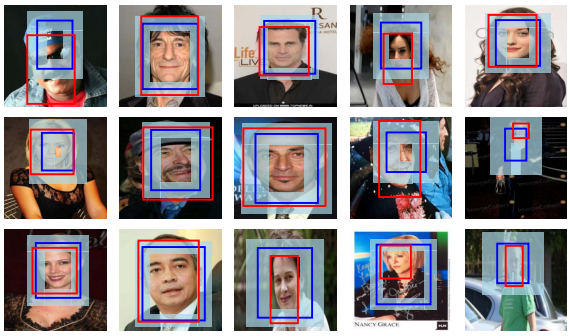

FIGURE 6. **Face bounding-box prediction on sample test images from the CelebA dataset [73] using K-DAREK (KDK2). Red boxes indicate ground-truth bounding boxes and blue boxes indicate K-DAREK predictions. The shaded blue region indicates the error estimates in the x- and y-coordinates. Note that it uses only 100 projected dimensions out of a 25k-dimensional feature space of ResNet-18.**

of a wider average bound than the probabilistic baselines in every variant, reflecting the same tightness-coverage tradeoff observed in the Fixed Budget experiment.

TABLE 9. **Gap-region ($x \in [-3, 3]$) violation rate (%) and average uncertainty width, across five base functions $g(x)$, with the bump term $10\exp(-x^2/8)$ held fixed.**

	Violation (%)				Average uncertainty u_f			
$g(x)$	GP	SNGP	DUE	KDK2	GP	SNGP	DUE	KDK2
zero	0.000	79.100	98.467	**0.000**	0.572	2.881	0.337	7.052
linear	0.000	78.533	97.433	**0.000**	0.614	3.205	0.428	9.172
quadratic	0.000	64.100	54.433	**0.000**	2.453	3.297	2.662	30.704
sine	0.000	72.067	79.900	**0.000**	2.287	3.025	0.697	36.805
tanh	7.500	73.467	74.833	**0.000**	0.646	3.276	1.746	15.277

High dimension distance-awareness. We evaluated K-DAREK on the Celebrity Attribute face detection dataset [73]. The task is to predict the bounding box of the face in the given image. To reduce the dataset size, we randomly selected 8k images for training and 1k images for testing. We reduced feature dimensionality to (2, 5, 10, 20, 50, 100, and 200) using two feature extraction strategies: PCA components of input images and PCA components of ResNet-18 features. The RMSE, IoU, and SDA are reported in Table 10.

For the first approach, we resize the images to a fixed dimension of $224\times224\times3$ and then select the n top PCA components. For the second approach, we resize the images to a fixed dimension of $224\times224\times3$, apply the feature extractor of pretrained ResNet-18 to get $7\times7\times512$ features, and select the n top PCA components. Each of ENS1 and ENS2 has 5 KAN models. The DK1, DK2, KDK2, KDK3, ENS1, and ENS2 models have 10 hidden units and 15 knots for each spline. GP was trained on 15 training points selected using K-means from the training dataset, and DUE and APXGP use 15 inducing points. SNGP uses 1024 random Fourier kernels. All models were trained with a learning rate of 0.1 and decayed by a factor of 0.9 every 200 iterations.

GP was trained for 2000 epochs, DUE for 5000 epochs, SNGP for 10000 epochs, and the remaining models for 1000 iterations. RMSE and IoU remain consistent across models and features, with ResNet-18 features yielding lower RMSE and higher IoU. DAREK's and K-DAREK's SDA (2) increase from approximately 60% at two dimensions to around 95% at 100 dimensions, outperforming ENS1 and ENS2, which do not follow a consistent pattern. We believe the pattern in SDA of DAREK and K-DAREK reflects a curse-of-dimensionality effect: as the input dimension grows, training data becomes sparser relative to the volume of the input space, so nearest-knot distances tend to grow with dimension. We do not have a rigorous account of this effect and leave it to future work. GP-based models achieve near-perfect SDA due to their kernel-based formulation. We evaluated SNGP's SDA metric with respect to training points selected by the K-means approach, since SNGP does not have specific training representatives (such as inducing points or knots). Fig. 6 shows a representative mix of correctly bounded and violated predictions. Violations occur where the numerically estimated Lipschitz constant $\mathcal{L}_f$ underestimates the true local variation of the target function, due to the MLP block's limited capacity on the coarse, low-dimensional feature projection used here, and occasional label noise in the CelebA ground-truth bounding boxes.

Multi-agent safe control: Learning-based control integrates data-driven modeling techniques, such as NNs, 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 [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94].

In this experiment, we evaluated both DAREK and K-DAREK in a multi-agent safe control [95], [96], [97], [98], [99], [100] setup adopted from [70]. Fig. 2 shows a diagram of the training and control loop beside a successful trajectory. The controlled agent (blue car) must safely navigate from its initial position 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

TABLE 10. Results for high-dimensional experiments across two feature extraction methods, PCA and ResNet-18.

Model	RMSE ↓ 2	5	10	20	50	100	200	IoU ↑ 2	5	10	20	50	100	200	SDA ↑ 2	5	10	20	50	100	200
(a) PCA components from images																					
DK1	0.136	0.133	0.133	0.133	0.126	0.129	0.149	0.451	0.456	0.454	0.452	0.469	0.466	0.410	0.676	0.856	0.947	0.967	0.974	0.952	0.969
DK2	0.137	0.139	0.134	0.190	0.123	0.184	0.135	0.448	0.443	0.454	0.375	0.486	0.339	0.451	0.677	0.857	0.947	0.972	0.974	0.950	0.971
KDK2	0.137	0.133	0.130	0.130	0.134	0.135	0.136	0.439	0.450	0.456	0.464	0.444	0.446	0.440	0.726	0.843	0.918	0.929	0.971	0.945	0.978
KDK3	0.137	0.131	0.130	0.128	0.130	0.132	0.133	0.447	0.459	0.464	0.470	0.462	0.456	0.454	0.692	0.741	0.862	0.919	0.917	0.911	0.932
ENS1	0.136	0.133	0.133	0.133	0.126	0.130	0.151	0.450	0.457	0.455	0.451	0.467	0.464	0.407	0.571	0.554	0.568	0.563	0.501	0.431	0.378
ENS2	0.135	0.131	0.131	0.128	0.183	0.265	0.129	0.449	0.456	0.446	0.469	0.311	0.145	0.431	0.527	0.542	0.545	0.530	0.588	0.564	0.525
GP	0.138	0.149	0.137	0.161	0.138	0.137	0.143	0.435	0.400	0.450	0.359	0.437	0.448	0.420	1.000	1.000	1.000	1.000	1.000	1.000	1.000
APXGP	0.138	0.137	0.137	0.137	0.137	0.137	0.137	0.427	0.447	0.447	0.447	0.447	0.447	0.447	0.977	1.000	1.000	1.000	1.000	1.000	1.000
DUE	0.135	0.130	0.130	0.119	0.103	0.104	0.112	0.450	0.470	0.480	0.523	0.554	0.556	0.534	0.682	0.930	0.993	1.000	1.000	1.000	1.000
SNGP	0.138	0.144	0.164	0.155	0.148	0.198	0.207	0.445	0.443	0.421	0.424	0.427	0.244	0.246	0.628	0.840	0.892	0.902	0.867	0.553	0.593
(b) PCA components from ResNet-18 features																					
DK1	0.100	0.097	0.090	0.088	0.079	0.074	0.079	0.535	0.538	0.562	0.557	0.586	0.606	0.589	0.704	0.686	0.875	0.892	0.788	0.837	0.935
DK2	0.100	0.100	0.093	0.103	0.083	0.073	0.072	0.533	0.535	0.553	0.470	0.573	0.608	0.626	0.691	0.707	0.899	0.918	0.858	0.882	0.902
KDK2	0.100	0.095	0.086	0.086	0.077	0.089	0.071	0.536	0.548	0.575	0.575	0.602	0.493	0.640	0.700	0.734	0.819	0.869	0.849	0.860	0.904
KDK3	0.101	0.095	0.086	0.087	0.084	0.090	0.130	0.531	0.546	0.574	0.567	0.553	0.518	0.381	0.690	0.724	0.791	0.830	0.849	0.897	0.908
ENS1	0.100	0.097	0.089	0.089	0.080	0.075	0.081	0.535	0.540	0.565	0.557	0.585	0.603	0.583	0.578	0.551	0.566	0.562	0.364	0.222	0.408
ENS2	0.100	0.094	0.087	0.084	0.174	0.069	0.065	0.530	0.551	0.569	0.575	0.161	0.628	0.646	0.515	0.533	0.573	0.562	0.461	0.550	0.542
GP	0.139	0.140	0.139	0.140	0.140	0.137	0.138	0.431	0.447	0.434	0.427	0.426	0.447	0.444	0.542	1.000	1.000	1.000	1.000	1.000	1.000
APXGP	0.133	0.137	0.137	0.137	0.137	0.137	0.137	0.445	0.447	0.447	0.447	0.447	0.447	0.447	0.729	1.000	1.000	1.000	1.000	1.000	1.000
DUE	0.099	0.095	0.087	0.085	0.076	0.072	0.068	0.537	0.548	0.581	0.575	0.608	0.637	0.650	0.722	0.937	0.990	0.997	1.000	1.000	1.000
SNGP	0.102	0.095	0.107	0.166	0.176	0.204	0.245	0.528	0.547	0.525	0.341	0.305	0.229	0.138	0.591	0.869	0.832	0.544	0.560	0.497	0.512

agents. Its dynamics are discrete control-affine, given by:

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) + \mathbf{g}(\mathbf{x}_t)\mathbf{a}_t + \boldsymbol{\delta}(\mathbf{x}_t), \tag{26}$$

where state $\mathbf{x} \in \mathbb{R}^4$ contains position and velocity in x and y directions. Here, $\mathbf{f}$ denotes the nominal drift dynamics, $\mathbf{g}$ is the input gain matrix, $\mathbf{a}$ is the control input (accelerations in x and y directions), and $\boldsymbol{\delta}$ represents unknown uncertainty in the dynamics. The control policy first computes an optimal but potentially unsafe trajectory using an MPC controller that ignores obstacles. Then, following [70, eq. 17], a 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 unknown uncertainty $\boldsymbol{\delta}$ is bounded inside a polytope. We use the worst-case error bounds predicted by DK2, KDK2, and KDK3 to define this polytope. Models use 5 hidden units. 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 models are trained offline for 300 epochs on 5000 noise-free data points collected. We add uniform noise at three levels (5%, 10%, and 15%) independently to each of the position, velocity, and acceleration, denoted by $\boldsymbol{\delta}_p$, $\boldsymbol{\delta}_v$, and $\boldsymbol{\delta}_a$. Considering all permutations of these noise levels results in 27 experimental setups, each repeated 50 times. The dynamics in Equation 26 update position and velocity as $\mathbf{p}_{t+1} = \mathbf{p}_t + \mathbf{v}_t dt + \delta_p$ and $\mathbf{v}_{t+1} = \mathbf{v}_t - k_v|\mathbf{v}_t|^2 + k_d(|\mathbf{v}_t| + 1)dt(\mathbf{a} + \delta_a) + \delta_v$, where k_d is the acceleration scaling factor. The unknown part of the system is $\tilde{\mathbf{p}}_{t+1} = \delta_p$ and $\tilde{\mathbf{v}}_{t+1} = k_d(|\mathbf{v}_t| + 1)dt\delta_a + \delta_v$. The Lipschitz constant derivation for this unknown dynamics model with respect to velocity gives $\mathcal{L}_{\partial\tilde{p}_{t+1}/\partial v} = 0$, and differentiating the velocity update gives $\mathcal{L}_{\partial\tilde{v}_{t+1}/\partial v} = k_d\delta_a dt$.

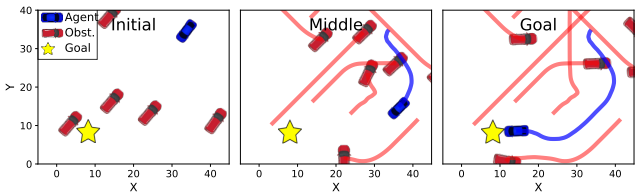

FIGURE 7. Three stages (Initial, Middle, Goal) of a successful trajectory using K-DAREK bounds, where the controlled agent (blue car) safely navigates to the goal (star) while avoiding obstacle agents (red cars).

We use $k_d = 1.0$, and $dt = 0.1$ for the simulation step time. The position and velocity disturbances, δ_p and δ_v, are not captured by this Lipschitz constant; they are instead captured through the error-at-knots term (Sec. V-B-3), since they enter the dynamics additively rather than through a term whose sensitivity the Lipschitz constant measures. Detailed results of this simulation are reported in Table 11.

The *Average* column reports the mean over all 27 setups corresponding to a specific value of $\bar{\boldsymbol{\delta}}_a$. The other three columns report the results for individual setups where $\bar{\boldsymbol{\delta}}_v = \bar{\boldsymbol{\delta}}_p \in$ (5%, 10%, and 15%), with $\bar{\boldsymbol{\delta}}_a$ varying across rows. On average, K-DAREK achieves a slightly higher success rate along with lower collision and stuck rates. A realization of safe navigation with K-DAREK is illustrated in Fig. 7.

VII. CONCLUSION

We presented a general framework for quantifying distance-aware worst-case error in a hybrid model that combines deep dense NNs and spline-based networks, namely K-DAREK. We derived an explicit error bound for the MLP block using its Lipschitz constant, thereby linking test point uncertainty to proximity to the 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

TABLE 11. Results for the multi-agent safe control experiment, where $\bar{\delta}_a$, $\bar{\delta}_v$, and $\bar{\delta}_p$ are uniform disturbance rates on acceleration, velocity, and position. DK2 refers to two-layer DAREK, and KDK2 and KDK3 are two-layer and three-layer K-DAREK, respectively, as described in the text. The KDK2 and KDK3 achieve a lower collision rate and slightly higher success rate than the DK2 model.

		$\bar{\delta}_v = \bar{\delta}_p = 0.05$			$\bar{\delta}_v = \bar{\delta}_p = 0.10$			$\bar{\delta}_v = \bar{\delta}_p = 0.15$			Average		
$\bar{\delta}_a$	Model	DK2	KDK2	KDK3	DK2	KDK2	KDK3	DK2	KDK2	KDK3	DK2	KDK2	KDK3
0.05	Success	0.90	0.92	0.92	0.96	0.98	0.96	0.96	0.96	0.94	0.953	0.953	0.951
	Collision	0.08	0.04	0.06	0.02	0.00	0.02	0.00	0.00	0.00	0.018	0.011	0.015
	Stuck	0.02	0.04	0.02	0.02	0.02	0.02	0.04	0.04	0.06	0.028	0.035	0.033
0.10	Success	0.96	0.98	0.96	0.96	0.94	0.98	0.92	0.94	0.94	0.953	0.957	0.964
	Collision	0.02	0.00	0.02	0.00	0.02	0.00	0.00	0.00	0.00	0.011	0.006	0.011
	Stuck	0.02	0.02	0.02	0.04	0.04	0.02	0.08	0.06	0.06	0.028	0.033	0.033
0.15	Success	0.92	0.94	0.98	0.96	0.96	0.98	0.96	0.96	0.92	0.947	0.955	0.955
	Collision	0.02	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.011	0.004	0.008
	Stuck	0.06	0.04	0.02	0.04	0.04	0.02	0.04	0.04	0.08	0.031	0.035	0.033

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 yields an error bound that is distance-aware and does not require sampling, ensembling, or distributional assumptions about the data, and is based on Lipschitz continuity. We further evaluated whether the component-wise construction of K-DAREK's bound yields joint distance-awareness in higher dimensions through the SDA metric on the face-detection task, where K-DAREK reaches SDA values of approximately 95%. Also, on real data regression tasks, K-DAREK achieves zero or near-zero coverage violations across most datasets, while achieving competitive accuracy compared to baselines. Our model avoids over-generalization (feature collapse) observed in SNGP, and its behavior is consistent in low and high amounts of data. Additionally, in multi-agent safe control tasks, K-DAREK improves success rates and reduces collisions. We also conduct an ablation study to separately quantify the effects of the MLP block, spectral normalization, and knot-selection strategy, and spline capacity on accuracy, empirical coverage, and average bound width. We additionally quantify how violation rate degrades as the assumed Lipschitz constant is underestimated, showing that the guarantee's reliability depends directly on the accuracy of this estimate, consistent with the conditional nature of the guarantee discussed in Sec. V.

This framework provides a theoretically grounded understanding of the reliability of prediction and error estimation, as well as computational efficiency, paving the way for robust learning in settings that demand deterministic, worst-case error bounds with an assumed Lipschitz constant or risk-aware predictions. However, several limitations of the current framework suggest directions for future work. First, the analysis assumes that the target function is continuous and Lipschitz, and it does not directly apply to predictors with discontinuous or Boolean outputs. Second, spectral normalization controls the Lipschitz constant of each MLP block, but the spline block is not explicitly constrained in the same way. Moreover, the current formulation divides the Lipschitz budget uniformly among splines within a group (Remark 5). A non-uniform allocation that accounts for each spline's local curvature could yield a tighter overall bound. Third, knots are selected from the training data, which makes the hybrid network data-adaptive and ties the resulting error bound to observed samples. We compare several alternative knot-selection strategies in Table 5; a more exhaustive comparison across additional tasks is left for future work (Remark 3). Fourth, the theoretical guarantee relies on knowledge of the target function's Lipschitz constant. In practice, this constant may only be approximated (as in our simulations); our sensitivity analysis (Sec. VI) quantifies how the violation rate degrades under such approximation, and accounting for this approximation more formally in the error bound is left for future work. Fifth, restricting the Chebyshev expansion of the MLP block in KKAN architecture to its first two degrees simplifies the error analysis at the cost of expressiveness; a higher-order expansion with a corresponding error analysis is left for future work. Finally, the present work focuses on worst-case analysis; developing hybrid worst-case/probabilistic methods to tighten the bounds in data-rich scenarios is left for future research.

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] R. Balestriero *et al.*, "A spline theory of deep learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 374–383.

[8] G. Wahba, "Smoothing noisy data with spline functions," *Numerische mathematik*, vol. 24, no. 5, pp. 383–393, 1975.

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

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

[11] S. Aziznejad, H. Gupta, J. Campos, and M. Unser, "Deep neural networks with trainable activations and controlled lipschitz constant,"

IEEE Transactions on Signal Processing, vol. 68, pp. 4688–4699, 2020.
[12] Z. Wang, R. Romagnoli, S. Mowlavi, and Y. Nakahira, "Physics-informed deep b-spline networks for dynamical systems," *Journal of machine learning research*, 2026.
[13] 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.
[14] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. The MIT Press, 2006.
[15] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016, pp. 1050–1059.
[16] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *ICML*, 2015, pp. 1613–1622.
[17] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *NeurIPS*, vol. 30, 2017.
[18] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *ICML*, 2017, pp. 1321–1330.
[19] 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.
[20] 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.
[21] M. Ataei and V. Dhiman, "DADEE: Well-calibrated uncertainty quantification in neural networks for barriers-based robot safety," *arXiv:2407.00616*, 2024.
[22] L. Jaulin, M. Kieffer, O. Didrit, E. Walter, L. Jaulin, M. Kieffer, O. Didrit, and É. Walter, *Interval analysis*. Springer, 2001.
[23] M. Ataei, M. J. Khojasteh, and V. Dhiman, "DAREK-Distance aware error for Kolmogorov networks," in *IEEE ICASSP*, 2025, pp. 1–5.
[24] J. A. Lopez and V. Kreinovich, "Towards decision making under interval uncertainty," in *NAFIPS*. Springer, 2023, pp. 335–337.
[25] D. Bertsimas, X. Boix, K. V. Carballo, and D. d. Hertog, "Robust upper bounds for adversarial training," *arXiv preprint arXiv:2112.09279*, 2021.
[26] D. Bertsimas, J. Dunn, C. Pawlowski, and Y. D. Zhuo, "Robust classification," *INFORMS Journal on Optimization*, vol. 1, no. 1, pp. 2–34, 2019.
[27] 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.
[28] G. N. Nair, "A nonstochastic information theory for communication and state estimation," *IEEE Transactions on automatic control*, vol. 58, no. 6, pp. 1497–1510, 2013.
[29] S. Prajna, A. Jadbabaie, and G. J. Pappas, "A framework for worst-case and stochastic safety verification using barrier certificates," *IEEE Transactions on Automatic Control*, vol. 52, no. 8, pp. 1415–1428, 2007.
[30] 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.
[31] 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.
[32] 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.
[33] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
[34] C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus, "Kolmogorov-arnold networks (kans) for time series analysis," in *2024 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2024, pp. 1–6.
[35] S. Patra, S. Panda, B. K. Parida, M. Arya, K. Jacobs, D. I. Bondar, and A. Sen, "Physics informed kolmogorov-arnold neural networks for dynamical analysis via efficient-kan and wav-kan," *Journal of Machine Learning Research*, vol. 26, no. 233, pp. 1–39, 2025.
[36] A. Sen, I. V. Lukin, K. Jacobs, L. Kaplan, A. G. Sotnikov, and D. I. Bondar, "Physics-informed time series analysis with kolmogorov-arnold networks under ehrenfest constraints," *Physical Review Research*, vol. 8, no. 2, p. 023018, 2026.
[37] L. Li, Y. Zhang, G. Wang, and K. Xia, "Kolmogorov–arnold graph neural networks for molecular property prediction," *Nature Machine Intelligence*, vol. 7, no. 8, pp. 1346–1354, 2025.
[38] A. A. Howard, N. Zolman, B. Jacob, S. L. Brunton, and P. Stinis, "Sindy-kans: Sparse identification of non-linear dynamics through kolmogorov-arnold networks," *arXiv preprint arXiv:2603.18548*, 2026.
[39] J. Schmidt-Hieber, "The Kolmogorov–Arnold representation theorem revisited," *Neural networks*, vol. 137, pp. 119–126, 2021.
[40] H. Montanelli and H. Yang, "Error bounds for deep relu networks using the kolmogorov–arnold superposition theorem," *Neural Networks*, vol. 129, pp. 1–6, 2020.
[41] 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.
[42] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *arXiv:1802.05957*, 2018.
[43] J. Van Amersfoort, L. Smith, A. Jesson, O. Key, and Y. Gal, "On feature collapse and deep kernel learning for single forward pass uncertainty," *arXiv:2102.11409*, 2021.
[44] M. Ataei, V. Dhiman, and M. J. Khojasteh, "K-darek: Distance aware error for kurkova kolmogorov networks," in *2025 59th Asilomar Conference on Signals, Systems, and Computers*, 2025, pp. 1336–1342.
[45] 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.
[46] 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," *AI Review*, pp. 1–77, 2023.
[47] S. Morris, "Hilbert 13: Are there any genuine continuous multivariate real-valued functions?" *Bull. AMS*, vol. 58, no. 1, pp. 107–118, 2021.
[48] F. Girosi and T. Poggio, "Representation properties of networks: Kolmogorov's theorem is irrelevant," *Neural Computation*, vol. 1, no. 4, pp. 465–469, 1989.
[49] V. Kůrkovà, "Kolmogorov's theorem is relevant," *Neural computation*, vol. 3, no. 4, pp. 617–622, 1991.
[50] V. Kůrkovà, "Kolmogorov's theorem and multilayer neural networks," *Neural networks*, vol. 5, no. 3, pp. 501–506, 1992.
[51] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *AISTATS*, 2016, pp. 370–378.
[52] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *IJCV*, vol. 132, no. 12, pp. 5635–5662, 2024.
[53] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv:1702.08608*, 2017.
[54] 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.
[55] R. Marcinkevičs and J. E. Vogt, "Interpretability and explainability: a machine learning zoo mini-tour. 2020, p," *arXiv:2012.01805*, 2012.
[56] C. Rudin, C. Chen, Z. Chen, H. Huang, L. Semenova, and C. Zhong, "Interpretable machine learning: Fundamental principles and 10 grand challenges," *Statistic Surveys*, vol. 16, pp. 1–85, 2022.
[57] N. Nanda, L. Chan, T. Lieberum, J. Smith, and J. Steinhardt, "Progress measures for grokking via mechanistic interpretability," *arXiv:2301.05217*, 2023.
[58] N. Makke and S. Chawla, "Interpretable scientific discovery with symbolic regression: a review," *AI Review*, vol. 57, no. 1, p. 2, 2024.
[59] T. Tohme, M. J. Khojasteh, M. Sadr, F. Meyer, and K. Youcef-Toumi, "ISR: Invertible symbolic regression," *arXiv:2405.06848*, 2024.
[60] K. Kacprzyk and M. van der Schaar, "Beyond size-based metrics: Measuring task-specific complexity in symbolic regression," in *The 28th International Conference on Artificial Intelligence and Statistics*, 2025.
[61] K. Xu, H. Zhang, S. Wang, Y. Wang, S. Jana, X. Lin, and C.-J. Hsieh, "Fast and complete: Enabling complete neural network verification with rapid and massively parallel incomplete verifiers," *arXiv preprint arXiv:2011.13824*, 2020.
[62] C. Liu, T. Arnon, C. Lazarus, C. Strong, C. Barrett, and M. J. Kochenderfer, "Algorithms for verifying deep neural networks," *Foundations and Trends in Optimization*, vol. 4, no. 3-4, pp. 244–404, 2021.

[63] H. Zhang, T.-W. Weng, P.-Y. Chen, C.-J. Hsieh, and L. Daniel, "Efficient neural network robustness certification with general activation functions," *Advances in neural information processing systems*, vol. 31, 2018.

[64] K. Xu, Z. Shi, H. Zhang, Y. Wang, K.-W. Chang, M. Huang, B. Kailkhura, X. Lin, and C.-J. Hsieh, "Automatic perturbation analysis for scalable certified robustness and beyond," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1129–1141, 2020.

[65] Z. Shi, H. Li, C.-J. Hsieh, and H. Zhang, "Certified training with branch-and-bound for lyapunov-stable neural control," *arXiv preprint arXiv:2411.18235*, 2024.

[66] M. Ataei, M. J. Khojasteh, and V. Dhiman, "Distance-aware error for Spline networks: A bottom-up approach to uncertainty," *arXiv:2501.04757v2*, 2026.

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

[68] D. Bertsimas, D. Den Hertog, and J. Pauphilet, "Probabilistic guarantees in robust optimization," *SIAM Journal on Optimization*, vol. 31, no. 4, pp. 2893–2920, 2021.

[69] I. Wang, B. Van Parys, and B. Stellato, "Learning decision-focused uncertainty sets in robust optimization," *arXiv preprint arXiv:2305.19225*, 2023.

[70] 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.

[71] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix gaussian posteriors," in *International conference on machine learning*. PMLR, 2016, pp. 1708–1716.

[72] A. Frank, "UCI machine learning repository," 2010.

[73] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[74] J. Hensman, A. Matthews, and Z. Ghahramani, "Scalable variational gaussian process classification," in *Artificial intelligence and statistics*, 2015, pp. 351–360.

[75] R. C. De Amorim and B. Mirkin, "Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering," *Pattern Recognition*, vol. 45, no. 3, pp. 1061–1075, 2012.

[76] G. Wahba and Y. Wang, "Spline function: Overview," *Wahba, G. and Wang, Y.(2014)'Spline Function', Encyclopedia of Statistical Sciences*, pp. 1–27, 1990.

[77] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[78] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[79] 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.

[80] 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.

[81] A. Lederer, J. Umlauft, and S. Hirche, "Uniform error bounds for gaussian process regression with application to safe control," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[82] A. Capone, G. Noske, J. Umlauft, T. Beckers, A. Lederer, and S. Hirche, "Localized active learning of gaussian process state space models," in *Learning for Dynamics and Control*. PMLR, 2020, pp. 490–499.

[83] K. Long, Y. Yi, Z. Dai, S. Herbert, J. Cortés, and N. Atanasov, "Sensor-based distributionally robust control for safe robot navigation in dynamic environments," *The International Journal of Robotics Research*, vol. 45, no. 2, pp. 328–351, 2026.

[84] C. Wang, Y. Meng, Y. Li, S. L. Smith, and J. Liu, "Learning control barrier functions with high relative degree for safety-critical control," in *2021 European Control Conference (ECC)*. IEEE, 2021, pp. 1459–1464.

[85] L. Lindemann, H. Hu, A. Robey, H. Zhang, D. Dimarogonas, S. Tu, and N. Matni, "Learning hybrid control barrier functions from data," in *Conference on robot learning*. PMLR, 2021, pp. 1351–1370.

[86] D. D. Fan, J. Nguyen, R. Thakker, N. Alatur, A.-a. Aghamohammadi, and E. A. Theodorou, "Bayesian learning-based adaptive control for safety critical systems," in *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 4093–4099.

[87] A. Gahlawat, P. Zhao, A. Patterson, N. Hovakimyan, and E. Theodorou, "L1-gp: L1 adaptive control with bayesian learning," in *Learning for dynamics and control*. PMLR, 2020, pp. 826–837.

[88] Z. Marvi and B. Kiumarsi, "Safe off-policy reinforcement learning using barrier functions," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 2176–2181.

[89] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7139–7145.

[90] J. Kim, J. Shin, and I. Yang, "Hamilton-jacobi deep q-learning for deterministic continuous-time systems with lipschitz continuous controls," *Journal of Machine Learning Research*, vol. 22, no. 206, pp. 1–34, 2021.

[91] T. Sarkar, A. Rakhlin, and M. A. Dahleh, "Finite time lti system identification," *Journal of Machine Learning Research*, vol. 22, no. 26, pp. 1–61, 2021.

[92] S. Fattahi, N. Matni, and S. Sojoudi, "Learning sparse dynamical systems from a single sample trajectory," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2682–2689.

[93] M. Buisson-Fenet, F. Solowjow, and S. Trimpe, "Actively learning gaussian process dynamics," in *Learning for dynamics and control*. PMLR, 2020, pp. 5–15.

[94] N. Rahimi, S. Talebi, A. Deole, M. Mesbahi, S. Bandyopadhyay, and A. Rahmani, "Robust controller synthesis for vision-based spacecraft guidance and control," in *AIAA SCITECH 2022 Forum*, 2022, p. 2213.

[95] S. Zhang, O. So, K. Garg, and C. Fan, "Gcbf+: A neural graph control barrier function framework for distributed safe multiagent control," *IEEE Transactions on Robotics*, vol. 41, pp. 1533–1552, 2025.

[96] A. Prorok, M. Malencia, L. Carlone, G. S. Sukhatme, B. M. Sadler, and V. Kumar, "Beyond robustness: A taxonomy of approaches towards resilient multi-robot systems," *arXiv preprint arXiv:2109.12343*, 2021.

[97] U. Borrmann, L. Wang, A. D. Ames, and M. Egerstedt, "Control barrier certificates for safe swarm behavior," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 68–73, 2015.

[98] T. Beckers, S. Hirche, and L. Colombo, "Online learning-based formation control of multi-agent systems with gaussian processes," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021, pp. 2197–2202.

[99] H. Jing and Y. Nakahira, "Probabilistic safety and performance certificate for multi-agent safe control and learning," *IEEE Transactions on Control of Network Systems*, 2026.

[100] A. Navsalkar and A. R. Hota, "Data-driven risk-sensitive model predictive control for safe navigation in multi-robot systems," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 1442–1448.