MonoKAN: Certified Monotonic Kolmogorov-Arnold Network

Alejandro Polo-Molina[a,*], David Alfaya[a,b] and Jose Portela[a,c]

[a]Institute for Research in Technology (IIT), ICAI School of Engineering, Universidad Pontificia Comillas, C/del Rey Francisco 4, Madrid, 28008, Madrid, Spain

[b]Department of Applied Mathematics, ICAI School of Engineering, Universidad Pontificia Comillas, C/Alberto Aguilera 25, Madrid, 28015, Madrid, Spain

[c]Department of Quantitative Methods, ICADE, Universidad Pontificia Comillas, C/Alberto Aguilera 23, Madrid, 28015, Madrid, Spain

ABSTRACT

Artificial Neural Networks (ANNs) have significantly advanced various fields by effectively recognizing patterns and solving complex problems. Despite these advancements, their interpretability remains a critical challenge, especially in applications where transparency and accountability are essential. To address this, explainable AI (XAI) has made progress in demystifying ANNs, yet interpretability alone is often insufficient. In certain applications, model predictions must align with expert-imposed requirements, sometimes exemplified by partial monotonicity constraints. While monotonic approaches are found in the literature for traditional Multi-layer Perceptrons (MLPs), they still face difficulties in achieving both interpretability and certified partial monotonicity. Recently, the Kolmogorov-Arnold Network (KAN) architecture, based on learnable activation functions parametrized as splines, has been proposed as a more interpretable alternative to MLPs. Building on this, we introduce a novel ANN architecture called MonoKAN, which is based on the KAN architecture and achieves certified partial monotonicity while enhancing interpretability. To achieve this, we employ cubic Hermite splines, which guarantee monotonicity through a set of straightforward conditions. Additionally, by using positive weights in the linear combinations of these splines, we ensure that the network preserves the monotonic relationships between input and output. Our experiments demonstrate that MonoKAN not only enhances interpretability but also improves predictive performance across the majority of benchmarks, outperforming state-of-the-art monotonic MLP approaches.

1. Introduction

Artificial neural networks (ANNs) are the backbone of modern artificial intelligence (Lecun et al., 2015; Goodfellow et al., 2016). These computational systems are designed to recognize patterns and solve complex problems through learning from data, making them highly effective for tasks such as image and speech recognition (Hinton et al., 2012), predictive analytics (Liu et al., 2017) or many others (Sarvamangala and Kulkarni, 2022; Xu et al., 2020). By mimicking the brain's ability to process information and adapt through experience, ANNs have revolutionized fields ranging from computer vision to autonomous systems (Sarvamangala and Kulkarni, 2022; Voulodimos et al., 2018), and their development continues to drive forward the capabilities of machine learning and artificial intelligence as a whole.

Despite their impressive capabilities, ANNs face significant challenges regarding interpretability. As ANNs grow more complex, their decision-making processes become increasingly opaque, often described as "black boxes" due to the difficulty in understanding how specific inputs are translated into outputs. This lack of transparency can be problematic in critical applications such as healthcare or finance, where understanding the rationale behind decisions is crucial for trust and accountability (Cohen et al., 2021; Tjoa and Guan, 2021). Furthermore, the complexity of ANNs makes it hard to find and fix biases in the models, which can lead to unfair or harmful results. Addressing these interpretability issues is essential to ensure that ANNs can be safely and effectively integrated into high-stakes environments.

In response to the interpretability challenges of ANNs, the field of explainable artificial intelligence (XAI) has grown substantially in the last decades. Numerous studies have emerged aiming to demystify their inner workings (Zhang et al., 2020; Pizarroso et al., 2022; Morala et al., 2023). These studies represent critical strides toward making neural networks more transparent and trustworthy, facilitating their adoption in fields where understanding and accountability are paramount.

However, it is often the case where interpretability alone is insufficient in some critical applications (Rudin, 2019). Therefore, in some fields, it is a requisite to certify that the model predictions align with some requirements imposed by human experts (Cohen et al., 2021). Partial monotonicity is an example where incorporating prior knowledge from human experts into the model might sometimes be necessary. For instance, in university admissions, it is reasonable to expect that, all other variables being equal, an applicant with a higher GPA should have a higher probability of being accepted. If the model's predictions do not follow this monotonic relationship, it could lead to unfair and unethical

*Corresponding author

apolo@comillas.edu (A. Polo-Molina); dalfaya@comillas.edu (D. Alfaya); jportela@comillas.edu (J. Portela)

ORCID(s): 0000-0001-7051-2288 (A. Polo-Molina); 0000-0002-4247-1498 (D. Alfaya); 0000-0002-7839-8982 (J. Portela)

admission decisions. For instance, an applicant with a 4.0 GPA being rejected while an applicant with a 3.0 GPA is accepted, all other factors being equal, would be seen as unfair and could indicate bias in the model.

Partial monotonicity also plays a key role in domains such as healthcare, finance or criminal justice (Wang and Gupta, 2020). In organ transplant allocation, for example, policies often prioritize sicker patients with higher urgency scores to receive transplants sooner (Iserson and Moskop, 2007), as is the case with the United States Transplant Board (Bernstein, 2017). A model violating this principle, by assigning lower transplant priority to a patient with higher medical need, could result in unethical outcomes. Besides, in criminal sentencing, many legal systems impose stricter penalties on repeat offenders compared to first-time offenders (Allen et al., 2016). A model used for risk assessment that fails to respect this monotonic relationship could produce recommendations that are legally and socially unacceptable. Consequently, certified monotonic ML models allow users to enforce such domain-specific monotonic constraints during training, helping ensure that the model adheres to expected input-output relations.

As a result, the training of partial monotonic ANNs has become a prominent area of research in recent years. To tackle this issue, two primary strategies have emerged (Liu et al., 2020). First of all, there are some studies that enforce monotonicity by means of a regularization term that guides the ANNs training towards a partial monotonic solution (Sivaraman et al., 2020; Gupta et al., 2019; Monteiro et al., 2022). However, these approaches verify monotonicity only on a finite set of points, and hence none of the previous studies can certify the enforcement of partial monotonic constraints across all possible input values. Therefore, it is necessary to use some external certification algorithm after the training process to guarantee partial monotonicity. Regarding this type of algorithm, few examples are found in the literature (Liu et al., 2020; Polo-Molina et al., 2025). On the other hand, some studies propose designing constrained architectures that inherently ensure monotonicity (Runje and Shankaranarayana, 2023; Daniels and Velikova, 2010; You et al., 2017; Nolte et al., 2022). Although these methods can guarantee partial monotonicity, they often come with the trade-off of being overly restrictive or complex and challenging to implement (Liu et al., 2020).

Even though some of the aforementioned methods can lead up to certified partial monotonic ANNs, traditional Multi-layer Perceptron (MLP) architectures still have significant difficulties with interpretability. The complex and often opaque nature of the connections and weight adjustments in MLPs makes it challenging to understand and predict how inputs are being transformed into outputs. Therefore, existing approaches to obtaining monotonic MLPs hardly generate both interpretable and certified partial monotonic ANNs, often requiring post-hoc interpretability methods.

To address some of the aforementioned difficulties related to interpretability, a new ANN architecture, called the Kolmogorov-Arnold Network (KAN), has been recently proposed (Liu et al., 2024b). Unlike traditional MLPs, which rely on the universal approximation theorem, KANs leverage the Kolmogorov-Arnold representation theorem. This theorem states that any multivariate continuous function can be decomposed into a finite combination of univariate functions, enhancing the interpretability of the network.

However, the functions depicted by the Kolmogorov-Arnold theorem can be non-smooth, even fractal, and may not be learnable in practice (Liu et al., 2024b). Consequently, a KAN with the width and depth proposed by the Kolmogorov-Arnold theorem is often too simplistic in practice to approximate any function arbitrarily well using smooth splines.

Therefore, although the use of the Kolmogorov-Arnold representation theorem for ANNs was already studied (Sprecher and Draghici, 2002; Köppen, 2002), the major breakthrough occurred when Liu et al. (2024b) established the analogy between MLPs and KANs. In MLPs, the notion of a layer is clearly defined, and the model's power comes from stacking multiple layers to form deeper architectures. Similarly, defining a KAN layer allows for the creation of deep KANs through layer stacking, which significantly enhances the model's ability to capture increasingly complex functions.

Schematically, each KAN layer is composed of a set of nodes, with each node connected to all preceding nodes via activation functions on the edges. These univariate activation functions are the components subjected to training. Then, the outputs of these activation functions are aggregated to determine the node's output. According to (Liu et al., 2024b), this approach not only enhances interpretability by allowing visualization of relationships between variables, but also demonstrates faster neural scaling laws compared to MLPs due to its ability to decompose complex functions into simpler ones. Additionally, it can improve performance on numerous problems compared to MLPs (Poeta et al., 2024; Xu et al., 2024).

Building on these advantages, this paper proposes a novel KAN architecture called MonoKAN that forces the resulting KAN to be certified partially monotonic across the entire input space. To do so, while the original formulation of KANs proposes the use of B-splines, this paper replaces them with cubic Hermite splines and imposes constraints on their coefficients to ensure monotonicity. This substitution allows for more flexible and general imposition of monotonic conditions. An intuitive rationale for this change is that, while it is possible to achieve monotonicity with a combination of B-splines within a specific interval, B-splines are not inherently monotonic. In contrast, cubic Hermite splines can be imposed to be monotonic naturally (Fritsch and Carlson, 1980; Aràndiga et al., 2022), making them a more appropriate choice for ensuring the desired monotonic properties in the MonoKAN architecture.

Therefore, MonoKAN enhances the capability of the KAN framework by leveraging the intrinsic properties of cubic Hermite splines to achieve certified partial monotonicity. Consequently, the proposed MonoKAN architecture is able to encompass both the enhanced interpretability that the

KAN architecture presents with certified partial monotonicity. To the authors' knowledge, this is the first time that a monotonic approach for a KAN has been proposed.

The paper is structured as follows: Section 2 presents the KAN methodology that will be later used in Section 3 as the base to generate certified partial monotonic KANs. Besides, in Section 4, the experiments and corresponding results are detailed, demonstrating that the proposed approach surpasses the state-of-the-art methods in the majority of the experiments. Moreover, in Section 5, the computational complexity of the proposed method and the comparison with the state-of-the-art models is presented. Lastly, the main contributions are summarized in Section 6. Moreover, the code of the proposed algorithm and the results are available at https://github.com/alejandropolo/MonoKAN

2. Kolmogorov-Arnold Networks

As mentioned before, while Multi-Layer Perceptrons (MLPs) draw their inspiration from the universal approximation theorem, our attention shifts to the Kolmogorov-Arnold representation theorem.

The Kolmogorov-Arnold representation theorem states that any multivariate continuous function can be expressed as a finite sum of continuous functions of a single variable. Specifically, for any function $f : [0,1]^n \to \mathbb{R}$, there exist univariate functions $\phi_{q,p} : [0,1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$ such that

$$f(x_1, x_2, \ldots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right). \tag{1}$$

The major breakthrough presented in (Liu et al., 2024b) comes from recognizing the similarities between MLPs and KAN. Just as MLPs increase their depth and expressiveness by stacking multiple layers, KANs can similarly enhance their predictive power through layer stacking once a KAN layer is properly defined.

To understand this concept further, a KAN layer with n_{in} inputs and n_{out} outputs can be described as a matrix of 1D functions

$$\mathbf{\Phi} = \{\phi_{q,p}\}, \quad p = 1, 2, \cdots, n_{\text{in}}, \quad q = 1, 2 \cdots, n_{\text{out}},$$

where $\phi_{q,p}(\cdot)$ are functions parameterized by learnable coefficients. Consequently, applying a KAN layer with n_{in} inputs and n_{out} outputs to an input $\mathbf{x} \in \mathbb{R}^{n_{\text{in}}}$ is defined through the following action of the matrix of functions.

$$\begin{aligned} \mathbf{\Phi}(\mathbf{x}) &= \left(\phi_{q,p}\right)_{1 \le p \le n_{\text{in}}, 1 \le q \le n_{\text{out}}} \cdot (x_p)_{1 \le p \le n_{\text{in}}} \\ &= \sum_{p=1}^{n_{\text{in}}} \phi_{q,p}(x_p), \quad q = 1, 2, \cdots, n_{\text{out}}. \end{aligned}$$

Accordingly, the Kolmogorov-Arnold theorem (Eq. (1)) can be represented within the KAN framework as a composition of a KAN layer with $n_{\text{in}} = n$ and $n_{\text{out}} = 2n + 1$ and a KAN layer with $n_{\text{in}} = 2n + 1$ and $n_{\text{out}} = 1$.

Given that all functions to be learned are univariate, we can approximate each 1D function as a spline curve with learnable coefficients. However, it is important to note that the functions $\phi_{q,p}(\cdot)$ specified by the Kolmogorov-Arnold theorem are arbitrary. In practice, though, a specific class of functions parameterized by a finite number of parameters is typically used. This practical consideration justifies the need of using more KAN layers than just the proposed by Eq. (1).

Adopting the notation from (Liu et al., 2024b), we define the structure of a KAN as $[n_0, n_1, \cdots, n_L]$, where n_l denotes the number of nodes in the l^{th} layer. The i^{th} neuron in the l^{th} layer is represented by (l, i), and its activation value by $x_{l,i}$. Between layer l and layer $l + 1$, there are $n_l n_{l+1}$ activation functions and the activation function connecting (l, i) and $(l + 1, j)$ is denoted by

$$\begin{aligned} &\phi_{l,j,i}(\cdot), \quad l = 0, \cdots, L-1, \\ &\quad i = 1, \cdots, n_l, \quad j = 1, \cdots, n_{l+1}. \end{aligned}$$

The pre-activation input for $\phi_{l,j,i}(\cdot)$ is $x_{l,i}$, while its post-activation output is represented by $\tilde{x}_{l,j,i} := \phi_{l,j,i}(x_{l,i})$. Moreover, the activation value of the neuron $(l+1, j)$ is then calculated as the sum of all incoming post-activation values:

$$x_{l+1,j} = \sum_{i=1}^{n_l} \tilde{x}_{l,j,i} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}), \qquad j = 1, \cdots, n_{l+1}. \tag{2}$$

Therefore, the KAN layer can be stated in its matrix form as

$$\mathbf{x}_{l+1} = \underbrace{\begin{pmatrix} \phi_{l,1,1}(\cdot) & \phi_{l,1,2}(\cdot) & \cdots & \phi_{l,1,n_l}(\cdot) \\ \phi_{l,2,1}(\cdot) & \phi_{l,2,2}(\cdot) & \cdots & \phi_{l,2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{l,n_{l+1},1}(\cdot) & \phi_{l,n_{l+1},2}(\cdot) & \cdots & \phi_{l,n_{l+1},n_l}(\cdot) \end{pmatrix}}_{\mathbf{\Phi}_l} \mathbf{x}_l, \tag{3}$$

where $\mathbf{\Phi}_l$ is the function matrix corresponding to the l^{th} KAN layer. Consequently, a KAN with L-layers can be described as a composition of the function matrices $\mathbf{\Phi}_l, 0 \le l < L$, such that for a given input vector $\mathbf{x} \in \mathbb{R}^{n_0}$, the output of the KAN is:

$$\text{KAN}(\mathbf{x}) = \left(\mathbf{\Phi}_{L-1} \circ \cdots \circ \mathbf{\Phi}_1 \circ \mathbf{\Phi}_0\right)(\mathbf{x}).$$

3. MonoKAN

This section introduces the necessary theoretical development and the proposed algorithm for generating a set of sufficient conditions to ensure that a KAN is partially monotonic w.r.t. a specific subset of input variables. First of all, the concept of partial monotonicity will be explained, as well as the way to ensure monotonicity in cubic Hermite splines. Subsequently, the main theorem outlining the sufficient conditions for a KAN to be partially monotonic will be presented. Finally, the proposed algorithm to ensure that a KAN meets these conditions will be described.

3.1. Partial Monotonicity

To begin with, let us start by presenting the concept of partial monotonicity. Intuitively, a function $f : \mathbb{R}^n \to \mathbb{R}$ is increasing (resp. decreasing) partially monotonic w.r.t the r^{th} input, with $1 \leq r \leq n$, whenever the output increases (decreases) if the r^{th} input increases. Mathematically speaking, a function $f : \mathbb{R}^n \to \mathbb{R}$ is increasing (resp. decreasing) partially monotonic w.r.t. its r^{th} input if

$$f(x_1, \dots, x_r, \dots, x_n) \leq f(x_1, \dots, x'_r \dots, x_n), \forall\, x_r \leq x'_r \quad (4)$$
$$(\text{resp. } f(x_1, \dots, x_r, \dots, x_n) \geq f(x_1, \dots, x'_r \dots, x_n)).$$

Therefore, f will be partially monotonic w.r.t. a subset of its input variables $\{x_{i_1}, \dots, x_{i_k}\}$ where $0 \leq k \leq n$, if Eq. (4) holds for each i_j simultaneously with $j \in \{1, \dots, k\}$.

3.2. Monotonic Cubic Hermite Splines

As mentioned before, each function to be learned in a KAN layer is univariate, allowing for various parameterization methods for each 1D function. While the original formulation presented in (Liu et al., 2024b) employs B-splines to approximate these univariate functions, in this paper, it is proposed the use of cubic Hermite splines. The advantage of cubic Hermite splines lies in the well-known sufficient conditions for monotonicity (Fritsch and Carlson, 1980; Aràndiga et al., 2022). Besides, for a sufficiently smooth function and a fine enough grid, the resulting cubic Hermite spline converges uniformly to the desired function (Hall and Meyer, 1976).

A cubic Hermite spline, or cubic Hermite interpolator, is a type of spline where each segment is a third-degree polynomial defined by its values and first derivatives at the endpoints of the interval it spans. Consequently, the spline is C^1 continuous within the interval of definition.

To formally define a cubic Hermite spline, consider a set of knots x_k, values y_k and derivative values m_k at each of the knots x_k given by $\mathcal{X} = \{(x_k, y_k, m_k) \mid \forall\ k \in I = \{1, 2, \dots, n\}\}$. Then, the cubic Hermite spline p is a set of $n-1$ cubic polynomials such that, in each subinterval $I_k = [x_k, x_{k+1}]$, it is verified that

$$\begin{aligned} p(x_k) &= y_k, \forall\ k \in I \\ p'(x_k) &= m_k, \forall\ k \in I. \end{aligned} \quad (5)$$

Therefore, the above conditions ensure that the spline matches both the function values and the slopes at each data point. An intuitive example illustrating this behavior is shown in Figure 1, where the spline passes through the knots and aligns with the expected derivatives.

Furthermore, on each subinterval $I_k = [x_k, x_{k+1}]$, the cubic Hermite spline p can be expressed in its Hermite form as

$$p(t) = h_{00}(t)y_k + h_{10}(t)(x_{k+1} - x_k)m_k + h_{01}(t)y_{k+1} + h_{11}(t)(x_{k+1} - x_k)m_{k+1},$$

where $t = \frac{x - x_k}{x_{k+1} - x_k}$ and $h_{00}, h_{10}, h_{01}, h_{11}$ are the Hermite basis functions defined as follows

$$h_{00}(t) = 2t^3 - 3t^2 + 1,$$

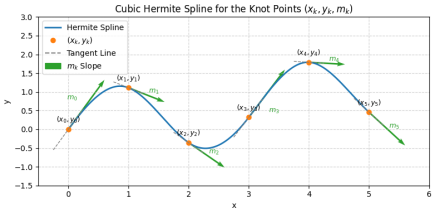

Figure 1: Illustration of a cubic Hermite spline. The spline smoothly interpolates the function values while respecting the slope (derivative) constraints at each knot.

$$h_{10}(t) = t^3 - 2t^2 + t,$$
$$h_{01}(t) = -2t^3 + 3t^2,$$
$$h_{11}(t) = t^3 - t^2.$$

Once the terminology of cubic Hermite splines has been established, we now consider the conditions required for the resulting spline to be monotonic as shown in Fritsch and Carlson (1980). According to Eq. (5), to achieve an increasing (resp. decreasing) monotonic cubic Hermite spline, it is necessary that $y_k \leq y_{k+1},\ \forall\ k \in I$ (resp. $y_k \geq y_{k+1},\ \forall\ i \in I$). Additionally, to ensure monotonicity, it is clear that considering

$$d_k = \frac{y_{k+1} - y_k}{x_{k+1} - x_k},$$

the slope of the secant line between two successive points x_k and x_{k+1}, then the derivative at each point within the interval I_k must match the sign of d_k to maintain monotonicity. Specifically, if $d_k = 0$, then both m_k and m_{k+1} must also be zero, as any other configuration would disrupt monotonicity between x_k and x_{k+1}. These conditions, stated in the following lemma, establish necessary conditions for a cubic Hermite spline to be monotonic.

Lemma 1 (Necessary conditions for monotonicity, (Aràndiga et al., 2022, Theorem 1.1))**.** *Let p_k be an increasing (resp. decreasing) monotone cubic Hermite spline of the data $\mathcal{X} = \{(x_k, y_k, m_k), (x_{k+1}, y_{k+1}, m_{k+1})\}$ such that the control points verify that $y_k \leq y_{k+1}$ (resp. $y_k \geq y_{k+1}$). Then*

$$\begin{aligned} &m_k \geq 0 \quad \textit{and} \quad m_{k+1} \geq 0. \\ &\big(\textit{resp. } m_k \leq 0 \quad \textit{and} \quad m_{k+1} \leq 0\big) \end{aligned}$$

Moreover, if $d_k = 0$ then p_k is monotone (in fact, constant) if and only if $m_k = m_{k+1} = 0$.

For the more general case when $d_k \neq 0$, Fritsch and Carlson (1980) introduced the parameters α_k and β_k, defined as

$$\alpha_k := \frac{m_k}{d_k},$$
$$\beta_k := \frac{m_{k+1}}{d_k}.$$

These parameters provide the necessary framework for establishing sufficient conditions for monotonicity.

Lemma 2 (Sufficient conditions for monotonicity, (Fritsch and Carlson, 1980, Lemma 2 and §4)). *Let $I_k = [x_k, x_{k+1}]$ be an interval between two knot points and p_k be a cubic Hermite spline of the data $\mathcal{X} = \{(x_k, y_k, m_k), (x_{k+1}, y_{k+1}, m_{k+1})\}$ such that the control points verify that $y_k < y_{k+1}$ (resp. $y_k > y_{k+1}$). Then, the cubic Hermite spline p_k is increasingly (resp. decreasingly) monotone on I_k if*

$$\alpha_k := \frac{m_k}{d_k} \geq 0, \qquad \beta_k := \frac{m_{k+1}}{d_k} \geq 0$$
$$\left(resp.\ \alpha_k := \frac{m_k}{d_k} \leq 0, \qquad \beta_k := \frac{m_{k+1}}{d_k} \leq 0\right)$$

and

$$\alpha_k^2 + \beta_k^2 \leq 9.$$

By adhering to these conditions, one can ensure that the cubic Hermite spline remains monotonic over its entire domain.

3.3. Mathematical Certification of Partial Monotonicity of a KAN

Having introduced the definition of partial monotonicity and the necessary conditions for a cubic Hermite spline to be monotonic, we now present the main theoretical result, which provides a set of sufficient conditions for a KAN to be certified partial monotonic. For simplicity, we will assume that the KAN is partially monotonic with respect to the r^{th} input. Therefore, when handling multiple monotonic features, the same conditions applied to the r^{th} input will be applied to each monotonic feature.

Recall that, according to Eq. (3), a KAN can be described as a combination of univariate functions, parametrized in this paper as cubic Hermite splines, followed by a multivariate sum. Since a linear combination of monotonic functions with positive coefficients is monotonic, and the composition of monotonic functions is also monotonic (see Proposition B.1), we propose constructing a partially monotonic KAN by ensuring that each of the cubic Hermite splines in the KAN is monotonic.

To achieve this, consider a KAN with $n_0 = n$ inputs, expected to be increasingly (decreasingly) partially monotonic with respect to the r^{th} input, where $1 \leq r \leq n$. Consequently, to obtain an increasingly (decreasingly) partially monotonic KAN with respect to the r^{th} input, it is sufficient to ensure that the n_1 activations originating from the r^{th} input are increasingly (decreasingly) monotonic and that any of the following neurons, where the output of the activation functions generated by the r^{th} input is considered as part of the input, must also be increasingly monotonic. This idea of this procedure is illustrated in Figure 2, which provides an example of a partial monotonic KAN.

On the other hand, as mentioned in (Liu et al., 2024b), the activation value of the $(l+1, j)$ node is not computed merely as a sum of spline outputs of the previous layer, but rather as a combination of spline transformations and a standard activation function. Specifically, each activation $\phi_{l,j,i}$ is defined as a linear combination of a spline $\varphi_{l,j,i}$ and a base activation function $\mathbf{b}$ (e.g., Sigmoid or SiLU), both evaluated at the pre-activation $x_{l,i}$. This construction is visually detailed in Figure 3, where the roles of $\varphi_{l,j,i}$ and $\mathbf{b}$, as well as their respective weights $\omega^{\varphi}_{l,j,i}$ and $\omega^{b}_{l,j,i}$, are explicitly depicted. Therefore, Eq. (2) is transformed into:

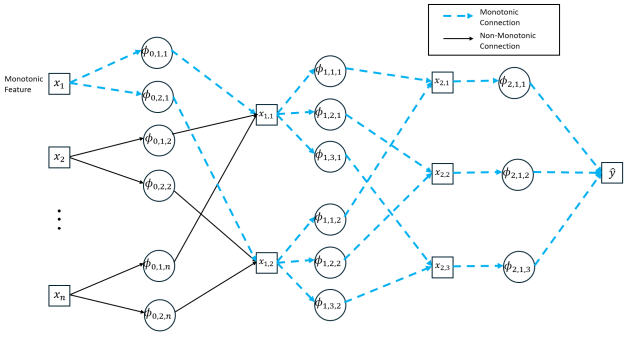

Figure 2: Scheme of monotonic and non-monotonic connections of a partial monotonic KAN w.r.t the first input with layers $[n, 2, 3, 1]$

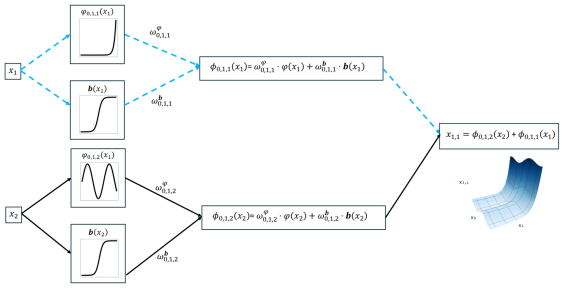

Figure 3: Illustration of a simplified MonoKAN, partially monotonic w.r.t. the first input, with two inputs and one output. Each connection between neuron (l, i) and $(l+1, j)$ includes a cubic Hermite spline $\varphi_{l,j,i}$ and a base activation $\mathbf{b}$, weighted respectively by $\omega^{\varphi}_{l,j,i}$ and $\omega^{\mathbf{b}}_{l,j,i}$. The outputs of these functions are summed to produce the final activation.

$$x_{l+1,j} = \sum_{i=1}^{n_l} \tilde{x}_{l,j,i} = \sum_{i=1}^{n_l} \phi_{l,j,i}(x_{l,i}) = \sum_{i=1}^{n_l} \left(\omega^{\varphi}_{l,j,i} \cdot \varphi_{l,j,i}(x_{l,i}) + \omega^{\mathbf{b}}_{l,j,i} \cdot \mathbf{b}(x_{l,i})\right) + \theta_{l,j}, \quad \forall\, j = 1, \cdots, n_{l+1}, \tag{6}$$

where $\omega^{\varphi}_{l,j,i}$ and $\omega^{\mathbf{b}}_{l,j,i}$ are the weights associated respectively with the spline function and the base activation function connecting neuron (l, i) to neuron $(l+1, j)$. Moreover, $\theta_{l,j}$ represents the bias added to neuron $(l+1, j)$. Besides, each spline $\varphi_{l,j,i}$ is given by the cubic Hermite spline of the data $\mathcal{X} = \{(x^k_{l,j,i}, y^k_{l,j,i}, m^k_{l,j,i}) \mid \forall\ 1 \leq k \leq K\}$ where K is the number of knots.

Consequently, if $\omega^{\varphi}_{l,j,i}$ and $\omega^{\mathbf{b}}_{l,j,i}$ are positive and $\varphi_{l,j,i}$ and $\mathbf{b}$ are monotonic, for all $1 \leq i \leq n^l$, $1 \leq j \leq n^{l+1}$ and $0 \leq l \leq L-1$, then the resulting activation value function is also monotonic. Moreover, conditions established in Lemma 1 and Lemma 2 give us an intuitive way of imposing monotonicity for each of the splines $\varphi_{l,j,i}$.

Additionally, it is proposed that the cubic Hermite spline is extended linearly outside the interval of the definition of the spline $I = [x^1, x^K]$, with slope m^1 to the left of x^1 and slope m^K to the right of x^K. This linear extrapolation ensures that each of the splines is C^1 continuous in $\mathbb{R}$. Moreover, it also guarantees that the splines are monotonic, not just within the interval of definition, but in $\mathbb{R}$. Hence, the resulting KAN maintains monotonic consistency beyond the data domain and thereby certifies the monotonicity of the model across $\mathbb{R}^n$.

This idea is presented in Theorem 3 that states the set of sufficient conditions needed to guarantee partial monotonicity of a KAN w.r.t the r^{th} input. A detailed glossary of the notation used can be found in Appendix A. Moreover, a complete proof of the above theorem can be found in Appendix B.

Theorem 3 (Certified Partial Monotonicity Conditions)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a KAN composed of L layers and n inputs, where each spline function is defined over a common interval $I = [x^1, x^K]$ with K knots. For each layer l, let $y^k_{l,j,i}$ denote the spline value at knot k for the connection from neuron i in layer l to neuron j in layer $l+1$. Let $d^k_{l,j,i} = (y^{k+1}_{l,j,i} - y^k_{l,j,i})/(x^{k+1}_{l,j,i} - x^k_{l,j,i})$ be the difference between the knot values, and let $m^k_{l,j,i}$ be the corresponding spline slope at knot k. Denote by $\omega^{\varphi}_{l,j,i}$ and $\omega^{\mathbf{b}}_{l,j,i}$ the weights applied to the spline and the base activation function output, respectively. Then, assuming the base activation function $\mathbf{b}$ is monotonically increasing, the KAN f is certified increasingly (respectively decreasingly) monotonic with respect to the r^{th} input variable if the following conditions are satisfied $\forall\ 1 \leq k \leq K$:*

Input layer conditions for increasing (resp. decreasing) monotonicity with respect to the r^{th} input:

1. $\omega^{\varphi}_{0,j,r} \geq 0$, $\omega^{\mathbf{b}}_{0,j,r} \geq 0$ *(resp. $\omega^{\varphi}_{0,j,r} \geq 0, \omega^{\mathbf{b}}_{0,j,r} \leq 0$) (non-negative weights on input x_r)*
2. $y^{k+1}_{0,j,r} \geq y^k_{0,j,r}$, *i.e.* $d^k_{0,j,r} \geq 0$ *(resp. $y^{k+1}_{0,j,r} \leq y^k_{0,j,r}$, i.e. $d^k_{0,j,r} \leq 0$) (monotonic spline values)*
3. *If $d^k_{0,j,r} = 0$, then $m^k_{0,j,r} = m^{k+1}_{0,j,r} = 0$ (zero slope at flat intervals)*
4. *If $d^k_{0,j,r} > 0$, then $m^k_{0,j,r} \geq 0$, $m^{k+1}_{0,j,r} \geq 0$ (resp. if $d^k_{0,j,r} < 0$, $m^k_{0,j,r}$, $m^{k+1}_{0,j,r} \leq 0$) (positive slopes)*
5. *If $d^k_{0,j,r} > 0$, then $\left(\alpha^k_{0,j,r}\right)^2 + \left(\beta^k_{0,j,r}\right)^2 \leq 9$ (sufficient condition for monotonicity (Lemma 2))*

Hidden layers ($1 \leq l \leq L-1$) increasing conditions:

6. $\omega^{\varphi}_{l,j,i} \geq 0$, $\omega^{\mathbf{b}}_{l,j,i} \geq 0$ *(non-negative weights)*
7. $y^{k+1}_{l,j,i} \geq y^k_{l,j,i}$ *(monotonic spline values)*
8. *If $d^k_{l,j,i} = 0$, then $m^k_{l,j,i} = m^{k+1}_{l,j,i} = 0$ (zero slope at flat intervals)*
9. *If $d^k_{l,j,i} > 0$, then $m^k_{l,j,i}, m^{k+1}_{l,j,i} \geq 0$ (positive slopes at non flat intervals)*
10. *If $d^k_{l,j,i} > 0$, then $\left(\alpha^k_{l,j,i}\right)^2 + \left(\beta^k_{l,j,i}\right)^2 \leq 9$ (sufficient condition for monotonicity (Lemma 2)).*

Lastly, it is worth mentioning that the proposed method, considering a linear combination with positive coefficients of monotonic functions, could be seen as analogous to (Archer and Wang, 1993), where a traditional MLP architecture with a ReLU activation function is constrained to have positive weights. However, the constraint proposed in (Archer and Wang, 1993) significantly reduces the expressive power as it forces the output function to be convex (Liu et al., 2020). In contrast, MonoKAN is capable of generating monotonic non-convex functions because it leverages monotonic cubic Hermite splines, which allow for flexible piecewise constructions and can model complex, non-convex shapes.

3.4. MonoKAN Algorithm

Finally, we present the algorithm that guarantees a KAN satisfies the sufficient conditions defined in Theorem 3, thereby certifying the network as partially monotonic. To enforce these constraints during training, we introduce a clamping mechanism that adjusts the learned parameters after each optimization step to ensure that the parameters remain within the valid range.

Although implemented as simple bound enforcement (i.e., clipping values to lie within predefined intervals), this procedure is mathematically equivalent to a projection onto a convex constraint set. As such, it can be interpreted as a special case of Projected Gradient Descent (PGD), where the projection ensures constraint satisfaction after each update (Bubeck, 2015). This projection step is performed independently of the gradient computation and does not alter the optimizer's internal state, making it compatible with commonly used algorithms such as SGD and Adam.

Mathematically, consider $f : \mathbb{R}^n \to \mathbb{R}$, a KAN with L layers, expected to be partially monotonic with respect to the r^{th} input ($1 \leq r \leq n$). The sufficient conditions outlined in Theorem 3 are enforced through the clamping mechanism described in the *applyCons* Algorithm, which is applied at every training epoch. As detailed in Section 3.3, clamping must be applied to the spline activations directly influenced by x_r and the downstream layers. A full pseudocode implementation is provided in the *MonoKAN* Algorithm.

4. Experiments

To assess the practical applicability of the proposed method, we divide our experimental analysis into two parts. First, we present a case study based on a two-dimensional setting from the *ESL* dataset (Ben-David et al., 1989), intended to provide intuitive insight into the behavior of

Algorithm 1 *applyCons*

Require: Parameters: 1-D arrays $\omega^{\varphi}_{l,:,i}, \omega^{\mathbf{b}}_{l,:,i}$ and 2-D arrays $y^k_{l,:,i}, m^k_{l,:,i}, x^k_{l,:,i}$ with $0 \leq k \leq K-1$.
Ensure: Adjusted parameters fulfilling sufficient conditions from Theorem 3 for i^{th} input of layer l.

$\omega^{\varphi}_{l,:,i}, \omega^{\mathbf{b}}_{l,:,i} \leftarrow \max(0, \omega^{\varphi}_{l,:,i}), \max(0, \omega^{\mathbf{b}}_{l,:,i})$
for k in range$(K-1)$ **do**
  $y^{k+1}_{l,:,i} \leftarrow \max(y^{k+1}_{l,:,i}, y^k_{l,:,i})$ {Impose that the sequence of control points is increasing}
  $d^k_{l,:,i} \leftarrow \frac{y^{k+1}_{l,:,i} - y^k_{l,:,i}}{x^{k+1}_{l,:,i} - x^k_{l,:,i}}$
  if $d^k_{l,:,i} = 0$ **then**
    $m^k_{l,:,i}, m^{k+1}_{l,:,i} \leftarrow 0$
  else
    $m^k_{l,:,i}, m^{k+1}_{l,:,i} \leftarrow \max\left(0, m^k_{l,:,i}\right), \max\left(0, m^{k+1}_{l,:,i}\right)$ {Impose positivity of the derivatives}
    $\alpha^k_{l,:,i} \leftarrow \frac{m^k_{l,:,i}}{d^k_{l,:,i}}$
    $\beta^k_{l,:,i} \leftarrow \frac{m^{k+1}_{l,:,i}}{d^k_{l,:,i}}$
    if $\left(\alpha^k_{l,:,i}\right)^2 + \left(\beta^k_{l,:,i}\right)^2 > 9$ **then**
      $\tau^k_{l,:,i} \leftarrow \frac{3}{\sqrt{(\alpha^k_{l,:,i})^2 + (\beta^k_{l,:,i})^2}}$
      $\alpha^k_{l,:,i} \leftarrow \tau^k_{l,:,i} \cdot \alpha^k_{l,:,i}$
      $\beta^k_{l,:,i} \leftarrow \tau^k_{l,:,i} \cdot \beta^k_{l,:,i}$
      $m^k_{l,:,i} \leftarrow \alpha^k_{l,:,i} \cdot d^k_{l,:,i}$
      $m^{k+1}_{l,:,i} \leftarrow \beta^k_{l,:,i} \cdot d^k_{l,:,i}$
    end if
  end if
end for
return $\omega^{\varphi}_{l,:,i}, \omega^{\mathbf{b}}_{l,:,i}, y^k_{l,:,i}$ and $m^k_{l,:,i}$.

Algorithm 2 *MonoKAN Algorithm*

Require: KAN model f with L layers and K knots, maximum number of epochs *max_epochs*, index r of the increasing (resp. decreasing) monotonic feature.
Ensure: Adjusted parameters of the KAN to fulfill sufficient conditions from Theorem 3.

for epoch $= 1$ **to** *max_epochs* **do**
  Compute loss: $\mathcal{L} \leftarrow \mathit{ComputeLoss}(f)$
  Perform optimizer step: $f \leftarrow \mathit{OptimizerStep}(f, \mathcal{L})$
  for l in range(L) **do**
    if $l = 0$ **then**
      $\omega^{\varphi}_{0,:,r}, \omega^{\mathbf{b}}_{0,:,r}, y^k_{0,:,r}, m^k_{0,:,r} \leftarrow \mathit{applyCons}(\omega^{\varphi}_{0,:,r}, \omega^{\mathbf{b}}_{0,:,r}, y^k_{0,:,r}, m^k_{0,:,r}, x_{0,:,r})$
      (resp. $\omega^{\varphi}_{0,:,r}, -\omega^{\mathbf{b}}_{0,:,r}, -y^k_{0,:,r}, -m^k_{0,:,r} \leftarrow \mathit{applyCons}(\omega^{\varphi}_{0,:,r}, -\omega^{\mathbf{b}}_{0,:,r}, -y^k_{0,:,r}, -m^k_{0,:,r}, x_{0,:,r})$)
    else
      for i in range$\left(n^l\right)$ **do**
        $\omega^{\varphi}_{l,:,i}, \omega^{\mathbf{b}}_{l,:,i}, y^k_{l,:,i}, m^k_{l,:,r} \leftarrow \mathit{applyCons}(\omega^{\varphi}_{l,:,i}, \omega^{\mathbf{b}}_{l,:,i}, y^k_{l,:,i}, m^k_{l,:,r}, x_{l,:,i})$
      end for
    end if
  end for
end for
return Partial monotonic KAN w.r.t the r^{th} input

MonoKAN and to illustrate the ethical importance of certified partial monotonicity. This case study aims to visually analyze how unconstrained or partially constrained models can produce unfair or counterintuitive predictions, motivating the need for strict monotonicity guarantees in certain domains.

Next, we conduct a comprehensive set of experiments across multiple benchmark datasets to evaluate MonoKAN's performance relative to existing state-of-the-art models. Although no prior work has introduced a certified partial monotonic Kolmogorov–Arnold Network, we compare against leading monotonic MLP-based methods reported in the literature.

To ensure a fair and consistent comparison, we adopted the experimental procedures established by Liu et al. (2020) and Sivaraman et al. (2020), which are widely accepted in the literature. Therefore, for each dataset, the experiments were conducted three times to report the mean and the standard deviation. Moreover, the dataset is divided using a three-way split, where 80% of the data is used for training and validation, and the remaining 20% for testing. Within the training portion, an additional 80%/20% split is used to separate a validation set for early stopping. Consequently, although we benchmark our results against the state-of-the-art monotonic architectures (Nolte et al., 2022; Runje and Shankaranarayana, 2023), we reran their experiments using the methodology of Liu et al. (2020) and Sivaraman et al. (2020). This approach ensures methodological consistency and allows for a fair comparison across the different studies. To support reproducibility and clarify the scope of our evaluation, we have added Appendix C, which contains detailed descriptions of all datasets used, including task type, input dimensionality, number of monotonic features, and the rationale behind the constraints applied. It is important to note that the monotonic feature selection criteria were not selected by the authors but are instead defined in the original benchmark studies (Liu et al., 2020; Sivaraman et al., 2020).

All computations were carried out on a server equipped with two Intel(R) Xeon(R) Platinum 8480C CPUs. Each processor provides 56 physical cores and supports 2 threads per core, resulting in a total of 224 hardware threads across both sockets. The CPUs operate at a maximum clock speed of 3.8 GHz. The system is also provisioned with approximately 2.1 TB of RAM, ensuring sufficient memory for all experimental workloads. Moreover, for GPU-accelerated experiments, the server was provisioned with an NVIDIA H200 GPU (driver version 550.90.07, CUDA 12.4). The proposed code was developed based on the Pytorch framework (Paszke et al., 2019). The results and the proposed

MonoKAN code can be accessed on GitHub at https://github.com/alejandropolo/MonoKAN.

4.1. Case Study: Visualizing Monotonicity and Fairness in 2D Predictions

First of all, we consider the *ESL* dataset (Ben-David et al., 1989), which is one of the benchmarks in the literature of monotonic datasets (Cano et al., 2019). The dataset consists of 488 records, each representing a job candidate evaluated for an industrial job. In particular, each of the records contains four input features representing the candidate's scores from standardized psychometric tests conducted during the assessment process. Besides, the output feature of the dataset is the alignment between the candidate's profile and the job requirements.

Out of the four input variables, we decided to select the first two input features to consider a two-dimensional case study. Therefore, this 2D scenario serves as an ideal dataset to assess and visualize the proposed method and how the conditions presented in Theorem 3, lead to a certified partial monotonic KAN. Moreover, we decided to transform the output variable into a binary classification task by assigning a value of 1 to candidates with a suitability score greater than 5, and 0 otherwise.

Given that the model's output is used to select candidates, ensuring fairness in the model's predictions is essential. In particular, a fair model should reflect an increase in a candidate's predicted suitability for the job whenever there is an improvement in any of the psychometric test scores, assuming all other factors remain constant. Consequently, penalizing candidates for demonstrating stronger competencies undermines merit-based selection and can lead to unethical outcomes (Hunter and Schmidt, 1976). Specifically, a non-monotonic model may disadvantage more qualified applicants, producing hiring decisions that appear arbitrary or biased. Therefore, enforcing monotonicity with respect to each of these input features is essential to ensure that the model's behavior aligns with fundamental fairness principles.

Moreover, an important characteristic of this dataset is the uneven distribution of samples across the input space, which results in sparse coverage in certain regions. Such sparsity may lead unconstrained models to overfit or behave unreliably in less populated areas, producing erratic or unintuitive predictions. This reinforces the relevance of incorporating certified monotonic constraints to ensure model stability and trustworthy behavior.

To evaluate the impact of certified partial monotonicity on model behavior and performance, we trained four variants of KANs on the two-dimensional ESL dataset: an unconstrained KAN using the original B-splines, an unconstrained KAN using the proposed cubic Hermite splines, a partially constrained MonoKAN (monotonic with respect to the first psychometric test), and a fully constrained MonoKAN (monotonic with respect to both input features). Table 1 reports the training and test accuracy for all four models. Notably, all models achieve nearly identical performance on the training and test sets. This suggests that enforcing monotonicity, whether partially or fully, does not appear to compromise predictive accuracy in this setting, as exploratory data analysis supports a monotonic relationship between inputs and outputs.

Table 1
Accuracy on the ESL dataset under different monotonicity constraints

Model	Train	Test
Unconstrained Hermite KAN	0.80	0.85
Unconstrained B-splines KAN	0.80	0.85
MonoKAN (1 feature monotonic)	0.79	0.85
MonoKAN (2 features monotonic)	0.79	0.85

Although the accuracy scores across all four models are nearly identical, the key distinction lies in the fairness and monotonicity of their predictions, as illustrated in Figure 4. Both unconstrained models (Figure 4 (a),(b)) displays several violations of monotonicity: increasing the score in either psychometric test 1 or test 2 can lead to a lower predicted suitability score. For example, considering Figure 4 (a), comparing the selected candidates A and B, we observe that although candidate B has a higher score on psychometric test 1 and an equal score on test 2, the model assigns a higher suitability score to candidate A, demonstrating a counterintuitive and unfair outcome. A similar issue is observed between candidates A and C: both have the same score on psychometric test 1, but candidate C has a higher score on test 2, yet receives a lower prediction. The same behaviour can be easily identified in the case of the unconstrained KAN using the B-splines formulation. Consequently, these instances reveal how unconstrained models may fail to uphold basic fairness expectations in high-stakes contexts such as hiring.

On the other hand, the partially constrained model (Figure 4 (c)), which enforces monotonicity only with respect to psychometric test 1, successfully eliminates violations along that axis. However, it still produces irregularities with respect to test 2. This is evident in the region around candidates G and H, where higher values in test 2 still lead to lower predicted outcomes, indicating that partial monotonicity is insufficient when fairness must be guaranteed in all relevant dimensions. In contrast, the fully constrained MonoKAN model (Figure 4 (d) enforces certified monotonicity with respect to both inputs, resulting in a smooth and consistent decision surface. The model output increases as either psychometric test score improves, ensuring alignment with ethical expectations and eliminating unfair behaviors observed in the other configurations.

To better illustrate these monotonicity violations at a finer granularity, Figure 5 presents one-dimensional slices of the prediction's surface of each of the aforementioned trained models, fixing one input while varying the other along the minimum and the maximum value of the dataset. These plots directly correspond to the candidate comparisons highlighted in Figure 4, allowing us to isolate how

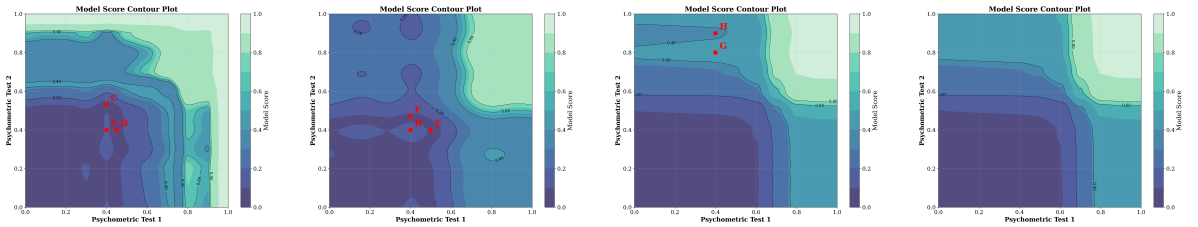

Figure 4: Contour plots of model outputs over the 2D input space (psychometric test 1 vs. psychometric test 2) for (a) unconstrained cubic Hermite KAN, (b) unconstrained B-spline KAN, (c) partially constrained MonoKAN (monotonic with respect to the first psychometric test), and (d) fully constrained MonoKAN. In (a) and (b), the model exhibits several violations of monotonicity, as highlighted by candidate comparisons (e.g., A vs. B, and A vs. C), where improved input scores lead to lower predicted suitability. In (c), monotonicity is enforced only along test 1, improving consistency along that axis but still allowing irregularities along test 2 (G vs H). In (d), full monotonicity constraints result in a smooth, fair decision surface aligned with ethical expectations.

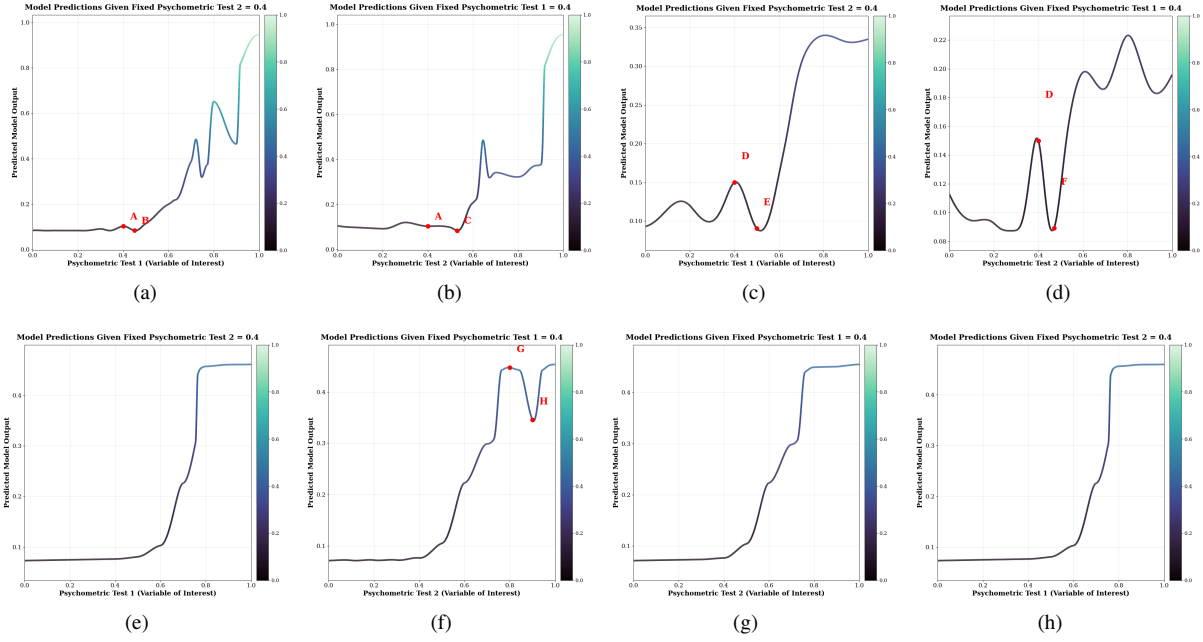

Figure 5: One-dimensional slices of the model outputs across psychometric test 1 and test 2, with the complementary variable fixed at 0.4, to visualize local monotonicity violations. Each row corresponds to a different model architecture: (a, b) unconstrained cubic Hermite KAN, (c, d) unconstrained B-spline KAN, (e, f) partially constrained MonoKAN (monotonic with respect to the first psychometric test), and (g, h) fully constrained MonoKAN. Red dots highlight specific counterexamples (e.g., A vs. B, D vs. E, G vs. H) where increasing an input leads to a decrease in the predicted output, violating monotonicity. These examples are consistent with the patterns observed in the contour plots of Figure 4, providing further evidence of undesirable behaviors in unconstrained models. Notably, all violations are eliminated in the fully constrained MonoKAN (g, h), confirming its certified monotonic behavior across both input dimensions.

model predictions behave along each axis. For instance, in Figures 5(a)–(d), clear local drops in predicted suitability are observed when increasing one psychometric test score while holding the other fixed, confirming the violations highlighted in the contour plots. These non-monotonic trends are particularly evident around points A–B or A-C and D–E, D-F, emphasizing the ethical risks of using unconstrained models. In the partially constrained MonoKAN model, shown in Figures 5(e)–(f), we observe that no monotonicity violations occur when psychometric test 1 is varied (Figure 5(e)), as constraints are imposed along this dimension. However, when psychometric test 2 is varied (Figure 5(f)), the model still exhibits non-monotonic behavior, e.g., a drop between points G and H, highlighting that enforcing monotonicity in only one input dimension may be insufficient in fairness-critical applications. Finally, these irregularities are fully eliminated in Figures 5(g)–(h), where the fully constrained MonoKAN produces strictly non-decreasing outputs along

both inputs. This further supports the effectiveness of certified monotonicity in enforcing reliable and interpretable model behavior.

Finally, to understand how these behaviors emerge, Figure 6 shows the learned spline activations for each neuron. In both unconstrained models, the spline activations exhibit local non-monotonicities that contribute to the counterintuitive predictions. These irregularities are partially corrected in the partially constrained model, and fully eliminated in the certified MonoKAN, where all activations conform to the required monotonicity constraints.

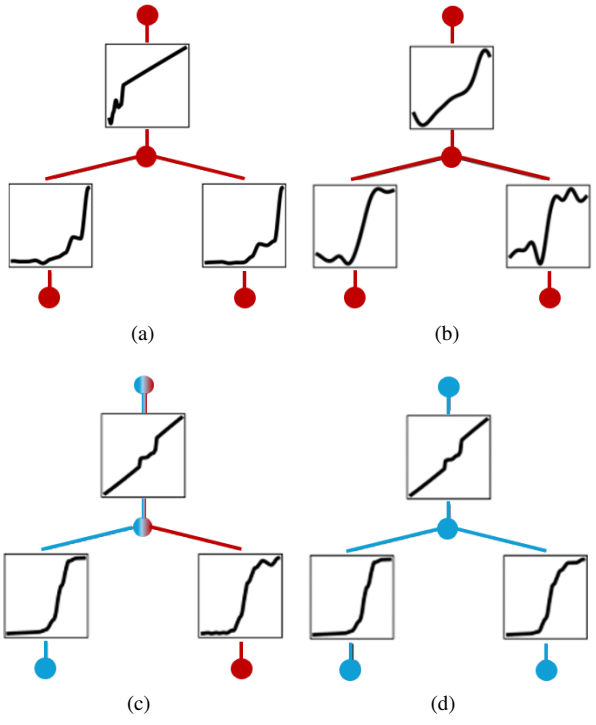

Figure 6: Spline activation functions learned by the three models: (a) unconstrained cubic Hemite KAN, (b) unconstrained B-spline KAN, (c) partially constrained MonoKAN (monotonic with respect to psychometric test 1), and (c) fully constrained MonoKAN (monotonic with respect to both input features). The black curves represent the learned univariate spline activations. The unconstrained model exhibits non-monotonic shapes while the partially constrained model exhibits non-monotonic spline activations in the second input feature. Finally, in the certified monotonic MonoKAN all activations follow monotonic patterns.

4.2. Benchmarking MonoKAN Against State-of-the-Art Algorithms

After the illustrative example presented in the previous section, we now evaluate the performance of the proposed algorithm on several real-world datasets. In the initial set of experiments, we used the following datasets proposed by (Liu et al., 2020): COMPAS (Angwin et al., 2016), a classification dataset containing 6,172 records and 13 features, including 4 features labeled as monotonic; Blog Feedback Regression (Buza, 2014), a high-dimensional regression dataset with 54,270 samples and 276 features, 8 of which are designated monotonic; and Loan Defaulter, a large-scale classification dataset with nearly half a million records and 28 features, including 5 monotonic ones [0]. Therefore, the aforementioned datasets enable a realistic evaluation of MonoKAN in real-world, large-scale scenarios that involve high-dimensional feature spaces.

Moreover, following the approach described in (Nolte et al., 2022), we evaluated both MonoKAN and the Expressive Monotonic Network using two configurations: one with the full set of input features and another with a reduced subset selected via Ridge regression. In the reduced setup, we trained a Ridge model and selected the top 20 features for the Blog Feedback dataset and the top 15 for the Loan Defaulter dataset, based on the absolute value of the learned coefficients. For each model, we reported the results corresponding to the best-performing configuration.

To improve clarity and reproducibility, we have added Appendix C, which provides detailed descriptions of each dataset, including task type, number of features, number of monotonic constraints, and the rationale behind the monotonic assignments. Besides, it is important to note that the monotonic features, described in Appendix C, were defined according to the original benchmark specifications and were not determined by the authors.

On the other hand, for the second set of experiments, we employed datasets specified by Sivaraman et al. (2020): the Auto MPG dataset [1], which is a regression dataset with 3 monotonic features and is one of the benchmarks in the literature (Cano et al., 2019), and the Heart Disease dataset [2], which is a classification dataset featuring two monotonic variables. The results obtained are going to be compared with COMET (Sivaraman et al., 2020), Min-Max Net (Daniels and Velikova, 2010), Deep Lattice Network (You et al., 2017), Constrained Monotonic Networks (Runje and Shankaranarayana, 2023) and (Nolte et al., 2022).

4.2.1. Results

The obtained results after performing the experiments are summarized in Tables 2 and 3. As observed, the results obtained by the proposed MonoKAN outperform the state-of-the-art in four out of five datasets, while in the remaining experiment, our method equals the best option.

When considering the number of parameters for each model, KAN remains competitive compared to most of the approaches proposed in the literature. However, in cases where the number of input variables is substantial, KAN exhibits a higher parameter count than some approaches. This increase in parameters for datasets with numerous inputs is due to KAN's architecture, which generates at least one spline in the first layer for each input. Consequently, the model complexity and the number of parameters grow

[0] https://www.kaggle.com/datasets/wordsforthewise/lending-club/code

[1] https://archive.ics.uci.edu/dataset/9/auto+mpg

[2] https://archive.ics.uci.edu/dataset/45/heart+disease

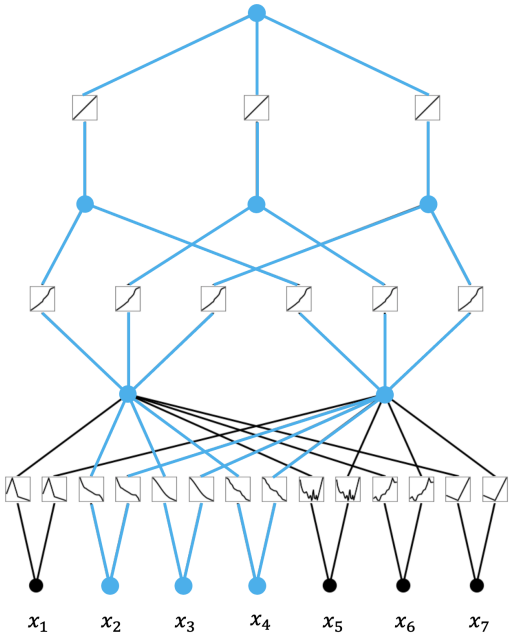

Figure 7: Spline activations of a trained decreasing partial monotonic MonoKAN w.r.t the input variables x_2, x_3 and x_4 using the Auto MPG dataset.

proportionally with the number of input variables, impacting the overall efficiency and computational requirements of KANs for datasets with a large number of variables.

On the other hand, it is important to note that MonoKAN inherits all the additional advantages of using KAN architectures compared with traditional MLPs described in the introduction, especially its enhanced interpretability arising from being easier to visualize. For instance, in Figure 7, we can observe a trained MonoKAN model using the Auto MPG dataset. This illustration highlights the specific relationships between each input variable and the output, demonstrating the certified decreasing partial monotonicity concerning input variables x_2, x_3, and x_4. The visualization effectively showcases how the MonoKAN model maintains monotonic behavior w.r.t these selected features. This is crucial for understanding and verifying the model's adherence to monotonic constraints, which can be essential for applications requiring reliable and interpretable predictions. Additionally, MonoKAN provides insight into the model's behavior and performance, allowing for a deeper analysis of the variable interactions and their impact on the output.

5. Computational Implementation Details

This section provides a detailed analysis of the computational characteristics of the proposed MonoKAN architecture, including parameter complexity, memory usage, runtime behavior, and scalability considerations.

To begin with, regarding parameter complexity, a KAN with depth L and width N requires $O(N^2L(G+k))$ parameters, where G represents the number of spline grid points and k is the spline degree (Liu et al., 2024a; Kundu et al., 2024). In our proposed MonoKAN framework, we utilize cubic Hermite splines resulting in a parameter complexity of $O(N^2L(2G))$, reamining comparable with the original KAN formulation. Hence, switching from B-splines to Hermite splines does not introduce significant differences in parameter complexity.

In comparison, a conventional MLP with the same architecture would require $O(N^2L)$ parameters (Kundu et al., 2024). Although this suggests that KANs and MonoKAN introduce an increase in parameter count, empirical studies show that KANs often require half the number of parameters than MLPs to achieve comparable performance (Liu et al., 2024b; Tran et al., 2024). This efficiency stems from the greater expressiveness of spline-based activations, which allows KAN-based models to achieve similar or better performance with reduced depth or width.

However, a key challenge in KANs is their dense connectivity: for a KAN model with n input features and a first hidden layer with n_1 neurons, the network must learn $n \cdot n_1$ univariate spline functions in the first layer, since each input is connected to every neuron through a separate spline. Therefore, this formulation can be inefficient in high-dimensional settings. To address this, sparse activation schemes have been proposed, where each neuron connects to only a subset of input features, thereby reducing the number of spline modules and memory overhead (Liu et al., 2024a). Furthermore, MultKAN (Liu et al., 2024a), a newly proposed implementation of KAN, demonstrates that memory usage can be reduced from $O(LN^2G)$ to $O(LNG)$ by eliminating unnecessary input expansions, an optimization strategy that could be incorporated into MonoKAN in future work.

On the other hand, in terms of runtime, KANs tend to be slower than MLPs, despite using fewer parameters. This is largely due to the computational cost associated with evaluating and optimizing spline functions. Prior work reports training time slowdowns ranging from 6.5x to over 100x relative to MLPs (Tran et al., 2024; Kundu et al., 2024). However, recent developments have significantly narrowed this gap. In particular, the introduction of MultKAN (Liu et al., 2024a) has improved runtime efficiency, achieving training speeds only 3.5x slower than MLPs. Further optimizations such as GPU-compatible implementations have enabled orders-of-magnitude speedups, reducing training from hours to seconds (Liu et al., 2024a). In this study, some of these advances, such as leveraging GPU computations, have been implemented. However, not all advances have been considered, as further optimizing KAN runtime is out of the scope of the paper.

While runtime efficiency remains a broader challenge for KAN-based architectures compared to MLPs, it is important to emphasize that the addition of certified monotonic constraints in MonoKAN introduces only minimal overhead relative to the standard KAN. In particular, the proposed constraints are enforced through lightweight projection operations on the network parameters as the projection is done

Method	COMPAS		Blog Feedback		Loan Defaulter	
	Parameters	Test Acc	Parameters	RMSE	Parameters	Test Acc
Isotonic	N.A.	67.6%	N.A.	0.203	N.A.	62.1%
XGBoost (Chen and Guestrin, 2016)	N.A.	68.5% ± 0.1%	N.A.	0.176 ± 0.005	N.A.	63.7% ± 0.1%
Crystal (Milani Fard et al., 2016)	25840	66.3% ± 0.1%	15840	0.164 ± 0.002	16940	65.0% ± 0.1%
DLN (You et al., 2017)	31403	67.9% ± 0.3%	27903	0.161 ± 0.001	29949	65.1% ± 0.2%
Min-Max Net (Daniels and Velikova, 2010)	42000	67.8% ± 0.1%	27700	0.163 ± 0.001	29000	64.9% ± 0.1%
Non-Neg-DNN	23112	67.3% ± 0.9%	8492	0.168 ± 0.001	8502	65.1% ± 0.1%
Certified (Liu et al., 2020)	23112	68.8% ± 0.2%	8492	0.158 ± 0.001	8502	65.2% ± 0.1%
Constrained (Runje and Shankaranarayana, 2023)	2317	65.3% ± 0.0%	1101	0.156 ± 0.001	577	65.0% ± 0.0%
Expressive (Nolte et al., 2022)	37	68.7% ± 0.0%	177	0.158[1] ± 0.003	753	**65.3**% ± 0.0 %
MonoKAN	4101	**69.0%** ± 0.0%	8546	**0.155**[1] ± 0.0	1926	**65.3**%[1] ± 0.1 %

Table 2
Comparison of the proposed MonoKAN with the state-of-the-art certified partial monotonic MLPs.
[1] Following (Nolte et al., 2022), we apply Ridge regression-based feature selection, using the top 20 features for Blog Feedback and top 15 for Loan Defaulter

Method	Auto MPG	Heart Disease
	MSE	Test Acc
Min-Max Net (Daniels and Velikova, 2010)	10.14 ± 1.54	0.75 ± 0.04
DLN (You et al., 2017)	13.34 ± 2.42	0.86 ± 0.02
COMET (Sivaraman et al., 2020)	8.81 ± 1.81	0.86 ± 0.03
Constrained (Runje and Shankaranarayana, 2023)	9.05 ± 0.25	0.86 ± 0.01
Expressive (Nolte et al., 2022)	6.23 ± 0.50	0.81 ± 0.03
MonoKAN	**6.18 ± 0.02**	**0.89 ± 0.00**

Table 3
Comparison of the proposed MonoKAN with the state-of-the-art certified partial monotonic MLPs.

using PyTorch clamping function (Paszke et al., 2019). This operation scales linearly with the number of parameters and has no significant impact at training time. Consequently, MonoKAN achieves certified partial monotonicity while preserving the computational efficiency of its underlying architecture.

To validate this claim, we conducted a runtime analysis comparing the training time of MonoKAN, the proposed cubic Hermite-based KAN (HermiteKAN), and the original B-spline-based KAN across a wide range of network sizes and depths. As shown in Figure 8, MonoKAN introduces only a marginal increase in training time over HermiteKAN, while both remain substantially more efficient than the original B-spline formulation. For instance, even at higher model sizes with four hidden layers and over 2,000 parameters, MonoKAN's mean training time per epoch remains below 0.05 seconds, showing a consistent runtime advantage over the B-spline KAN, which can exceed 0.06 seconds. These results empirically confirm that the introduction of monotonic constraints has minimal impact on training efficiency when using cubic Hermite splines and that the computational overhead is negligible relative to the gains in certified monotonicity and fairness.

Additionally, considering the comparison against the baselines of certified monotonic algorithms, MonoKAN offers a practical and scalable alternative. In contrast to prior approaches such as Certified Monotonic Neural Networks (Liu et al., 2020), which rely on Mixed Integer Linear Programming (MILP) solvers for post-hoc verification, and Counterexample-Guided Monotonic Networks (Sivaraman et al., 2020), which leverage Satisfiability Modulo Theories (SMT) solvers to identify violations, our method avoids solving expensive combinatorial problems. For instance, the MILP-based method proposed by (Liu et al., 2020) transforms the monotonicity verification task into an NP optimization problem where each neuron's activation must be modeled by binary variables, leading to exponential worst-case complexity in the number of neurons and layers (e.g., $O\left(2^{\sum_{l=1}^{L} n_l}\right)$ for a ReLU network with L layers). To mitigate intractability in deep architectures, (Liu et al., 2020) proposes to enforce monotonicity layer-wise, but this restricts the function class to compositions of monotonic functions, limiting expressiveness.

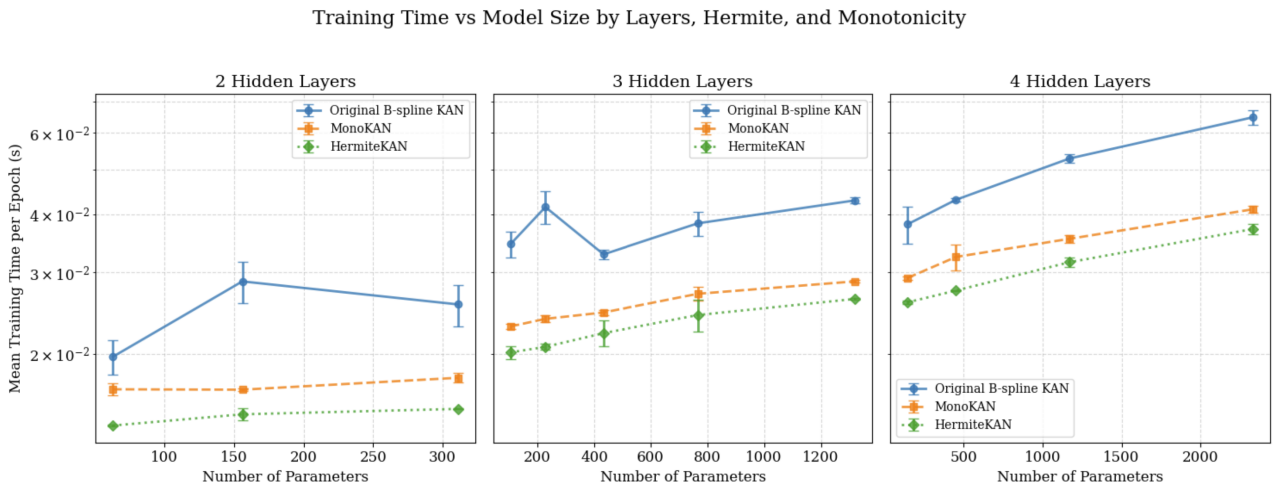

Figure 8: Mean training time per epoch (in seconds) as a function of model size, measured by number of parameters, across networks with 2, 3, and 4 hidden layers. We compare the original B-spline KAN, HermiteKAN (a non-monotonic version using the proposed cubic Hermite splines), and MonoKAN (HermiteKAN with certified monotonicity constraints). Results are averaged over 5 runs with standard deviation shown as error bars. MonoKAN introduces minimal overhead relative to HermiteKAN, and both remain substantially more efficient than the original B-spline KAN across all tested configurations.

Similarly, the SMT-based approach proposed by (Sivaraman et al., 2020) iteratively searches for monotonicity violations by formulating logical queries over network outputs. When violations are found, the corresponding counterexamples are injected into the training data with modified targets to steer the model towards monotonicity. Consequently, this process requires repeatedly querying SMT solvers and retraining the network, resulting in substantial computational overhead. Besides, both methods are also limited to piecewise linear activation functions such as ReLU, and their reliance on solver-based iterations makes them difficult to scale or extend.

In contrast, MonoKAN achieves certified monotonicity directly by design, using simple and efficient clamping operations during training, which scale linearly with the number of parameters. Furthermore, unlike MILP or SMT-based methods, which are tailored for ReLU activations and cannot be easily extended to other types of networks, MonoKAN supports a broader range of activation functions, broadening its applicability. Besides, our approach requires no additional verification steps or post-training adjustments, making it more efficient and readily deployable in practical applications.

Finally, to complement the theoretical comparison, we also performed an empirical evaluation of training time per epoch to benchmark MonoKAN against the most recent state-of-the-art monotonic models: Constrained Monotonic Networks (Runje and Shankaranarayana, 2023) and Expressive Monotonic Networks (Nolte et al., 2022). As reported in Table 4, MonoKAN achieves superior or equal predictive performance on all three benchmark datasets while maintaining comparable training times. In particular, MonoKAN trains significantly faster than the Expressive method on the COMPAS dataset, and remains comparable on the Loan Defaulter and Blog Feedback datasets. MonoKAN also outperforms Constrained Networks in terms of runtime in the Loan Defaulter dataset and offers better performance in all three tasks. These results support our claim that MonoKAN achieves certified monotonicity without compromising efficiency, and highlight its practical suitability for real-world applications.

6. Conclusion

This paper proposes a novel artificial neural network (ANN) architecture called MonoKAN, which is based on the Kolmogorov-Arnold Network (KAN). MonoKAN is designed to certify partial monotonicity across the entire input space, not just within the domain of the training data, while enhancing interpretability. To achieve this, we replace the B-splines, proposed in the original formulation of KAN, with cubic Hermite splines, which offer well-established conditions for monotonicity and can uniformly approximate sufficiently smooth functions. Our experiments demonstrate that MonoKAN consistently outperforms existing state-of-the-art methods in terms of performance metrics. Moreover, it retains the interpretability benefits of KANs, enabling effective visualization of model behavior. This combination of interpretability and certified partial monotonicity addresses a crucial need for more trustworthy and explainable AI models.

Future research will focus on extending the architecture to splines of arbitrary degrees and investigating the effects of pruning, a key characteristic of KANs. Moreover, following the advances presented in (Liu et al., 2024a), we aim to explore the integration of sparse activation mechanisms and shared spline modules into MonoKAN. These enhancements could significantly reduce memory usage and training time,

Method	COMPAS			Blog Feedback			Loan Defaulter		
	Params	Acc	Time (s)	Params	RMSE	Time (s)	Params	Acc	Time (s)
Constrained (Runje and Shankaranarayana, 2023)	2317	65.3% ± 0.0	31.80 ± 2.9	1101	0.156 ± 0.0	233.44 ± 15.1	577	65.0% ± 0.0	163.59 ± 2.6
Expressive (Nolte et al., 2022)	37	68.7% ± 0.0	101.87 ± 0.9	177	0.158 ± 0.0	396.34 ± 0.4	753	**65.3%** ± 0.0	91.7 ± 12.9
MonoKAN	4101	**69.0%** ± 0.0	48.96 ± 4.0	8546	**0.155**[1] ± 0.0	526.55 ± 177.0	1926	**65.3%**[1] ± 0.1	101.87 ± 0.9

Table 4
Comparison of MonoKAN with state-of-the-art certified partial monotonic MLPs. Each "Time" cell reports the mean and the standard deviation of training time per execution (in seconds).

making MonoKAN more scalable in high-dimensional settings without compromising its certified monotonicity guarantees. Additionally, we plan to explore whether the certified monotonicity constraints enforced by MonoKAN lead to more stable or interpretable results when evaluated with post-hoc attribution methods such as SHAP or Integrated Gradients, in order to further validate its utility in high-stakes, trust-sensitive applications. Finally, we acknowledge that in other domains, a more relaxed form of constraint (e.g., via an ε-tolerant monotonicity margin) may be desirable to balance interpretability with flexibility. To address such scenarios, we include as future work an extension of MonoKAN to support approximate monotonicity when full constraint enforcement is not strictly required.

CRediT authorship contribution statement

Jose Portela: Writing – review & editing, Validation, Supervision, Resources, Project administration, Methodology, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All datasets used in this research are available online.

Acknowledgments

This research was supported by funding from CDTI, with Grant Number MIG-20221006 associated with the ATMOSPHERE Project and grant PID2022-142024NB-I00 funded by MCIN/AEI/10.13039/501100011033.

A. Symbol Glossary for Theorem 3

To support the readability of Theorem 3, in this appendix we provide a detailed glossary of the mathematical symbols used throughout the monotonicity conditions. To this matter, Table 5 summarizes the key notation, including the indexing conventions for network layers, neurons, spline knots, and the associated weights and parameters. The definitions are consistent across all layers of the KAN architecture and are referenced throughout the theoretical results.

Symbol	Meaning
L	Total number of network layers
l	Layer index, $0 \leq l \leq L-1$
$I = [x^1, x^K]$	Spline definition interval
K	Number of spline knots in the interval of definition
n_l	Number of neurons in the l^{th} layer
n_{l+1}	Number of neurons in the $(l+1)^{th}$ layer
i	Neuron index in layer l, $1 \leq i \leq n_l$
j	Neuron index in layer $l+1$, $1 \leq j \leq n_{l+1}$
(l,i)	i^{th} neuron in the layer l
$(l+1,j)$	j^{th} neuron in the layer $l+1$
r	Index of input variable with imposed monotonicity
$\varphi_{l,j,i}(\cdot)$	Spline activation connecting the (l,i)-neuron and the $(l+1,j)$-neuron
$\mathbf{b}(\cdot)$	Base activation function (e.g. Sigmoid, Tanh, etc.)
$\omega^{\varphi}_{l,j,i}$	Weight on spline activation connecting the (l,i)-neuron and the $(l+1,j)$-neuron
$\omega^{\mathbf{b}}_{l,j,i}$	Weight on base activation function connecting the (l,i)-neuron and the $(l+1,j)$-neuron
k	Spline knot index, $1 \leq k \leq K$
$y^k_{l,j,i}$	Spline value at knot k
$d^k_{l,j,i}$	Knot difference: $(y^{k+1}_{l,j,i} - y^k_{l,j,i})/(x^{k+1}_{l,j,i} - x^k_{l,j,i})$
$m^k_{l,j,i}$	Spline slope at knot k
$\alpha^k_{l,j,i}$	Norm. slope: m^k/d^k
$\beta^k_{l,j,i}$	Norm. slope: m^{k+1}/d^{k+1}

Table 5
Glossary of symbols used in Theorem 3.

B. Proof of Theorem 3

This appendix presents a proof of Theorem 3 that states a sufficient condition for a Kolmogorov Arnold Network (KAN) to be partially monotonic. First of all, let us start by presenting a proposition that states that the composition of univariate monotonic function is also monotonic.

Proposition B.1. *Let $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \to \mathbb{R}$ be two continuous functions. Then*

1. *$g \circ f$ is increasingly monotonic if both f and g are increasingly monotonic.*

2. *$g \circ f$ is decreasingly monotonic if f is decreasingly monotonic and g is increasingly monotonic.*

Proof.

1. Assume f and g are increasingly monotonic. By definition, $\forall\ x_1, x_2 \in \mathbb{R}$ such that $x_1 \leq x_2$, we have $f(x_1) \leq f(x_2)$ and $g(y_1) \leq g(y_2)\ \forall\ y_1, y_2 \in \mathbb{R}$ with $y_1 \leq y_2$. Therefore, consider any $x_1, x_2 \in \mathbb{R}$ such that $x_1 \leq x_2$. Then,

   $$f(x_1) \leq f(x_2).$$

   Applying g to both sides, since g is increasing, we get

   $$g(f(x_1)) \leq g(f(x_2)).$$

   Thus, $g \circ f$ is increasingly monotonic.
2. Assume f is decreasingly monotonic and g is increasingly monotonic. By definition, $\forall\ x_1, x_2 \in \mathbb{R}$ with $x_1 \leq x_2$, we have $f(x_1) \geq f(x_2)$ and $g(y_1) \leq g(y_2)$ $\forall\ y_1, y_2 \in \mathbb{R}$ with $y_1 \leq y_2$. Consider any $x_1, x_2 \in \mathbb{R}$ such that $x_1 \leq x_2$. Then,

   $$f(x_1) \geq f(x_2).$$

   Applying g to both sides, since g is increasing, we get

   $$g(f(x_1)) \geq g(f(x_2)).$$

   Thus, $g \circ f$ is decreasingly monotonic.

□

Consequently, to obtain a KAN that is increasingly (resp. decreasingly) partially monotonic with respect to the r^{th} input, it is sufficient to ensure that the n_1 activations from the r^{th} input in the first layer are increasingly (decreasingly) monotonic and that for the rest of the nodes from the following layers, where the activation function outputs generated by the r^{th} input are considered part of the input, are also increasingly monotonic. Therefore, according to the above proposition, the KAN would be increasingly (decreasingly) partially monotonic. Considering this idea, it is obtained Theorem B.2 that gives a sufficient condition for a KAN to be partially monotonic.

Theorem B.2. *Given $f : \mathbb{R}^n \to \mathbb{R}$ a KAN with L layers and K knots in the interval of definition $I = [x^1, x^K]$, then if the basis function $\mathbf{b}$ is increasingly monotonic and the following conditions are met.*

1. $\omega^{\varphi}_{0,j,r} \geq 0, \omega^{\mathbf{b}}_{0,j,r} \geq 0$, *(resp. $\omega^{\varphi}_{0,j,r} \geq 0, \omega^{\mathbf{b}}_{0,j,r} \leq 0$)*
2. $y^{k+1}_{0,j,r} \geq y^{k}_{0,j,r}$, *i.e.* $d^{k}_{0,j,r} \geq 0$, *(resp. $y^{k+1}_{0,j,r} \leq y^{k}_{0,j,r}$, i.e. $d^{k}_{0,j,r} \leq 0$)*
3. *if* $d^{k}_{0,j,r} = 0,\ m^{k}_{0,j,r} = m^{k+1}_{0,j,r} = 0,$
4. *if* $d^{k}_{0,j,r} > 0,\ m^{k}_{0,j,r},\ m^{k+1}_{0,j,r} \geq 0$, *(resp. if* $d^{k}_{0,j,r} < 0,\ m^{k}_{0,j,r},\ m^{k+1}_{0,j,r} \leq 0$*)*
5. *if* $d^{k}_{0,j,r} > 0,\ \left(\alpha^{k}_{0,j,r}\right)^2 + \left(\beta^{k}_{0,j,r}\right)^2 \leq 9,$
6. $\omega^{\varphi}_{l,j,i} \geq 0, \omega^{\mathbf{b}}_{l,j,i} \geq 0,$
7. $y^{k+1}_{l,j,i} \geq y^{k}_{l,j,i}$, *i.e.* $d^{k}_{l,j,i} \geq 0,$
8. *if* $d^{k}_{l,j,i} = 0,\ m^{k}_{l,j,i} = m^{k+1}_{l,j,i} = 0,$
9. *if* $d^{k}_{l,j,i} > 0,\ m^{k}_{l,j,i},\ m^{k+1}_{l,j,i} \geq 0,$
10. *if* $d^{k}_{l,j,i} > 0,\ \left(\alpha^{k}_{l,j,i}\right)^2 + \left(\beta^{k}_{l,j,i}\right)^2 \leq 9,$

where $\alpha^{k}_{l,j,i} := \frac{m^{k}_{l,j,i}}{d^{k}_{l,j,i}}$, $\beta^{k}_{l,j,i} := \frac{m^{k+1}_{l,j,i}}{d^{k+1}_{l,j,i}}$ and $1 \leq l \leq L-1, \forall\ 1 \leq k \leq K-1, 1 \leq i \leq n^l, 1 \leq j \leq n^{l+1}$, then f is increasingly (resp. decreasingly) partially monotonic w.r.t the r^{th} input

Proof. Let us prove the theorem by induction over the number of layers of the KAN. Without loss of generality, we will consider the case of increasing monotonicity. The case for decreasing monotonicity is followed by analogous arguments.

Base Case ($n = 1$)

Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is a KAN with 1 layer such that the KAN's structure is $[n, 1]$. Therefore, by Eq. (6),

$$\hat{y} = f(\mathbf{x}_0) = \sum_{i=1}^{n_0} \left(\omega^{\varphi}_{0,1,i} \cdot \varphi_{0,1,i}(x_{0,i}) + \omega^{\mathbf{b}}_{0,1,i} \cdot \mathbf{b}(x_{0,i}) \right) + \theta_{0,1}.$$

Considering conditions (1) − (5) and Lemma 1 and Lemma 2, then it is clear that $\varphi_{0,1,r}$ is monotone. Additionally, the proposed linear extrapolation of the cubic Hermite spline, with slopes m^1 to the left of x^1 and m^K to the right of x^K, ensures that the spline $\varphi_{0,1,r}$ is C^1 continuous and monotonic across $\mathbb{R}$, not just within the data domain. Therefore, the linear combination of these monotonic functions, with positive coefficients, remains monotonic, and the composition of monotonic functions is also monotonic (by Proposition B.1). Thus, f is partially monotonic with respect to the r^{th} input across $\mathbb{R}^n$.

Induction Step. Suppose true the result for l layers and let us prove it for the $(l+1)^{th}$ layer.

Considering a KAN $f : \mathbb{R}^n \to \mathbb{R}$ with structure $[n, \ldots, n^l, n^{l+1} = 1]$. Then by Eq. (6),

$$\hat{y} = f(\mathbf{x}_0) = \sum_{i=1}^{n_l} \left(\omega^{\varphi}_{l,1,i} \cdot \varphi_{l,1,i}(x_{l,i}) + \omega^{\mathbf{b}}_{l,1,i} \cdot \mathbf{b}(x_{l,i}) \right) + \theta_{l,1}.$$

By conditions (6) − (10) and Lemma 1 and Lemma 2, $\varphi_{l,1,i}$ is monotone $\forall\ 1 \leq i \leq n^l$. Moreover, as $x_{l,i}$ is obtained from the input $\mathbf{x}_0$ as a KAN with structure $[n, \ldots, n^{l-1}, 1]$ which satisfies all the hypothesis of the Theorem, then, by the induction hypothesis, $x_{l,i}$ is partially monotone w.r.t. the r^{th} input. Therefore, considering Proposition B.1 and the same reasoning as in the base case, f is partially monotone w.r.t. the r^{th} input across $\mathbb{R}^n$. □

C. Dataset Descriptions

This appendix provides a detailed overview of the datasets used in our experiments. For each dataset, we specify

Table 6
Summary of datasets used in the experiments

Dataset	Task	Feature Dim.	Mono Features	Train/Validation/Test Size
COMPAS	Classification	13	4	3949 / 988 / 1235
Blog Feedback	Regression	276	8	37841 / 9461 / 6968
Loan Defaulter	Classification	28	5	334957 / 83740 / 70212
Auto MPG	Regression	8	3	250 / 79 / 80
Heart Disease	Classification	13	2	193 / 49 / 61

the task type, input dimensionality, number of monotonic features, and the size of the training and test sets.

COMPAS: The COMPAS dataset (Angwin et al., 2016) contains criminal history and demographic information for 6,172 individuals arrested in Florida. The objective is to predict the likelihood of recidivism within two years, based on 13 input features, which is modeled as a binary classification problem. To reflect ethical considerations, monotonicity is enforced with respect to four features that indicate criminal history: number of prior adult convictions, number of juvenile felonies, number of juvenile misdemeanors, and number of other convictions. Higher values for these variables should correspond to an increased predicted risk.

Blog Feedback: The Blog Feedback dataset (Buza, 2014) includes over 54,000 examples derived from blog posts, with the goal of predicting the number of comments a post will receive in the 24 hours following publication. Each sample consists of 276 features extracted from the post metadata and publication context. Eight of these features (A51, A52, A53, A54, A56, A57, A58, and A59) are expected to be monotonically non-decreasing with the target variable. Following common practice, we filter out extreme outliers by removing samples with targets above the 90th percentile to improve robustness to skewed error metrics such as MSE.

Loan Defaulter:[3] Based on publicly available lending data covering loans issued between 2007 and 2015, this dataset includes 488,909 entries with 28 input features. These features span credit history, income data, and loan status details. The task is to predict whether a borrower will default. Monotonicity constraints are imposed on a subset of features with strong financial interpretation: the model should predict increasing default probability with more public bankruptcies and higher debt-to-income ratio, and decreasing default probability with higher credit scores, longer employment history, and greater annual income.

Auto MPG:[4] This dataset includes fuel efficiency data for various car models. The prediction task is to estimate miles-per-gallon (MPG), framed as a regression problem. The input consists of 8 features, including engine and vehicle specifications. Monotonicity is enforced for three attributes: weight, horsepower, and displacement, all of which are assumed to be negatively correlated with MPG due to mechanical and energy-efficiency constraints. The dataset is sourced from the UCI Machine Learning Repository.

Heart Disease: [5] The Heart Disease dataset contains 303 patient records with 13 medical features. The task is to classify the presence or absence of heart disease. Based on medical rationale, monotonicity constraints are applied to two input variables: trestbps and cholesterol. Higher values for these features are assumed to increase the risk of heart disease, and the model is designed to reflect this domain knowledge

[3] https://www.kaggle.com/datasets/wordsforthewise/lending-club/code

[4] https://archive.ics.uci.edu/dataset/9/auto+mpg

[5] https://archive.ics.uci.edu/dataset/45/heart+disease

References

Allen, H.E., Latessa, E.J., Ponder, B.S., 2016. Corrections in America : an introduction. Pearson.

Angwin, J., Larson, J., Mattu, S., Kirchner, L., 2016. Machine Bias—There's Software Used across the Country to Predict Future Criminals. And It's Biased against Blacks. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing.

Aràndiga, F., Baeza, A., Yáñez, D.F., 2022. Monotone cubic spline interpolation for functions with a strong gradient. Applied Numerical Mathematics 172, 591–607. doi:10.1016/J.APNUM.2021.11.007.

Archer, N.P., Wang, S., 1993. Application of the Back Propagation Neural Network Algorithm with Monotonicity Constraints for Two-Group Classification Problems. Decision Sciences 24, 60–75. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1540-5915.1993.tb00462.xhttps://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-5915.1993.tb00462.xhttps://onlinelibrary.wiley.com/doi/10.1111/j.1540-5915.1993.tb00462.x, doi:10.1111/J.1540-5915.1993.TB00462.X.

Ben-David, A., Sterling, L., Pao, Y.H., 1989. Learning and classification of monotonic ordinal concepts. Computational Intelligence 5, 45–49. URL: https://onlinelibrary.wiley.com/doi/full/10.1111/j.1467-8640.1989.tb00314.xhttps://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.1989.tb00314.xhttps://onlinelibrary.wiley.com/doi/10.1111/j.1467-8640.1989.tb00314.x, doi:10.1111/J.1467-8640.1989.TB00314.X.

Bernstein, L., 2017. Who deserves a liver? Officials try to make organ transplants fairer. - The Washington Post. URL: https://www.washingtonpost.com/national/health-science/who-deserves-a-liver-officials-try-to-make-organ-transplants-fairer/2017/09/01/7e24b8e0-81fe-11e7-902a-2a9f2d808496_story.html.

Bubeck, S., 2015. Convex optimization : algorithms and complexity. Now Publishers Inc.

Buza, K., 2014. Feedback Prediction for Blogs. Studies in Classification, Data Analysis, and Knowledge Organization 47, 145–152. URL: https://link.springer.com/chapter/10.1007/978-3-319-01595-8_16, doi:10.1007/978-3-319-01595-8{_}16.

Cano, J.R., Gutiérrez, P.A., Krawczyk, B., Woźniak, M., García, S., 2019. Monotonic classification: An overview on algorithms, performance

measures and data sets. Neurocomputing 341, 168–182. doi:10.1016/J.NEUCOM.2019.02.024.

Chen, T., Guestrin, C., 2016. XGBoost: A Scalable Tree Boosting System. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining 13-17-August-2016, 785–794. URL: https://dl.acm.org/doi/pdf/10.1145/2939672.2939785, doi:10.1145/2939672.2939785.

Cohen, S.N., Snow, D., Szpruch, L., 2021. Black-Box Model Risk in Finance. SSRN Electronic Journal URL: https://papers.ssrn.com/abstract=3782412, doi:10.2139/SSRN.3782412.

Daniels, H., Velikova, M., 2010. Monotone and partially monotone neural networks. IEEE transactions on neural networks 21, 906–917. URL: https://pubmed.ncbi.nlm.nih.gov/20371402/, doi:10.1109/TNN.2010.2044803.

Fritsch, F.N., Carlson, R.E., 1980. Monotone piecewise cubic interpolation. SIAM Journal on Numerical Analysis 17, 238–246.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Gupta, A., Shukla, N., Marla, L., Kolbeinsson, A., Yellepeddi, K., 2019. How to Incorporate Monotonicity in Deep Networks While Preserving Flexibility? URL: https://arxiv.org/abs/1909.10662v3, doi:10.48550/arXiv.1909.10662.

Hall, C.A., Meyer, W.W., 1976. Optimal error bounds for cubic spline interpolation. Journal of Approximation Theory 16, 105–122. doi:10.1016/0021-9045(76)90040-X.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Processing Magazine 29, 82–97. doi:10.1109/MSP.2012.2205597.

Hunter, J.E., Schmidt, F.L., 1976. Critical analysis of the statistical and ethical implications of various definitions of test bias. Psychological Bulletin 83, 1053–1071. doi:10.1037/0033-2909.83.6.1053.

Iserson, K.V., Moskop, J.C., 2007. Triage in Medicine, Part I: Concept, History, and Types. Annals of Emergency Medicine 49, 275–281. URL: https://www.sciencedirect.com/science/article/pii/S0196064406007049, doi:10.1016/J.ANNEMERGMED.2006.05.019.

Köppen, M., 2002. On the Training of a Kolmogorov Network. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 2415 LNCS, 474–479. URL: https://link.springer.com/chapter/10.1007/3-540-46084-5_77, doi:10.1007/3-540-46084-5{_}77.

Kundu, A., Sarkar, A., Sadhu, A., 2024. KANQAS: Kolmogorov-Arnold Network for Quantum Architecture Search. EPJ Quantum Technology 11. URL: http://dx.doi.org/10.1140/epjqt/s40507-024-00289-z, doi:10.1140/epjqt/s40507-024-00289-z.

Lecun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 2015 521:7553 521, 436–444. URL: https://www.nature.com/articles/nature14539, doi:10.1038/nature14539.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F.E., 2017. A survey of deep neural network architectures and their applications. Neurocomputing 234, 11–26. URL: https://www.sciencedirect.com/science/article/pii/S0925231216315533, doi:10.1016/j.neucom.2016.12.038.

Liu, X., Han, X., Zhang, N., Liu, Q., 2020. Certified Monotonic Neural Networks. Advances in Neural Information Processing Systems 33, 15427–15438. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/b139aeda1c2914e3b579aafd3ceeb1bd-Paper.pdf, doi:10.48550/arXiv.2011.10219.

Liu, Z., Ma, P., Wang, Y., Matusik, W., Tegmark, M., 2024a. KAN 2.0: Kolmogorov-Arnold Networks Meet Science URL: https://arxiv.org/pdf/2408.10205.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljači´c, M.S., Hou, T.Y., Tegmark, M., 2024b. KAN: Kolmogorov-Arnold Networks URL: https://arxiv.org/abs/2404.19756v2.

Milani Fard, M., Canini, K., Cotter, A., Pfeifer, J., Gupta, M., 2016. Fast and Flexible Monotonic Functions with Ensembles of Lattices. Advances in Neural Information Processing Systems 29.

Monteiro, J., Ahmed, M.O., Hajimirsadeghi, H., Mori, G., 2022. Monotonicity regularization: Improved penalties and novel applications to disentangled representation learning and robust classification , 1381–1391URL: https://proceedings.mlr.press/v180/monteiro22a.html, doi:10.48550/arXiv.2205.08247.

Morala, P., Cifuentes, J.A., Lillo, R.E., Ucar, I., 2023. NN2Poly: A Polynomial Representation for Deep Feed-Forward Artificial Neural Networks. IEEE Transactions on Neural Networks and Learning Systems doi:10.1109/TNNLS.2023.3330328.

Nolte, N., Kitouni, O., Williams, M., 2022. Expressive Monotonic Neural Networks.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. Advances in Neural Information Processing Systems 32. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Pizarroso, J., Portela, J., Muñoz, A., 2022. NeuralSens: Sensitivity Analysis of Neural Networks. Journal of Statistical Software 102, 1–36. URL: https://www.jstatsoft.org/index.php/jss/article/view/v102i07, doi:10.18637/JSS.V102.I07.

Poeta, E., Giobergia, F., Pastor, E., Cerquitelli, T., Baralis, E., 2024. A Benchmarking Study of Kolmogorov-Arnold Networks on Tabular Data. 18th IEEE International Conference on Application of Information and Communication Technologies, AICT 2024 doi:10.1109/AICT61888.2024.10740444.

Polo-Molina, A., Alfaya, D., Portela, J., 2025. A Mathematical Certification for Positivity Conditions in Neural Networks With Applications to Partial Monotonicity and Trustworthy AI. IEEE Transactions on Neural Networks and Learning Systems , 1–16URL: https://ieeexplore.ieee.org/document/11203279, doi:10.1109/TNNLS.2025.3614431.

Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence 2019 1:5 1, 206–215. URL: https://www.nature.com/articles/s42256-019-0048-x, doi:10.1038/s42256-019-0048-x.

Runje, D., Shankaranarayana, S.M., 2023. Constrained Monotonic Neural Networks. Proceedings of the 40th International Conference on Machine Learning 202, 29338–29353. URL: https://proceedings.mlr.press/v202/runje23a.html, doi:10.5555/3737916.3740623.

Sarvamangala, D.R., Kulkarni, R.V., 2022. Convolutional neural networks in medical image understanding: a survey. Evolutionary Intelligence 15, 1–22. URL: https://link.springer.com/article/10.1007/s12065-020-00540-3, doi:10.1007/S12065-020-00540-3.

Sivaraman, A., Farnadi, G., Millstein, T., van den Broeck, G., 2020. Counterexample-Guided Learning of Monotonic Neural Networks. Advances in Neural Information Processing Systems 2020-December. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/8ab70731b1553f17c11a3bbc87e0b605-Paper.pdf, doi:10.48550/arXiv.2006.08852.

Sprecher, D.A., Draghici, S., 2002. Space-filling curves and Kolmogorov superposition-based neural networks. Neural Networks 15, 57–67. doi:10.1016/S0893-6080(01)00107-1.

Tjoa, E., Guan, C., 2021. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. IEEE Transactions on Neural Networks and Learning Systems 32, 4793–4813. doi:10.1109/TNNLS.2020.3027314.

Tran, V.D., Xuan Hieu Le, T., Tran, T.D., Luan Pham, H., Duong Le, V.T., Hai Vu, T., Nguyen, V.T., Nakashima, Y., 2024. Exploring the Limitations of Kolmogorov-Arnold Networks in Classification: Insights to Software Training and Hardware Implementation. Proceedings - 2024 12th International Symposium on Computing and Networking Workshops, CANDARW 2024 , 110–116URL: https://arxiv.org/pdf/2407.17790, doi:10.1109/CANDARW64572.2024.00026.

Voulodimos, A., Doulamis, N., Doulamis, A., Protopapadakis, E., 2018. Deep Learning for Computer Vision: A Brief Review. Computational Intelligence and Neuroscience 2018. doi:10.1155/2018/7068349.

Wang, S., Gupta, M., 2020. Deontological Ethics By Monotonicity Shape Constraints. Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics 108, 2043–2054. URL: https://proceedings.mlr.press/v108/wang20e.html.

Xu, K., Chen, L., Wang, S., 2024. Kolmogorov-Arnold Networks for Time Series: Bridging Predictive Power and Interpretability URL: https://arxiv.org/pdf/2406.02496.

Xu, X., Cao, D., Zhou, Y., Gao, J., 2020. Application of neural network algorithm in fault diagnosis of mechanical intelligence. Mechanical Systems and Signal Processing 141, 106625. doi:10.1016/J.YMSSP.2020.106625.

You, S., Ding, D., Canini, K., Pfeifer, J., Gupta, M.R., 2017. Deep Lattice Networks and Partial Monotonic Functions. Advances in Neural Information Processing Systems 2017-December, 2982–2990. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf, doi:10.48550/arXiv.1709.06680.

Zhang, Y., Tiňo, P., Leonardis, A., Tang, K., 2020. A Survey on Neural Network Interpretability. IEEE Transactions on Emerging Topics in Computational Intelligence 5, 726–742. URL: http://dx.doi.org/10.1109/TETCI.2021.3100641, doi:10.1109/TETCI.2021.3100641.