Softly Symbolifying Kolmogorov–Arnold Networks

James Bagrow[1,2,*] and Josh Bongard[3,2]

[1]Mathematics & Statistics, University of Vermont, Burlington, VT, United States
[2]Vermont Complex Systems Center, University of Vermont, Burlington, VT, United States
[3]Computer Science, University of Vermont, Burlington, VT, United States
[*]Corresponding author. Email: james.bagrow@uvm.edu, Homepage: bagrow.com

November 27, 2025

Abstract Kolmogorov-Arnold Networks (KANs) offer a promising path toward interpretable machine learning: their learnable activations can be studied individually, while collectively fitting complex data accurately. In practice, however, trained activations often lack *symbolic fidelity*, learning pathological decompositions with no meaningful correspondence to interpretable forms. We propose Softly Symbolified Kolmogorov-Arnold Networks (S2KAN), which integrate symbolic primitives directly into training. Each activation draws from a dictionary of symbolic and dense terms, with learnable gates that sparsify the representation. Crucially, this sparsification is differentiable, enabling end-to-end optimization, and is guided by a principled Minimum Description Length objective. When symbolic terms suffice, S2KAN discovers interpretable forms; when they do not, it gracefully degrades to dense splines. We demonstrate competitive or superior accuracy with substantially smaller models across symbolic benchmarks, dynamical systems forecasting, and real-world prediction tasks, and observe evidence of emergent self-sparsification even without regularization pressure.

***Keywords*—** differentiable sparsity, neuro-symbolic learning, symbolic regression, minimum description length, dynamical systems, interpretable neural networks, scientific machine learning

1 Introduction

Neural networks and deep learning are powerful but opaque tools for modeling complex systems [1, 2, 3, 4]. In scientific problems, accurate predictions are often not enough, and practitioners seek interpretable models that reveal underlying mechanisms, suggest novel hypotheses, or confirm existing theories [5, 6, 7]. Such interpretability can be achieved through parsimony, seeking smaller or less parameterized models [8, 9, 10, 11], or through the use of interpretable building blocks, components such as mathematical functions or symbolic expressions that can be interrogated and understood individually [12, 13, 14, 15]. These avenues are not mutually exclusive, and in fact the most interpretable models often combine both: sparse combinations of symbolic primitives that are simultaneously compact and semantically meaningful.

Kolmogorov–Arnold Networks (KANs) [16] have emerged as a promising alternative to traditional neural networks, replacing fixed activation functions with learnable univariate functions on each edge. Using flexible representations for the activation functions allows KANs to be highly performant [16, 17, 18]. But this design also lets practitioners examine the internal workings of the KAN by inspecting the learned activations, and possibly infer mathematical expressions (e.g., $\sin(x)$, e^x, x^2) for those activations [16, 19], although interpreting many such functions becomes challenging in large KANs. This combination of accuracy and interpretability makes KANs attractive for scientific machine learning [20, 21, 22].

However, the standard KAN reliance on flexible dense representations is at odds with interpretability. Splines, the typical choice of representation, can fit arbitrary shapes, but the resulting activations may bear no resemblance to recognizable mathematical functions. The standard remedy—post-hoc symbolification, where each trained activation is independently fitted to a symbolic form [16]—is problematic: splines may learn shapes difficult to fit symbolically, each activation is converted without considering network-level effects, and symbolic forms are not explored during training. The result: learned activations often lack *symbolic fidelity*: accurate predictions but no meaningful correspondence to interpretable expressions (Fig. 1). What is needed is a method that treats symbolic and dense representations on equal footing, automatically selects among them while pruning unnecessary components, and does so continuously during training—yielding compact, interpretable models that retain competitive accuracy.

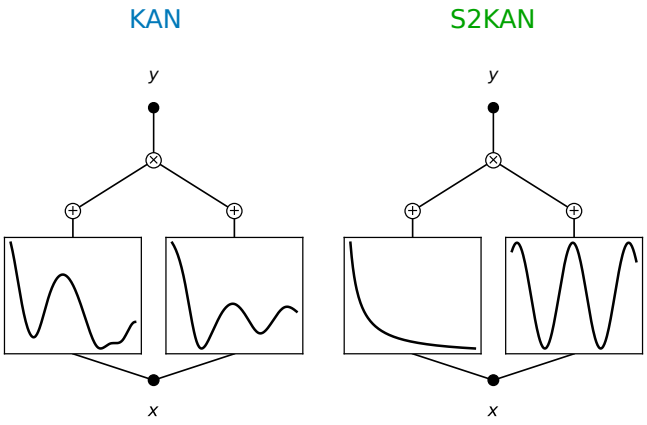

Figure 1: Learning $y = \mathrm{sinc}(x) = \sin(x)/x$ with a multiplicative Kolmogorov–Arnold network (KAN). The standard KAN lacks symbolic fidelity, exhibiting a numerically accurate but otherwise pathological decomposition, whereas S2KAN perfectly captures the underlying sinc function.

We propose **Softly Symbolified KANs (S2KAN)**, which integrate symbolic primitives and sparse bases directly into the training process through a differentiable gating mechanism. Our key contributions are:

- An activation function dictionary that combines dense representations, sparse basis functions, and symbolic primitives;
- A training process that symbolifies and sparsifies jointly with the rest of learning, with emergent self-sparsification even without regularization pressure;
- A Minimum Description Length objective that provides a principled accuracy–parsimony tradeoff.

Our method provides a best-of-both-worlds approach: when activation functions admit sparse symbolic representations, S2KAN selects interpretable basis functions; when they do not, the method gracefully degrades to dense representations, recovering standard KAN behavior. This ensures that symbolic structure is discovered when present, without sacrificing approximation quality when it is not. Rather than maximizing accuracy alone, S2KAN exposes the tradeoff between predictive performance and model complexity—often achieving competitive or even superior accuracy with far smaller models.

The rest of this paper is organized as follows. Section 2 describes Kolmogorov–Arnold networks and the standard symbolification method. Section 3 presents the Softly Symbolified KAN (S2KAN) method: the activation function dictionary, differentiable gating mechanism, and MDL-based loss function. Section 4 evaluates S2KAN on symbolic benchmarks, chaotic dynamical systems, and real-world prediction tasks. Section 5 examines the learning dynamics and self-sparsification behavior. We conclude with a discussion in Sec. 6.

2 Background

2.1 Kolmogorov–Arnold Networks

Motivated by the Kolmogorov–Arnold Representation Theorem [23, 24, 25], a KAN with L layers and shape $[n_0, n_1, \ldots, n_L]$ computes:

$$x_j^{(\ell+1)} = \sum_{i=1}^{n_\ell} \phi_{\ell ij}(x_i^{(\ell)}) \tag{1}$$

where $\phi_{\ell ij} : \mathbb{R} \to \mathbb{R}$ is a learnable univariate activation function on the edge from neuron i in layer ℓ to neuron j in layer $\ell + 1$.

The $\phi_{\ell ij}$ can be parameterized in many ways, including radial basis functions [26], Fourier series [27] or sinusoidal functions [28], wavelets [29], and Chebyshev polynomials [30]. The original KAN formulation [16, 19] used a combination of a B-spline and a fixed base function $b(x)$ (typically SiLU):

$$\phi(x) = w_b b(x) + w_s \mathrm{spline}(x), \tag{2}$$

where w_b and w_s are learnable scale parameters, and

$$\mathrm{spline}(x) = \sum_{m=1}^{G+K} c_m B_m(x), \tag{3}$$

where B_m are B-spline basis functions of order K over G grid intervals, and c_m are learnable coefficients. Because the B-splines are polynomials, the nonpolynomial base function $b(x)$ ensures that composition through many layers of the KAN does not just result in a high-order polynomial.

When using B-splines, KANs also implement grid updates, where the spline knots are periodically adjusted to better cover the activation function's input range, and grid refinement, where the number of knots is increased during training; see [16] for details. KAN 2.0 [19] introduces multiplication nodes (Fig. 1), which compute products of incoming activations rather than sums, enabling more compact representations of multiplicative functions.

2.2 Post-hoc symbolification

Standard KANs rely exclusively on dense representations and do not discover symbolic forms during training. To find symbolic representations, KANs symbolify after training by fitting each learned activation $\phi_{\ell ij}$ to candidate symbolic functions from a library $\mathcal{S} = \{1, x, x^2, 1/x, \sin, \cos, \exp, \log, \ldots\}$. For each candidate $S \in \mathcal{S}$, the method solves the separable optimization:

$$a^*, b^*, c^*, d^* = \arg\min_{a,b,c,d} \sum_x \left[\phi_{\ell ij}(x) - (cS(ax + b) + d)\right]^2, \tag{4}$$

where the sum is over preactivation values. Fitted candidates are ranked by a weighted sum of complexity (via user-defined scores) and R^2; the top-ranked function is selected if its R^2 exceeds a threshold. For full details, see [16, 19].

This approach has limitations: (1) the learned activation (base + spline) may be difficult to fit symbolically, (2) each function is converted independently without considering network-level effects, and (3) symbolic forms are never explored during training. The first limitation is particularly problematic: there is no *a priori* reason why the learned activation functions should map cleanly to symbolic forms. Indeed, in practice one often observes degenerate or pathological decompositions that, while making accurate predictions, lack what we call *symbolic fidelity*.

3 S2KAN: softly symbolifying KANs

Instead of learning activations and converting them post hoc, we propose learning activation functions as combinations of the standard KAN components and other symbolic terms. Depending on the choice of terms, the activation functions can form an overcomplete dictionary of functions (Sec. 3.1) so sparsification will be necessary to avoid overfitting and stabilize training. For a single activation function in isolation, this would be naturally approached via ℓ_1 regularization (e.g., LASSO), but in KANs activation functions are *composed across layers*, requiring end-to-end differentiability through the selection mechanism. We therefore introduce binary gating variables that select which terms are active, made differentiable via a continuous relaxation of the ℓ_0 norm (Sec. 3.2), with selection guided by a Minimum Description Length objective (Sec. 3.3).

3.1 Activation function dictionary

We organize the activation function using three categories of terms (or atoms):

$$\phi(x) = \sum_{s\in\mathcal{S}} z_s c_s \psi_s(x) + \sum_{f\in\mathcal{F}} z_f c_f \psi_f(x) + \sum_{r\in\mathcal{R}} z_r \phi_r(x;\mathbf{c}_r), \tag{5}$$

where the $z \in \{0,1\}$ are binary gates selecting active terms (Sec. 3.2), $c \in \mathbb{R}$ are learnable coefficients, and each $\psi : \mathbb{R} \to \mathbb{R}$ is a candidate univariate function. Here $\mathcal{S}$ indexes the symbolic library, $\mathcal{F}$ indexes sparse function bases, and $\mathcal{R}$ indexes dense parameterized representations. Equation (5) allows researchers to design problem-specific combinations as needed. Using the standard KAN B-spline dense representation for $\mathcal{R}$ and setting $\mathcal{S} = \mathcal{F} = \emptyset$, Eq. (5) recovers the original KAN formulation.

In this work, we consider the following dictionary:

Symbolic library ($\mathcal{S}$) A collection of elementary functions such as $1, x, \sin(x), 1/x, \log|x|, \ldots$. Each function receives an independent gate and coefficient.

Sparse function bases ($\mathcal{F}$) Families of orthogonal or structured functions indexed by degree/frequency:

- *Chebyshev polynomials:* $T_p(x)$ for degree $p = 0, 1, \ldots, P$, computed via recurrence.
- *Fourier basis:* $\{\sin(qx), \cos(qx)\}$ for mode $q = 1, \ldots, Q$.

Each term receives an independent gate and coefficient.

Dense representations ($\mathcal{R}$) A single gate controls an entire parameterized function class:

- *B-spline with SiLU residual:* $\phi_{\text{spline}}(x;\mathbf{c}) = c_0 \text{SiLU}(x) + \sum_{b=1}^{B} c_b B_b(x)$, where the B_b are B-spline basis functions on a fixed knot sequence and $\mathbf{c} = (c_0, c_1, \ldots, c_B)$.

This formulation allows for graceful degradation: when symbolic terms are insufficient—whether because the underlying function resists symbolic description or because the architecture lacks capacity to express it—the method naturally falls back to the dense spline representation.

3.2 Differentiable sparsity via Hard Concrete Distribution

We follow the ℓ_0 regularization approach of [11], which leverages the reparameterization trick to provide a continuous, differentiable relaxation of binary gates while maintaining the ability to produce exact zeros and ones.

For each gate i, introduce a learnable parameter $\alpha_i \in \mathbb{R}$ and define a stochastic binary variable via the Concrete (Gumbel-Softmax) distribution with temperature $\tau > 0$:

$$s_i = \sigma\left(\frac{\log u - \log(1-u) + \alpha_i}{\tau}\right), \quad u \sim \text{Uniform}(0,1), \tag{6}$$

where $\sigma(\cdot)$ is the sigmoid function. This provides a continuous relaxation in $(0,1)$, but cannot produce exact boundary values. To assign finite probability mass to exactly 0 and 1, apply a stretched and rectified transformation with parameters $\gamma < 0 < \zeta$ (typically $\gamma = -0.1, \zeta = 1.1$):

$$\bar{s}_i = s_i(\zeta - \gamma) + \gamma, \tag{7}$$

$$\tilde{z}_i = \min(1, \max(0, \bar{s}_i)). \tag{8}$$

The stretch extends the range to (γ, ζ) before clipping to $[0,1]$, allowing the relaxed gate to reach the boundaries exactly during training.

The expected value of this gate, or gate probability, used for regularization (Sec. 3.3), has a closed form:

$$\mathbb{E}[\tilde{z}_i] = \sigma\left(\alpha_i - \tau \log \frac{-\gamma}{\zeta}\right). \tag{9}$$

During training, sample $\{z_i\}$ and learn updates to their probabilities jointly when updating other parameters. At inference, deterministically threshold: $z_i = \mathbb{1}_{\{\mathbb{E}[\tilde{z}_i]>1/2\}}$.

The gradient of the expected ℓ_0 penalty with respect to gate parameters is

$$\frac{\partial \mathbb{E}[\tilde{z}_i]}{\partial \alpha_i} = \mathbb{E}[\tilde{z}_i]\,(1 - \mathbb{E}[\tilde{z}_i])\,. \tag{10}$$

This gradient is maximal when $\mathbb{E}[\tilde{z}_i] = 0.5$ (maximum uncertainty) and vanishes as gates commit to 0 or 1, providing stable optimization.

Remarks An activation function sparsifies as its gates close, and the entire function can be omitted when all its gates close, allowing for architectural sparsification. Beyond the mechanism of sparsity, these gates offer additional benefits. Initializing α_i acts as a prior, and one can tune the model towards or away from symbolic entries by, for instance, setting $\alpha_i \approx -1$ for dense terms and $\alpha_i \approx 0$ otherwise. Gate probabilities $p_i = \mathbb{E}[\tilde{z}_i]$ also provide a natural convergence criterion: training can terminate early, possibly with some patience, once gates become decisive (e.g., when most $p_i < 0.01$ or $p_i > 0.99$).

3.3 Loss Function via Minimum Description Length

To balance model accuracy while minimizing complexity, our training objective is motivated by the Minimum Description Length (MDL) principle, which seeks to minimize the total bits required to encode both the model and the data given the model:

$$\mathcal{L}_{\text{MDL}} = \mathcal{L}_{\text{model}} + \mathcal{L}_{\text{data|model}}. \tag{11}$$

The first term captures model complexity while the second encodes residuals. Assuming iid residuals $\epsilon_t = y_t - \hat{y}_t \sim \mathcal{N}(0, \sigma^2)$, the data encoding term, $\mathcal{L}_{\text{data|model}}$, is proportional to the negative log-likelihood:

$$\mathcal{L}_{\text{data|model}} \propto \frac{n}{2} \log \sigma^2 + \frac{1}{2\sigma^2} \sum_{t=1}^{n} (y_t - \hat{y}_t)^2. \tag{12}$$

Since $\sum_{t=1}^{n}(y_t - \hat{y}_t)^2 = n\text{MSE}(y, \hat{y})$ and setting $\sigma^2 = \text{MSE}(y, \hat{y})$ (empirical variance), minimizing the mean squared error implicitly minimizes this encoding term. For $\mathcal{L}_{\text{model}}$, the model complexity term, let

$$k := \sum_{\ell ij} \sum_{m} \mathbb{E}[\tilde{z}_{\ell ijm}] \tag{13}$$

be the expected number of active terms over the activation functions in the network, where $\mathbb{E}[z]$ is given by Eq. (9). Under a BIC-style approximation [8], the model description length is $\mathcal{L}_{\text{model}} = \frac{k}{2} \log n$. Putting both terms together yields the per-sample training objective

$$\mathcal{L} = \text{MSE}(y, \hat{y}) + \beta \frac{k \log n}{2n}, \tag{14}$$

where hyperparameter $\beta \geq 0$ controls the sparsity-accuracy tradeoff.

3.3.1 Complexity-weighted sparsity

Standard MDL treats all parameters uniformly. However, when selecting among terms with varying structural complexity, we may sometimes wish to account for differential encoding costs, which we can do by incorporating *complexity weights*. For term ψ_m with complexity weight $w_m > 0$, the weighted number of active terms is $k_w := \sum_{\ell ij} \sum_m w_m \mathbb{E}[\tilde{z}_{\ell ijm}]$ which replaces k in Eq. (14). The gradient $\partial k_w / \partial \alpha_m = w_m \mathbb{E}[\tilde{z}_m](1 - \mathbb{E}[\tilde{z}_m])$ shows that higher complexity weights will induce stronger regularization pressure on the corresponding gates. Appropriate choices for w_m include uniform weighting ($w_m = 1$, recovering standard BIC), encoding cost ($w_m = 1 + \log_2(d+1)$ for degree/order d, reflecting bits to specify the function), or computational cost ($w_m \propto$ FLOPs required to evaluate ψ_m). Hand-tuning w_m is also common practice in symbolic regression. In this work, we do not pursue weighting schemes other than traditional MDL, but it may be beneficial to consider them in the future.

We discuss further setup and training details in Methods (Sec. A).

4 Results

We start by evaluating S2KAN on a toy example, predicting the function $y = \text{sinc}(x) = \sin(x)/x$. We generate 1024 training and 256 testing points and use a single-layer one-multiplication-unit KAN trained with 2000 epochs and a batch size of 32. For S2KAN we provide a symbolic library containing the reciprocal function $1/x$, as well as $P = 6$ and $Q = 4$ for the Chebyshev and Fourier bases, respectively. Other training details are as per Methods (Sec. A).

Figure 1 shows the non-symbolic decomposition typical for KANs. Baseline KAN, while fitting the data well (test MSE $< 10^{-5}$), lacks *symbolic fidelity*. In contrast, S2KAN, equipped with the reciprocal function, perfectly discovers the multiplicative decomposition of the sinc function.

Next, we turn to the Nguyen symbolic regression benchmark [31]. This benchmark contains a sample of mathematical functions designed to capture a variety of symbolic regression challenges. We focus in Table 1 on the first 10, the last two of which are bivariate. For each problem we generate 1024 training and 256 testing points. We apply three different architectures, a 'small' architecture with no hidden units, a 'large' architecture with one hidden layer of 3 summation units, and a 'large-mult' architecture with one hidden layer of 3 summation and 1 multiplication units. Models were trained for 10k epochs with a batch size of 128. S2KAN's first 200 epochs were warmup ($\beta = 0$). We report the best of 3 seeds. For the baseline, we report the accuracy (test R^2) for the original model and after symbolification at threshold 0.5 and 0.95 (Sec. 2.2). For S2KAN we used three values of β to study different levels of regularization, and for each we report test R^2 and the % of symbolic terms in the final model.

In almost all cases, the baseline model accurately represents the function (one exception is the small architecture for problem F10, which lacks the expressive power to capture the true function). The same or nearly the same predictive performance is observed across the S2KAN architectures (the one exception is the larger architectures at $\beta = 10$ for F5).

However, the performance comparison changes considerably when we consider the symbolified baseline models. Even at the stricter threshold of 0.95, rarely does the baseline model retain a good fit to the data; only for the small architectures for problems F1–F5 does the baseline reliably recover a predictive symbolic form. In contrast, in nearly all cases the S2KAN models are already symbolified, with small architectures achieving 100% symbolic terms in all cases except F10.

Unlike the baseline model, there is essentially no tradeoff between numeric accuracy and symbolicity in S2KAN.

4.1 Data-driven modeling of dynamical systems

We evaluate S2KAN on two dynamical systems that present challenging forecasting problems due to their chaotic attractors.

The first is the *Ikeda map* [32, 33], a discrete-time chaotic system arising from nonlinear optics that resists discovery by sparse regression methods like SINDy [10]:

$$\begin{aligned} x_{n+1} &= 1 + \mu\left(x_n \cos\left(\phi_n\right) - y_n \sin\left(\phi_n\right)\right), \\ y_{n+1} &= \mu\left(x_n \sin(\phi_n) + y_n \cos(\phi_n)\right), \end{aligned} \tag{15}$$

where $\phi_n = 0.4 - 6\left(1 + x_n^2 + y_n^2\right)^{-1}$ and bifurcation parameter $\mu = 0.9$. KANs have been shown to model the Ikeda map effectively [17, 18].

The second system is a continuous-time *three-species ecosystem*:

$$\begin{aligned} \frac{dN}{dt} &= N\left(1 - \frac{N}{K}\right) - x_p y_p \frac{NP}{N + N_0}, \\ \frac{dP}{dt} &= x_p P\left(y_p \frac{N}{N + N_0} - 1\right) - x_q y_q \frac{PQ}{P + P_0}, \\ \frac{dQ}{dt} &= x_q Q\left(y_q \frac{P}{P + P_0} - 1\right), \end{aligned} \tag{16}$$

where N, P, and Q represent primary producer, herbivore, and carnivore populations, with carrying capacity K serving as the bifurcation parameter. We set $K = 0.98$, $x_p = 0.4$, $y_p = 2.009$, $x_q = 0.08$, $y_q = 2.876$, $N_0 = 0.16129$, and $P_0 = 0.5$ to produce chaotic dynamics [34].

Data for both systems were generated and split into training and testing as per Panahi et al. [17]. To model these systems we use a [2, 4,4,4, 2] architecture for the Ikeda map and a [3, 3,3, 3] architecture for the ecosystem. The same architectures were used in prior work [18]. All models were trained with a batch size of 128. We equipped S2KAN with $\mathcal{S} = \{\sqrt{x}, 1/(1+x^2)\}$ (Ikeda), $S = \{1, x, x^2, 1/(1+x)\}$ (ecosystem), and $P = Q = 4$ (both). We report our results in Table 2 and Figs. 2 and 3.

On the Ikeda map, the baseline model showed a noticeable performance advantage over S2KAN in 1-step prediction and a slight disadvantage in multi-step prediction. However, we also see that the baseline model is far larger, containing 48 activation functions and 720 total parameters, compared to 32 functions and 122 parameters—the baseline model is nearly six times larger. With this capacity difference in mind, the improvement in multi-step performance is notable given the more parsimonious model. We see good multi-step forecasting, with a similar accuracy horizon for both models (Fig. 2).

On the ecosystem, we find that 1-step prediction is quite poor but multi-step prediction is reasonable, and the learned S2KAN is very small compared to the baseline, only 5% of the parameters. When we examine the trajectories (Fig. 3), however, we notice a problem: the S2KAN model has been crushed so heavily it collapses to a fixed point. This motivated investigating smaller values of β, eventually reaching $\beta = 0$. This model performed very well, with a long accuracy horizon and nearly half the multi-step error of the baseline model. Surprisingly, this model, which has no specific regularization pressure, still sparsified, and it outperformed the baseline with 13% fewer parameters. Given the relative performance at 1-step and multi-step prediction, it is likely that the baseline is overfitting the derivative while S2KAN is better capturing the underlying attractor.

Given this result on the ecosystem, we performed an unregularized experiment on the Ikeda map, setting $\beta = 0$ (Table 2 and Fig. 2). In this case, the unregularized model performed slightly worse than the regularized model, and did not reach full gate convergence even with the longer training time, but slightly outperformed the baseline.

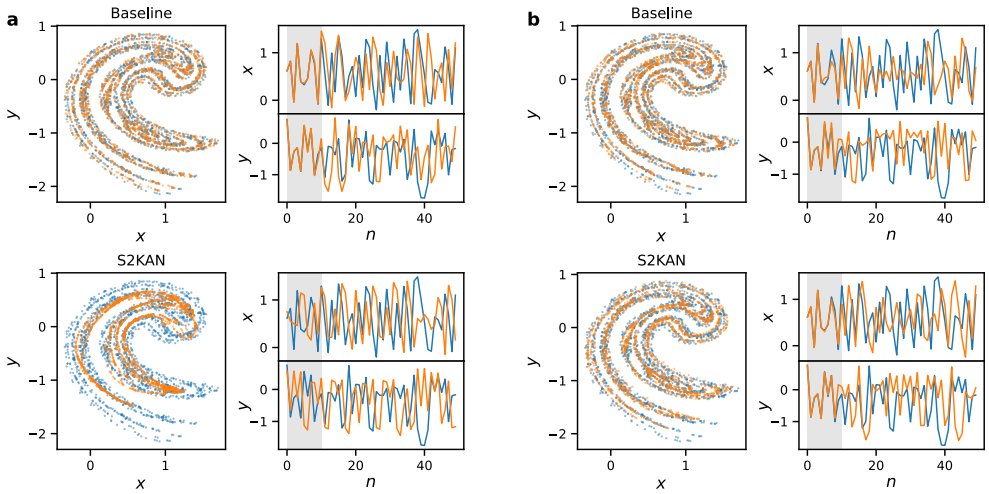

Figure 2: Dynamical system modeling for the Ikeda map. (a) $\beta = 0$, 20k epochs. (b) $\beta = 0.1$, 4k epochs. The regularized S2KAN accurately forecasts the dynamics from the initial condition as long as the baseline (shaded region).

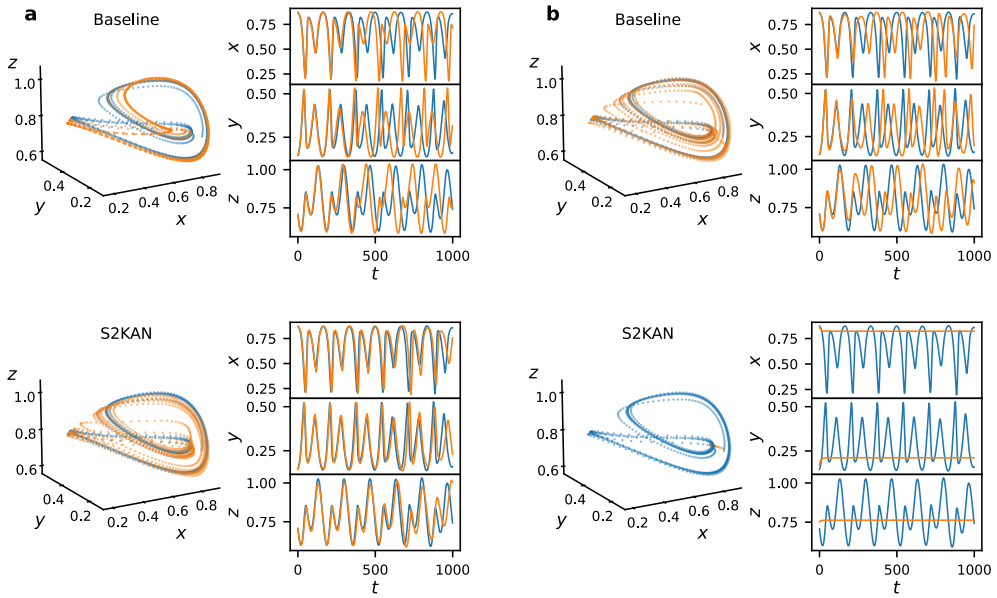

Figure 3: Dynamical system modeling for the ecosystem. (a) $\beta = 0$, 15k epochs. (b) $\beta = 0.1$, 10k epochs. The unregularized S2KAN captures the multi-step dynamics well.

Table 1: Nguyen benchmarks across architectures. Shapes: S = $[n_0, 1]$, L = $[n_0, 3, 1]$, LM = $[n_0, (3, 1), 1]$.

ID	Expression		Baseline KAN R^2			S2KAN R^2			S2KAN % Symbolic		
			—	t=0.5	t=0.95	β=0.1	β=1	β=10	0.1	1	10
F1	$x^3 + x^2 + x$	S	1.0000	-0.10	1.00	1.0000	1.0000	0.9981	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.12	0.16	1.0000	0.9997	0.9990	100	83	33
		LM	1.0000	-0.16	-0.16	1.0000	0.9998	0.9982	78	33	11
F2	$x^4 + x^3 + x^2 + x$	S	1.0000	-0.25	1.00	1.0000	0.9999	0.9913	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.08	0.66	1.0000	1.0000	0.9999	100	83	50
		LM	1.0000	-0.28	0.62	1.0000	0.9999	0.9998	89	44	33
F3	$x^5 + x^4 + x^3 + x^2 + x$	S	1.0000	-0.16	1.00	1.0000	1.0000	0.9943	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.14	0.49	1.0000	1.0000	0.9991	100	67	50
		LM	1.0000	0.07	0.60	1.0000	0.9999	0.9984	78	33	33
F4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	S	1.0000	-0.23	1.00	1.0000	1.0000	0.9856	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.19	1.00	1.0000	1.0000	0.9998	100	83	50
		LM	1.0000	-0.23	0.07	1.0000	1.0000	0.9986	100	67	33
F5	$\sin(x^2)\cos(x) - 1$	S	1.0000	1.00	1.00	0.9998	0.9980	0.9721	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.56	-1.20	0.9999	0.9997	0.0037	33	0	0
		LM	1.0000	-21.62	-6.51	0.9999	0.9997	0.0039	22	22	0
F6	$\sin(x) + \sin(x + x^2)$	S	1.0000	-0.05	-0.05	0.9998	0.9994	0.9968	100	100	100
	$x \in [-1, 1]$	L	1.0000	-0.05	0.68	1.0000	0.9997	0.9732	67	33	17
		LM	1.0000	-0.04	0.47	0.9999	0.9998	0.9918	44	44	11
F7	$\log(x + 1) + \log(x^2 + 1)$	S	1.0000	-2.68	-2.68	1.0000	1.0000	0.9998	100	100	100
	$x \in [0, 2]$	L	1.0000	-1.27	-1.27	1.0000	0.9999	0.9997	100	50	0
		LM	1.0000	-2.68	-2.38	1.0000	1.0000	0.9997	78	44	0
F8	$\sqrt{x}$	S	1.0000	-8.43	-8.43	0.9990	0.9987	0.9660	100	100	100
	$x \in [0, 4]$	L	1.0000	-7.81	-3.83	0.9998	0.9989	0.9534	83	50	67
		LM	1.0000	-4.45	-2.00	0.9998	0.9953	0.9839	78	22	0
F9	$\sin(x) + \sin(y^2)$	S	1.0000	-0.33	-0.33	0.9999	0.9988	0.9989	100	100	100
	$x, y \in [-1, 1]$	L	1.0000	-0.37	-0.37	0.9999	0.9987	0.9966	100	67	22
		LM	1.0000	-0.23	-0.19	0.9999	0.9992	0.9968	50	14	7
F10	$2\sin(x)\cos(y)$	S	-0.0008	-0.00	-0.00	-0.0006	-0.0006	0.0039	50	0	0
	$x, y \in [-\pi, \pi]$	L	1.0000	-0.09	-0.37	0.9998	0.9997	0.9968	100	100	56
		LM	1.0000	-0.09	0.56	0.9999	0.9999	0.9988	86	21	21

Table 2: Comparison of Baseline and S2KAN models on dynamical systems prediction tasks.

	Epochs	Shape	Model	β	RMSE 1-step	RMSE multi-	Active funcs.	k
Ikeda map	4 000	[2, 4, 4, 4, 2]	Baseline	–	0.0052	0.8711	48	720
			S2KAN	0.1	0.0196	0.8385	32	122.4
	20 000		Baseline	–	0.0028	0.8674	48	720
			S2KAN	0.0	0.1052	0.8067	43	829.5
Ecosystem	10 000	[3, 3, 3, 3]	Baseline	–	0.0003	0.2028	27	405
			S2KAN	0.1	0.0667	0.1640	12	21
	15 000		Baseline	–	0.0002	0.1835	27	405
			S2KAN	0.0	0.0007	0.1108	20	352.1

We explore model self-sparsification further in Sec. 5.

4.2 Real-world data

We apply S2KAN to two real-world datasets:

Concrete compressive strength Predict the compressive

strength (in MPa) of concrete based on sample properties. Dataset contains 1030 samples with eight features: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate (all in kg/m^3), and age (days). Compressive strength in concrete is well known to depend on the water-to-cement ratio [35], so we include this as a derived feature. We also compute total binder (cement plus slag and fly ash), total aggregate (coarse plus fine), and the water-to-binder ratio, which generalizes the water-to-cement relationship when supplementary cementitious materials are present. We include $\log(\text{age}+1)$ and $\sqrt{\text{age}}$; strength gain in concrete follows an approximately logarithmic relationship with curing time [35], motivating the inclusion of transformed age variables. For modeling, we used an 80/20 train/test split. Data were collected by Yeh [36, 37].

Superconductor critical temperature Predict critical temperature (in K) of superconductors based on their material properties. The original dataset includes many features and derived statistics; we focus on five that capture composition, electronic structure, and bonding: number of elements, weighted mean valence, valence entropy, weighted mean first ionization energy, and mean electron affinity. These same representative features were used in prior KAN modeling [18]. For modeling, we sampled 1000 train and 1000 test points. The data come from Japan's National Institute for Materials Science superconductor database [38, 39].

Our results are summarized in Table 3.

In both datasets S2KAN found considerably more compact representations, and for concrete the fitted functions contained no spline terms. The concrete models are significantly more compressed than the baseline model, with a small tradeoff in accuracy: for $\beta = 0.1$, S2KAN achieved test $R^2 = 0.91$ vs. 0.92 for baseline, but with only 102 of 448 functions active. The overall performance is quite good, approaching known state of the art ($R^2 = 0.93$, [40]). For higher $\beta = 0.5$, far more compression was achieved (29/448 functions) with predictive performance dropping to $R^2 = 0.86$. Tuning β allows experimenters to study the tradeoff in accuracy and parsimony.

For superconductivity, S2KAN outperformed baseline KAN (test $R^2 = 0.70$–0.72 vs. 0.62), however it did so with a heavier reliance on splines than for the concrete data. This may underscore the challenge of expressing the physics governing superconducting behavior as simple symbolic functions of bulk material properties. Interestingly, the more regularized and smaller model, $\beta = 0.5$, showed the best predictive performance, suggesting that symbolic representations provide useful inductive bias even when they cannot fully capture the underlying physics.

5 Learning dynamics and self-sparsification

To understand how S2KAN discovers sparse representations during training, we performed an experiment to track gate statistics throughout optimization. For each gate with learnable parameter α_i, the expected gate value $p := \mathbb{E}[\tilde{z}]$ (Eq. (9)) represents the probability the gate is open. We compute the total binary entropy across all gates in the network:

$$H = -\sum_m \left[p_m \log_2 p_m + (1 - p_m) \log_2 (1 - p_m) \right], \quad (17)$$

which measures the total uncertainty in the gate configuration. When all gates are fully decided ($p_m \approx 0$ or $p_m \approx 1$), entropy is near zero; when gates are maximally uncertain ($p_m \approx 0.5$), entropy is maximized. We also report *decisiveness*, the fraction of gates with $p_m < 0.01$ or $p_m > 0.99$, indicating convergence to discrete selection.

Figure 4 shows these statistics for shallow and deep architectures trained on the superconductor dataset. For this experiment, we used a symbolic library $\mathcal{S} = \{1, x, x^2, x^3, \sqrt{x}, \log(x+1), \exp(x)\}$, no Chebyshev terms ($P = 0$), and Fourier modes $Q = 2$.

During warmup ($\beta = 0$), gates remain undecided; once regularization begins, entropy rises then drops as gates commit to on/off states, with symbolic and spline gate probabilities separating as the network selects its preferred representation. This demonstrates that the differentiable sparsification process naturally encompasses an exploration–exploitation tradeoff. Notice (Fig. 4, left top) the shallower network quickly opens up the spline gates before closing them off once warmup ends and regularization pressure begins. The deeper network, in comparison, does not rely on the spline terms to the same extent (Fig. 4, left bottom). In both networks, the spline gates reach final state earlier than the gates for other terms. In the deeper network, there is also a slight tendency for the later layers to decide more quickly than the earlier layers (Fig. 4, right), although the difference is fairly small. In the shallow network, both layers decide at about the same rate.

We also explored the self-sparsification phenomenon observed in Sec. 4.1 and Table 2. During those unregularized training runs, we tracked the number of active terms. Examining in Fig. 5 their evolution over training, we again see a clear exploration–exploitation tradeoff with k rising before dropping, particularly for the ecosystem network (which performed well under the $\beta = 0$ condition). Even with no regularization pressure, the model, when focused only on minimizing error, will self-sparsify.

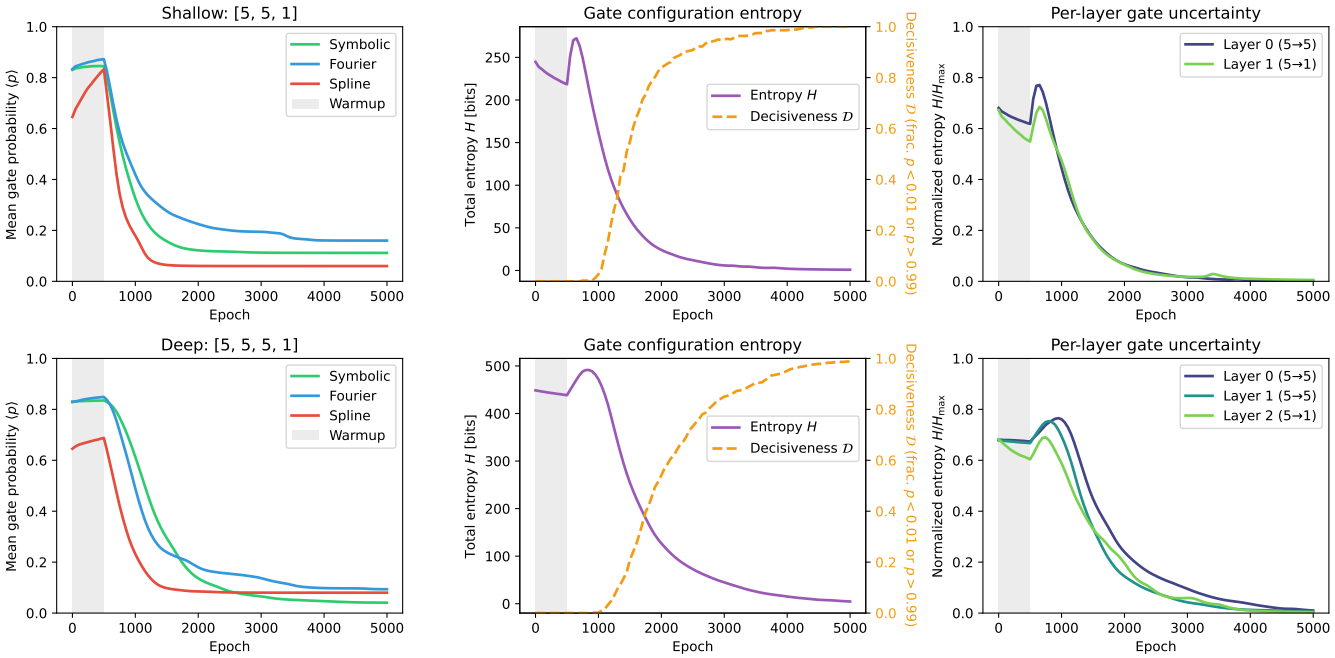

Figure 4: Learning dynamics for S2KAN on the superconductor dataset, comparing shallow $[5, 5, 1]$ (top) and deep $[5, 5, 5, 1]$ (bottom) architectures with $\beta = 0.5$. Gray shading indicates the warmup period ($\beta = 0$).

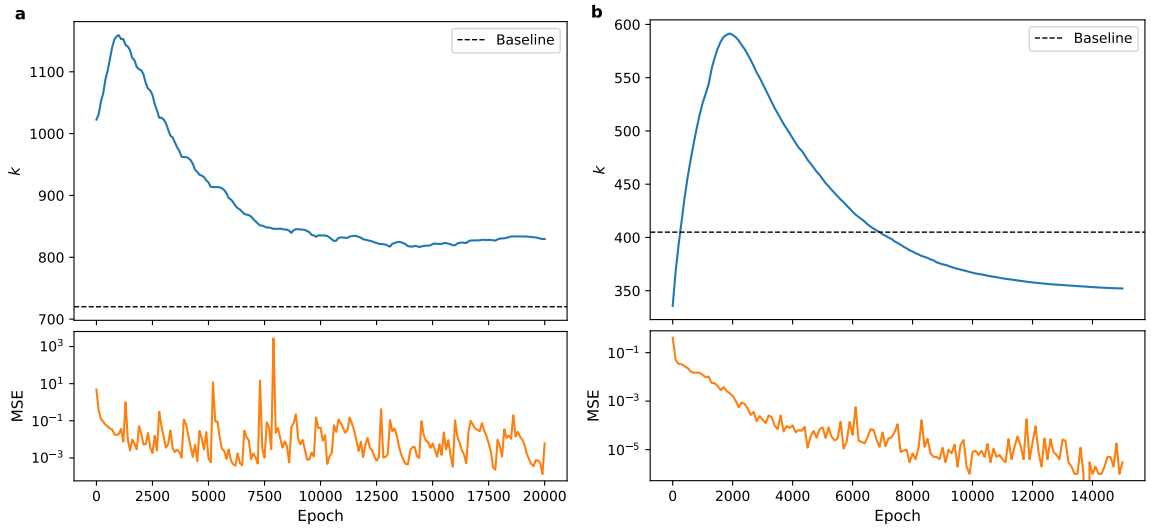

Figure 5: Self-sparsification of the Ikeda map (a) and ecosystem networks (b). Horizontal dashed lines indicate the number of active terms in the baseline KAN. Training MSEs are presented for reference.

Table 3: Performance comparison of baseline KAN and S2KAN at different regularization strengths on real-world datasets.

Dataset	N obs.	Features	Model	Shape	β	R^2	RMSE	Active	k	% Symb.
Concrete	824/206	13 (8+5)	Baseline	[13, 32, 1]	—	0.924	4.43 MPa	448	6720	0%
			S2KAN	[13, 32, 1]	0.1	0.909	4.84 MPa	102	134	100%
			S2KAN	[13, 32, 1]	0.5	0.864	5.93 MPa	29	33	100%
Supercond.	1000/1000	5 (of 81)	Baseline	[5, 5, 1]	—	0.621	21.13 K	30	450	0%
			S2KAN	[5, 5, 1]	0.1	0.703	18.70 K	25	240	18.9%
			S2KAN	[5, 5, 1]	0.5	0.724	18.02 K	17	76	40.8%

6 Discussion

Our experiments demonstrate that S2KAN achieves comparable or superior accuracy with substantially smaller, more interpretable models. The method's combination of differentiable gating, MDL-based selection, and graceful degradation enables symbolic discovery without sacrificing flexibility. We now discuss limitations and directions for future work.

One potential criticism of S2KAN is that it offers no particular guidance on how to choose the activation function dictionary. This limitation, however, is inherent to symbolic regression itself: methods such as PySR [14], SINDy [10], and AI Feynman [13] all require the user to specify candidate functions. Domain knowledge typically guides this choice—physical systems, for instance, suggest trigonometric functions, rational functions, or exponentials depending on the phenomena involved. Where S2KAN differs is in its graceful degradation: if the dictionary lacks the necessary symbolic terms to capture the underlying function, the method falls back to dense representations rather than failing outright. A suboptimal library thus incurs a cost in interpretability, not accuracy.

The symbolic library presents its own practical challenges. Terms such as $1/x$, $\log(x)$, or $\exp(x)$ can diverge for certain inputs, destabilizing training. Likewise, $\log(x)$ and $\sqrt{x}$ have restricted domains that may be violated during training. In this light, the standard KAN's exclusive reliance on dense representations is an unsung strength: splines and similar bases are smooth, bounded, and well-behaved across arbitrary input domains, avoiding the pathologies that symbolic terms can introduce. Yet these challenges are not unique to S2KAN: the problem is endemic to symbolic regression methods including neural network approaches like equation learner (EQL) networks [41, 42, 43]. Practical mitigations include input normalization, domain restriction, careful initialization, and replacing problematic terms with protected variants such as $\sqrt{|x|}$ or $\log(|x| + \epsilon)$. The EQL-div [42] and iEQL [43] variants offer a more principled solution, introducing symbolic primitives with learnable cutoffs that prevent divergences and provide smooth out-of-domain behavior; similar modifications could be incorporated into S2KAN. Beyond content, dictionary size also matters: larger dictionaries increase training cost proportionally, though inference cost depends only on selected terms—a heavily sparsified S2KAN may be faster than baseline if it eliminates spline evaluations in favor of simple symbolic primitives.

S2KAN takes an explicit approach to symbolic fidelity, incorporating symbolic primitives directly into the activation function dictionary. However, the training instabilities discussed above—divergences, domain violations, the need for protected operators—motivate an alternative: biasing dense representations implicitly toward forms with high symbolic fidelity. Rather than embedding symbolic functions explicitly, implicit approaches modify the training process or architecture to encourage learned activations that, while remaining dense, are more amenable to post-hoc symbolic fitting, effectively a symbolic inductive bias. Projective KAN [44] exemplifies this direction, using projection-based regularization to bias activations toward higher symbolic fidelity. Confining ill-behaved terms to the regularization rather than the main computational graph generally makes them easier to isolate and control. Combining explicit symbolic terms with implicit biases toward symbolic fidelity is a promising direction: such hybrid approaches may capture the benefits of both, offering guaranteed symbolic output when appropriate and greater stability when symbolic terms are ill-suited.

We have presented S2KAN, a method that integrates symbolic primitives directly into Kolmogorov–Arnold networks through differentiable gating and principled, MDL-based selection. By placing symbolic and dense representations on equal footing, S2KAN discovers interpretable structure when it is present while retaining the flexibility of standard KANs when it is not. Our experiments demonstrate that this approach yields compact, symbolic models with competitive or superior accuracy across symbolic benchmarks, chaotic dynamical systems, and real-world prediction tasks. As scientific machine learning increasingly demands models that are not only accurate but also interpretable, methods like S2KAN offer a path toward neural networks whose internal structure can be interrogated, understood, and trusted.

A Methods

S2KAN was implemented in PyTorch v2.8.0 [45]. Optimization was performed using Adam [46] with default pa-

rameters and a constant learning rate of 10^{-3} for both coefficients and gates. Non-spline coefficients were initialized from $U(-0.05, 0.05)$; spline coefficients were initialized as in [16]. For B-splines, we used $G = 10$ grid intervals and degree $K = 3$ for all experiments. Unless otherwise noted, spline grids were updated 10 times during the first 50 epochs; grid refinement was not used. Chebyshev polynomials are defined on $[-1, 1]$, so activation function domains are tracked per activation and updated alongside spline grid updates; inputs are rescaled to this domain before computing the Chebyshev basis. Fourier terms use natural frequencies $\sin(qx), \cos(qx)$ without domain rescaling.

For the Hard Concrete distribution, we used temperature $\tau = 2/3$ and stretch parameters $\gamma = -0.1$, $\zeta = 1.1$ throughout. Unless otherwise noted, gates were initialized with $\alpha_i \sim \mathcal{N}(0, 0.1)$ except for spline gates which used $\alpha_i = -1$, providing a slight bias toward symbolic representations at initialization. Neither temperature annealing nor β scheduling was used.

Unless otherwise noted, models were trained with a batch size of 128 and a warmup period of 200 epochs with $\beta = 0$. Early stopping terminated training when gate decisiveness exceeded 0.99, with patience $\min(500, 0.05 \times \#\,\text{epochs})$. For all experiments, baseline KAN used B-splines only with gates fixed open and no regularization ($\beta = 0$).

Code will be made available upon publication.

Sinc function (Fig. 1)— We generated 1024 training and 256 test points on $x \in [1, 15]$, with target $y = \sin(x)/x$. Both models used a single multiplication unit with no hidden layer, shape $[1, (0, 1)]$, trained for 2000 epochs with batch size 32. S2KAN used symbolic library $\mathcal{S} = \{1/x\}$, Chebyshev degree $P = 6$, Fourier modes $Q = 4$, regularization $\beta = 1.0$, and 100 warmup epochs. Early stopping was not used.

Nguyen benchmark (Table 1)— We evaluated the first 10 Nguyen problems [31], generating 1024 training and 256 test points per problem. Models were trained for 10k epochs and we report the best of 3 seeds. Three architectures were tested: small (no hidden layer), large (one hidden layer with 3 summation units), and large-mult (3 summation plus 1 multiplication unit). S2KAN used symbolic library $\mathcal{S} = \{1, x, x^2, \sin(x), \cos(x)\}$, $P = 11$, and $Q = 6$. Post-hoc symbolification of baseline KANs was applied at R^2 thresholds of 0.5 and 0.95.

Dynamical systems (Table 2, Figs. 2–3)— For the Ikeda map, we used architecture $[2, 4, 4, 4, 2]$ with symbolic library $\mathcal{S} = \{\sqrt{x}, 1/(1 + x^2)\}$, Chebyshev degree $P = 4$, and Fourier modes $Q = 4$. For the ecosystem, we used architecture $[3, 3, 3, 3]$ with symbolic library $\mathcal{S} = \{1, x, x^2, 1/(1 + x)\}$ and the same basis parameters. Spline grid updates and early stopping were not used. Training epochs varied by condition (see Table 2).

Real-world data (Table 3, Fig. 4)— For concrete compressive strength, we used the UCI concrete dataset [36, 37] with 8 raw features augmented by 5 derived features (water-cement ratio, water-binder ratio, total binder, total aggregate, log age), for 13 total. We used an 80/20 train-test split and tested architectures $[13, 32, 1]$ and $[13, 32, 16, 1]$. For superconductor critical temperature prediction, we used 5 features from the UCI superconductor dataset [38, 39]: number of elements, weighted mean valence, weighted mean first ionization energy, mean electron affinity, and valence entropy. We sampled 1000 training and 1000 test points and tested architectures $[5, 5, 1]$ and $[5, 5, 5, 1]$ (these architectures were previously used in [18]). Both tasks used symbolic library $\mathcal{S} = \{1, x, x^2, \sqrt{x}, \log(x + 1)\}$ (superconductor additionally included x^3 and $\exp(x)$), Fourier modes $Q = 2$, no Chebyshev basis, and were trained for 5000 epochs with batch size 64 and 500 warmup epochs. Early stopping was not used.

References

[1] Zachary C. Lipton. The mythos of model interpretability. *Commun. ACM*, 61(10):36–43, September 2018. ISSN 0001-0782. 1

[2] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018. doi: 10.1109/ACCESS.2018.2870052. 1

[3] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1 (5):206–215, 2019. 1

[4] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018. 1

[5] Carl G Hempel et al. *Aspects of scientific explanation*, volume 965. Free press New York, 1970. 1

[6] James Woodward. *Making things happen: A theory of causal explanation*. Oxford university press, 2005. 1

[7] Galit Shmueli. To explain or to predict? *Statistical science*, pages 289–310, 2010. 1

[8] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 00905364, 21688966. 1, 4

[9] Trevor Hastie, Robert Tibshirani, and Martin Wainwright. Statistical learning with sparsity. *Monographs on statistics and applied probability*, 143(143):8, 2015. 1

[10] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems.

Proceedings of the National Academy of Sciences, 113 (15):3932–3937, 2016. 1, 5, 10

[11] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, 2018. 1, 3

[12] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324 (5923):81–85, 2009. 1

[13] Silviu-Marian Udrescu and Max Tegmark. AI Feynman: A physics-inspired method for symbolic regression. *Science advances*, 6(16):eaay2631, 2020. 1, 10

[14] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020. 1, 10

[15] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio Olivetti de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021. 1

[16] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljacic, Thomas Y. Hou, and Max Tegmark. KAN: Kolmogorov–Arnold networks. In *The Thirteenth International Conference on Learning Representations*, 2025. 1, 2, 3, 11

[17] Shirin Panahi, Mohammadamin Moradi, Erik M. Bollt, and Ying-Cheng Lai. Data-driven model discovery with Kolmogorov–Arnold networks. *Phys. Rev. Res.*, 7:023037, Apr 2025. 1, 5

[18] James Bagrow and Josh Bongard. Multi-exit kolmogorov–arnold networks: enhancing accuracy and parsimony. *Machine Learning: Science and Technology*, 6(3):035037, aug 2025. 1, 5, 8, 11

[19] Ziming Liu, Pingchuan Ma, Yixuan Wang, Wojciech Matusik, and Max Tegmark. KAN 2.0: Kolmogorov–Arnold Networks meet science. *arXiv preprint arXiv:2408.10205*, 2024. 1, 2, 3

[20] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, Laurent Daudet, Maria Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborová. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91 (4):045002, 2019. 1

[21] Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng-Wei Qiu, et al. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4), 2021. 1

[22] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3 (6):422–440, 2021. 1

[23] Andreĭ Nikolaevich Kolmogorov. *On the representation of continuous functions of several variables by superpositions of continuous functions of a smaller number of variables*. American Mathematical Society, 1961. 2

[24] Vladimir I Arnold. On functions of three variables. *Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965*, pages 5–8, 2009. 2

[25] Andrei Nikolaevich Kolmogorov. On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition. In *Dokl. Akad. Nauk USSR*, volume 114, pages 953–956, 1957. 2

[26] Ziyao Li. Kolmogorov–Arnold networks are Radial Basis Function networks. *arXiv preprint arXiv:2405.06721*, 2024. 2

[27] GistNoesis. FourierKAN. https://github.com/GistNoesis/FourierKAN, 2024. Accessed: 2025-07-07. 2

[28] Eric Reinhardt, Dinesh Ramakrishnan, and Sergei Gleyzer. SineKAN: Kolmogorov–Arnold networks using sinusoidal activation functions. *Frontiers in Artificial Intelligence*, 7, 2025. ISSN 2624-8212. doi: 10.3389/frai.2024.1462952. 2

[29] Zavareh Bozorgasl and Hao Chen. Wav-KAN: Wavelet Kolmogorov–Arnold networks. *arXiv preprint arXiv:2405.12832*, 2024. 2

[30] SS Sidharth, AR Keerthana, R Gokul, and KP Anas. Chebyshev polynomial-based Kolmogorov–Arnold networks: An efficient architecture for nonlinear function approximation. *arXiv preprint arXiv:2405.07200*, 2024. 2

[31] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, R. I. McKay, and Edgar Galván-López. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119, 2011. 5, 11

[32] Kensuke Ikeda. Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system. *Optics communications*, 30(2):257–261, 1979. 5

[33] SM Hammel, CKRT Jones, and Jerome V Moloney. Global dynamical behavior of the optical field in a ring cavity. *Journal of the Optical Society of America B*, 2 (4):552–564, 1985. 5

[34] Kevin McCann and Peter Yodzis. Nonlinear dynamics and population disappearances. *The American Naturalist*, 144(5):873–879, 1994. 5

[35] Adam M. Neville. *Properties of Concrete*. Pearson, 5th edition, 2011. 8

[36] I.-C. Yeh. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998. ISSN 0008-8846. 8, 11

[37] I-Cheng Yeh. Analysis of strength of concrete using design of experiments and neural networks. *Journal of Materials in Civil Engineering*, 18(4):597–604, 2006. 8, 11

[38] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018. 8, 11

[39] Center for Basic Research on Materials. MDR SuperCon datasheet ver.240322. 8, 11

[40] Gaoyang Liu and Bochao Sun. Concrete compressive strength prediction using an explainable boosting machine model. *Case Studies in Construction Materials*, 18:e01845, 2023. ISSN 2214-5095. 8

[41] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016. 10

[42] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4442–4450. PMLR, 10–15 Jul 2018. 10

[43] Matthias Werner, Andrej Junginger, Philipp Hennig, and Georg Martius. Informed equation learning. *arXiv preprint arXiv:2105.06331*, 2021. 10

[44] Alastair Poole, Stig McArthur, and Saravan Kumar. Projective kolmogorov arnold neural networks (p-kans): Entropy-driven functional space discovery for interpretable machine learning. *arXiv preprint arXiv:2509.20049*, 2025. 10

[45] A Paszke. PyTorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019. 10

[46] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 10