KANs need curvature: penalties for compositional smoothness

James Bagrow[1,2,*]

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

Abstract Kolmogorov–Arnold networks (KANs) offer a potent combination of accuracy and interpretability, thanks to their compositions of learnable univariate activation functions. However, the activations of well-fitting KANs tend to exhibit pathologically high-curvature oscillations, making them difficult to interpret, and standard regularization penalties do not prevent this. Here we derive a basis-agnostic curvature penalty and show that penalized models can maintain accuracy while achieving substantially smoother activations. Accounting for how function composition shapes curvature, we prove an upper bound on the full model's curvature relative to the curvature penalty, and use this to motivate richer forms of penalties. Scientific machine learning is increasingly bottlenecked by the trade-off between accuracy and interpretability. Results such as ours that improve interpretability without sacrificing accuracy will further strengthen KANs as a practical tool for both prediction and insight.

1 Introduction

Kolmogorov–Arnold networks (KANs) [1] are growing in popularity as an alternative to traditional neural networks. KANs replace fixed nonlinearities with learnable univariate activations placed on each edge. This makes them effective in two ways [2]: they are highly accurate and more interpretable than standard deep networks. When both accuracy and interpretability matter, such as in scientific machine learning, KANs are a natural choice [3].

In practice, KANs often fit data well but develop high-curvature, kink-like oscillations in their activation functions [4, 5] (Fig. 1). This is problematic because interpretable activations must be readable, that is, smooth.[1] As Liu *et al.* remark, "deeper [more than two layers] representations may bring the advantages of smoother activations" [1, §2.3]. Yet KART [6, 7], the representation theorem underpinning KANs, offers no smoothness guarantee [8]. Much recent work has explored KAN architectures [9, 10, 11, 12, 13], training methods [14, 15, 16], and downstream applications [17, 18, 19], but the question of how to enforce smoothness in KAN activations remains open.

To address this gap, we make the following contributions:

1. We show that the standard KAN penalty structurally cannot suppress activation curvature and so cannot guarantee smooth activations (Sec. 2).
2. We derive a closed-form edge-wise curvature penalty that is basis-agnostic (Sec. 3), and demonstrate empirically that it yields substantially smoother KANs at similar accuracy, across function approximation, the Feynman benchmark, and overparameterized regimes (Sec. 4).
3. We prove through a compositional analysis that the edge-wise penalty upper-bounds the curvature of the full network (Sec. 5), and use this result to motivate richer penalties that draw on more than edge-wise information (Sec. 6).

[1]"Smooth" in the sense of low bending energy (roughness), not continuous differentiability.

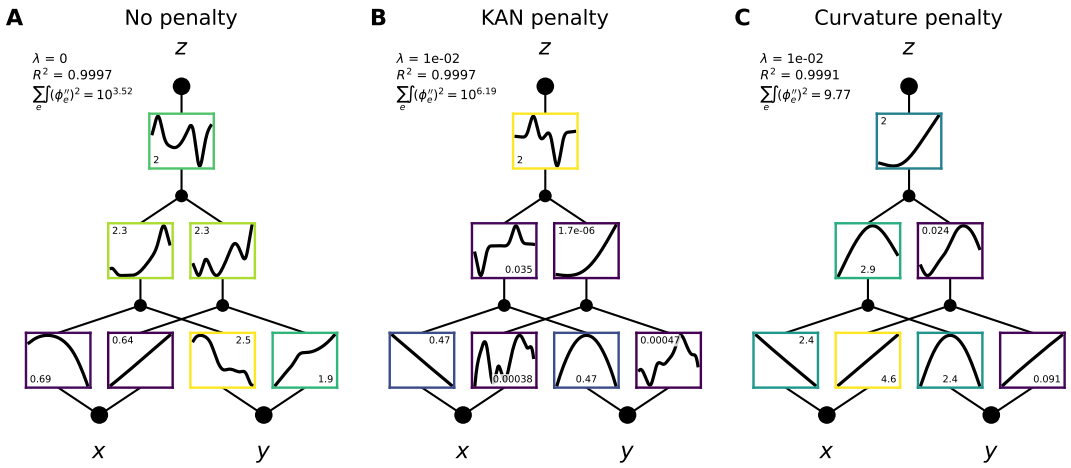

Figure 1: Trained networks on $\sin\left(x + y^2\right)$ over $[-2, 2]^2$ with grid size $G = 10$. Despite accurate fits, the unpenalized (**A**) and KAN-penalized (**B**) activations exhibit high-curvature, kink-like oscillations, very unlike the true function's components in appearance. The curvature-penalized model (**C**) presents smooth activation functions more closely aligned with the true function's x, y^2, and $\sin(\cdot)$ components. Plot frame color and label indicate the operating range ($\max_z \phi_e - \min_z \phi_e$) of each activation.

Section 7 positions our work relative to prior P-spline, KAN-curvature, and input-Hessian literature, and Sec. 8 concludes.

2 KAN's penalty can't enforce smoothness

A KAN of depth L and widths $[n_0, n_1, \ldots, n_L]$ maps an input $x \in \mathbb{R}^{n_0}$ to an output $f(x) \in \mathbb{R}^{n_L}$ via the layer-wise composition

$$z_c^{(\ell)} \;=\; \sum_{b=1}^{n_{\ell-1}} \phi_{cb}^{(\ell)}\left(z_b^{(\ell-1)}\right), \qquad (1)$$

for $\ell = 1, \ldots, L$ and $c = 1, \ldots, n_\ell$, where $z^{(0)} = x$, $f(x) = z^{(L)}$, and each edge $b \to c$ at layer ℓ carries its own learnable univariate activation function $\phi_{cb}^{(\ell)} : \mathbb{R} \to \mathbb{R}$. The original and most popular KAN form [1] implements activations with a base (residual) function, typically SiLU, and B-splines:

$$\phi(x) \;=\; \alpha\ \mathrm{SiLU}(x) \;+\; \beta\, c^\top B(x), \qquad (2)$$

where $B(x) = (B_1(x), \ldots, B_G(x))^\top$ are the B-spline basis functions on a fixed (uniform, we assume) knot grid of size G, $\alpha, \beta \in \mathbb{R}$ are scalar weights, and $c \in \mathbb{R}^G$ is the learnable spline coefficient vector. SiLU is the canonical choice for the base following [1] but any smooth substitute carries through our curvature analysis; non-B-spline bases such as Gaussian-RBF KANs are treated in App. B.

KANs are trained by minimizing $J(f) + \lambda R(f)$, where J is a data-fitting loss (mean squared error throughout this paper) and R is a penalty with strength $\lambda \geq 0$. Liu *et al.* [1] propose the **KAN penalty**,

$$R_{\mathrm{KAN}}(f) \;=\; \mu_1 \sum_e \left|\phi_e\right|_1 \;+\; \mu_2 \sum_\ell S\left(\rho^{(\ell)}\right), \qquad (3)$$

where the first sum runs over all edges of the network, $|\phi|_1 := \frac{1}{N}\sum_{s=1}^{N} |\phi(z^{(s)})|$ is an activation's average magnitude over N training samples, $S(\rho^{(\ell)}) := -\sum_{b,c} \rho_{cb}^{(\ell)} \log \rho_{cb}^{(\ell)}$ is the entropy of the within-layer magnitude distribution, and $\rho_{cb}^{(\ell)} := |\phi_{cb}^{(\ell)}|_1 / \sum_{b',c'} |\phi_{c'b'}^{(\ell)}|_1$ is the normalized magnitude of edge $b \to c$ at layer ℓ. The first term shrinks activation magnitudes while the second concentrates each layer's mass onto a few high-magnitude edges.

Figure 1 shows a KAN trained with no penalty (A) and with the penalty of Eq. 3 (B) on $f(x, y) = \sin(x + y^2)$. Both have high test accuracy yet many high-

curvature, oscillatory activation functions. Why?

In fact, it is structurally impossible for Eq. 3 to smooth out activations. To see this, observe that both terms in R_{KAN} depend only on the average magnitudes $|\phi_{cb}^{(\ell)}|_1$, which carry no derivative information about ϕ. A wildly oscillating ϕ and a smooth ϕ with the same average magnitude will incur the same penalty. The KAN penalty governs *which* edges carry magnitude, not how it varies across their support.

What is needed instead, and which is well known in the study of splines as *P-splines* [20], is to penalize the curvature of ϕ directly, as we discuss in the next section.

3 Activation function curvature

A natural choice of curvature penalty on a sufficiently smooth function g is its L^2 bending energy $\int |\nabla^2 g(x)|_F^2 \, dx$, the squared H^2 Sobolev seminorm.[2] For a univariate g this becomes $\int (g''(z))^2 \, dz$. We compute this term for each KAN edge activation ϕ_e on its spline support $\Omega_e \subset \mathbb{R}$, then sum over edges to obtain the curvature penalty $R(f)$, the focus of this paper. (A penalty of this form has been applied to KANs [5], with crucial differences; see Sec. 7.) This edge-wise penalty is related to the full curvature of f in Sec. 5.

Substituting the KAN edge form (Eq. 2) and integrating gives

$$\int_{\Omega_e} \big(\phi_e''(z)\big)^2 \, dz \;=\; \beta_e^2 \int (s_e'')^2 \, dz + \alpha_e^2 \int (u'')^2 \, dz + 2\,\beta_e\,\alpha_e \int s_e''\,u'' \, dz, \quad (4)$$

where $s_e(z) := c_e^\top B(z)$ is the spline component and $u(z) := \mathrm{SiLU}(z)$ is the fixed base. We treat the three terms as follows:

1. The Eilers–Marx P-spline reduction [20] gives $\int_{\Omega_e} (s_e''(z))^2 \, dz \;\approx\; h_e^{-3} \, \|D_2 c_e\|^2$ on uniform-knot cubic B-splines, where D_2 is the second-difference matrix on the G spline coefficients and $h_e = |\Omega_e|/G$ is the knot spacing. We drop the h_e^{-3} prefactor to prevent high-G grids from dominating the base term.

2. The integral is the constant $K_{\text{silu}} := \int (u'')^2 \, dz = (30 + \pi^2)/90 \approx 0.443$; because u is fixed, the resulting $\alpha_e^2 K_{\text{silu}}$ term acts as L^2 shrinkage on α_e.

3. The cross term is dropped. Its magnitude is bounded by the sum of squared terms and is small in practice because u'' is localized near $z = 0$, but more importantly, keeping it may let the optimizer reduce the penalty by anti-correlating s_e'' and u'' rather than minimizing curvature.

Combining (1)–(3) and summing over edges, with the scalar β_e folded inside the norm, yields our proposed **curvature penalty**:

$$R(f) \;=\; \sum_e \Big(\|D_2(\beta_e\, c_e)\|^2 + K_{\text{silu}}\, \alpha_e^2 \Big). \quad (5)$$

Remark. The proposed penalty has several nice properties: It works entirely on model coefficients and does not couple to training data. It has affine functions as its null space. It admits a clean Bayesian interpretation as a Gaussian prior on second differences. When applied to B-splines, the term $\|D_2 c_e\|^2$ is the central object of study in the field of penalized splines (P-splines) [21, 22], a rich field with a 30-year history that to the best of our knowledge has gone untouched by the KAN community.

Remark. The standard PyKAN package also implements a first-difference penalty on spline coefficients alongside Eq. 3 (disabled by default). But this also does not work well as a curvature penalty, instead serving as a sparsity prior that concentrates curvature on a small number of knots. Its null space is constants, not affine functions. This makes it more tailored to reward piecewise-constant functions than low-curvature ones.

Remark. More generally, we can apply our curvature penalty to any fixed basis whose second derivatives are square-integrable: $\int_{\Omega_e} (s_e''(z))^2 \, dz = c_e^\top M_e \, c_e$ with curvature Gram matrix $(M_e)_{ij} =$

[2] Throughout, "curvature" refers to bending energy $\int (\phi'')^2 \, dz$, whose integrand $(\phi'')^2$ is the small-slope linearization of squared differential-geometric curvature $(\phi'')^2/(1+\phi'^2)^3$. Bending energy is the standard surrogate elsewhere.

$\int B_i''(z)\, B_j''(z)\, dz$. We explore a non-B-spline basis for ϕ in App. B.

Remark. The curvature penalty works edge-wise. It is not a curvature penalty for the full function $f(x)$ learned by the KAN because it does not capture the composition of activation functions across layers. We return to this in Sec. 5 and show that the edge-wise penalty is an upper bound on the curvature of the full composition.

4 Lower curvature models with similar accuracy

Here we compare the curvature penalty to no-penalty and KAN-penalty conditions for several tasks commonly addressed by KAN models. Experimental details are given in Methods (App. A).

4.1 KANs for function approximation

Comparing results across two dimensions, test accuracy (RMSE or R^2) and total edge curvature $\sum_e \int (\phi_e''(z))^2\, dz$, we find that curvature-penalized KANs achieve substantially smoother activation functions at the same or nearly the same fit quality. For instance, in Fig. 1, all KANs fit the target function very well, but the no-penalty and KAN-penalty models do so with solutions that have orders of magnitude more curvature than the curvature-penalized model. Inspecting the individual ϕ_e in the plot, the curvature-penalized KAN most closely resembles the components of the target function, suggesting that the curvature penalty can aid in interpretability and downstream tasks such as symbolic regression.

Figure 2 compares no-penalty and curvature-penalized KANs across a sweep of penalty strengths, for the target $f(x, y) = \exp\left(\sin(\pi x) + y^2\right)$ studied by Liu *et al.* [1]. The unpenalized model fits very well, with test RMSE $< 10^{-3}$, but the penalized KAN is nearly as accurate over a wide range of λ values. Even with the relatively low grid size ($G = 10$), which itself should act as a smoothness prior, the penalized model achieves total edge curvatures one-third or less that of the unpenalized model.

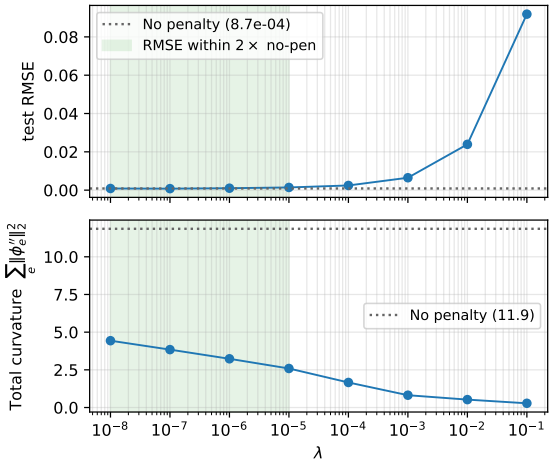

Figure 2: The penalty selects for a smoother basin among same-fit minima. Approximating $f(x, y) = \exp\left(\sin(\pi x) + y^2\right)$ with architecture $[2, 5, 1]$ and $G = 10$. The green shaded region: λ range where penalized test RMSE is within 2× the unpenalized baseline (dashed).

This result—reduced curvature with only a small cost to accuracy—holds broadly. In Table 1 we study the effects of penalties for KANs approximating popular functions taken from the Feynman Equation symbolic regression benchmark. Again, curvature-penalized KANs are the smoothest models, and are the most accurate (3 of 14 equations) or within no more than a factor of two error of the most accurate model (10 of 14 equations).

4.2 Curvature penalty stabilizes overparameterized KANs

When G is large, the KAN activations are over-parameterized and training becomes unstable. Standard practice has been to use grid extension [1]: start the model at low-G and then progressively increase G, each time re-projecting the activations from the old to the new basis using least-squares. While this can be effective, in part because starting at low G serves as a smoothness inductive bias, ideally one would prefer to spend all of one's training budget on the full resolution model.

Figure 3 shows the effects the curvature penalty can have on training stability for over-parameterized KANs. Compared to the KAN penalty, the curvature

		Test RMSE (median)			$\sum_e \int (\phi_e'')^2$ (median)		
Feynman eq.	Formula	No penalty	KAN	Curvature	No penalty	KAN	Curvature
I.6.20	$e^{-\theta^2/(2\sigma^2)}/\sqrt{2\pi\sigma^2}$	**0.0015**	*0.0022*	0.0034	52.7	305	**17.7**
I.6.20b	$e^{-(\theta-\theta_1)^2/(2\sigma^2)}/\sqrt{2\pi\sigma^2}$	**0.0058**	*0.0075*	*0.0070*	74.8	107	**11.8**
I.9.18	$Gm_1m_2/\left[(\Delta x)^2+(\Delta y)^2+(\Delta z)^2\right]$	**0.0022**	0.0069	*0.0022*	118	$1.2\cdot10^4$	**14.7**
I.12.11	$q\,(E_f + Bv\sin\theta)$	*0.0751*	*0.0720*	**0.0560**	112	99.5	**36.6**
I.16.6	$uv/(1+uv/c^2)$	**0.0061**	*0.0075*	*0.0070*	179	278	**29.0**
I.18.4	$(m_1r_1+m_2r_2)/(m_1+m_2)$	*0.0013*	0.0023	**0.0009**	60.2	480	**29.0**
I.26.2	$\arcsin(n\sin\theta_2)$	**0.0094**	*0.0098*	*0.0136*	84.7	72.3	**19.2**
I.29.16	$\sqrt{x_1^2+x_2^2-2x_1x_2\cos(\theta_1-\theta_2)}$	**0.1256**	*0.1527*	*0.1294*	390	393	**63.1**
I.30.3	$I_0\sin^2(n\theta/2)/\sin^2(\theta/2)$	*0.1787*	*0.1789*	**0.1763**	619	559	**153**
I.50.26	$x_1\,[\cos(\omega t)+\alpha\cos^2(\omega t)]$	**0.0365**	*0.0367*	*0.0450*	194	136	**67.9**
II.11.27	$n\alpha\,\epsilon E_f/(1-n\alpha/3)$	*0.0085*	**0.0076**	*0.0084*	766	1088	**627**
II.35.18	$n_0/[\,e^{\mu B/k_bT}+e^{-\mu B/k_bT}\,]$	**0.0016**	0.0056	*0.0018*	51.5	$2.1\cdot10^6$	**19.6**
III.10.19	$\mu\sqrt{B_x^2+B_y^2+B_z^2}$	**0.0038**	0.0081	*0.0054*	2340	2744	**258**
III.17.37	$\beta\,(1+\alpha\cos\theta)$	**0.0089**	*0.0095*	*0.0161*	95.4	76.3	**20.4**

Table 1: Feynman benchmark, $G = 10$, $n = 5$ seeds, architecture $[d_{\text{in}}, d_{\text{in}}, 1]$. Each penalty is shown at the single λ^* that minimizes the geometric mean of test RMSE across all 14 equations: $\lambda^* = 0.0001$ for the KAN penalty and $\lambda^* = 0.0001$ for the curvature penalty. Winner in each RMSE block bolded per row; non-winners within 2× the winner italicized. Curvature winner bolded per row. The curvature penalty matches or beats the KAN penalty on median RMSE in 9/14 equations and has the lowest total edge curvature in 14/14.

penalty achieves lower test RMSE for all values of λ. At its optimal λ, the curvature-penalized KAN is still improving at the end of 3000 Adam epochs, while the KAN penalty model plateaus after ~ 1000 epochs. While a grid extension run can still achieve lower RMSE [1], the gap is itself informative: it suggests that much of grid extension's effectiveness comes from the implicit smoothness regularization imposed by its low-G early stages, and that the explicit curvature penalty captures a substantial portion of this benefit at fixed high G.

Finally, we study the effects of the penalty for different optimizers and model capacities. Indeed, the preceding results all focus on Adam (default parameters, no weight decay) [23]. Figure 4 shows that the curvature penalty also helps when fitting with L-BFGS [24], another popular optimizer that is often used to train KANs due to its use of curvature (of the loss) information. The best combination—wider ($[2, 5, 1]$) architecture, L-BFGS, and curvature penalty—trains stably at fixed $G = 200$ to test RMSE $\sim 10^{-3}$, where unpenalized L-BFGS at the same G fails catastrophically (RMSE ~ 0.5). Likewise, the curvature penalty helps the KAN scale its capacity correctly: going from narrow to wide architectures ($[2, 1, 1] \to [2, 5, 1]$) improves fit by 10× (Adam) or 88× (L-BFGS). The KAN penalty produces no such benefit.

5 Compositional curvature

Despite the empirical effects shown above, an edge-wise curvature penalty (Eq. 5), which is our focus in this work, is not a curvature penalty for a function composition. Smoothness does not compose separably because the curvature of the composition depends on the curvatures of every layer through the chain rule, which introduces products of Jacobians. In this section, we derive this relationship and

1. observe that KAN layer Hessians are diagonal, yielding a simple per-edge chain-rule decomposition;

2. show that the per-layer penalty is an upper bound on the composition's curvature norm, motivating its use;

3. identify per-edge chain-rule weights that motivate a richer weighted penalty developed in Sec. 6.

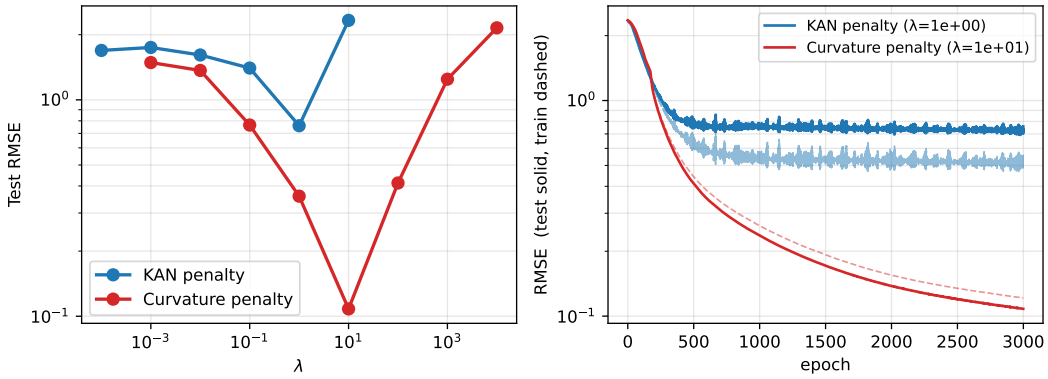

Figure 3: Over-parameterized KANs reach lower test RMSE under the curvature penalty than KAN, across all λ values. Penalty-vs-penalty comparison at high resolution $G = 200$, architecture $[2, 1, 1]$, on the target $f(x, y) = \exp\left(\sin(\pi x) + y^2\right)$. *Left*: final test RMSE as a function of the penalty coefficient λ. *Right*: training trajectories at each penalty's best λ (test RMSE solid, train RMSE dashed).

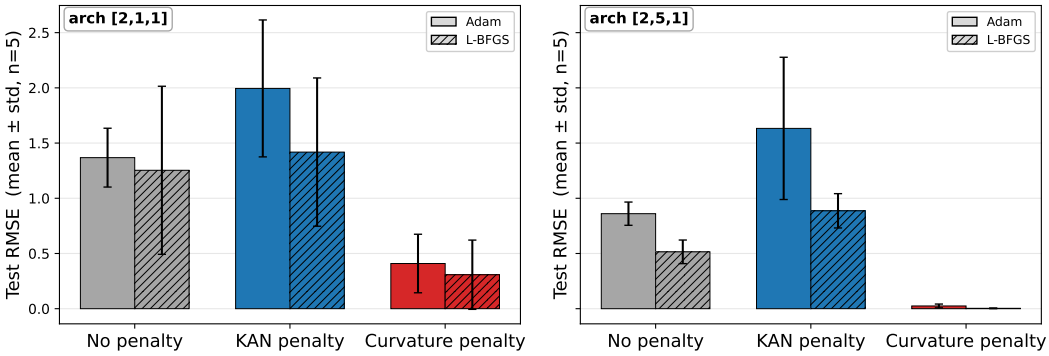

Figure 4: Curvature penalty helps multiple optimizers, improving the fit for $f(x, y) = \exp\left(\sin(\pi x) + y^2\right)$ at high grid resolution ($G = 200$) without grid extension.

5.1 Warmup

For two scalar functions in composition, $f(x) = \phi^{(2)}(\phi^{(1)}(x))$, the second derivative is

$$f''(x) = \phi^{(2)\prime\prime}(\phi^{(1)}(x))\left(\phi^{(1)\prime}(x)\right)^2 + \phi^{(2)\prime}(\phi^{(1)}(x))\phi^{(1)\prime\prime}(x). \quad (6)$$

The first term can give a significant amplification: a smooth $\phi^{(2)}$ applied to a steep (but smooth) $\phi^{(1)}$ can produce a wildly curved composition. The edge-wise penalty misses this amplification, which is the key qualitative reason a (uniform) edge-wise penalty cannot be a tight curvature penalty on f.

5.2 Tensor formulation

Now let $\phi^{(1)} : \mathbb{R}^{n_0} \to \mathbb{R}^{n_1}$ and $\phi^{(2)} : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2}$, and $f = \phi^{(2)} \circ \phi^{(1)}$. Use indices i, j for inputs ($\mathbb{R}^{n_0}$), b and c for the intermediates ($\mathbb{R}^{n_1}$), and a for outputs ($\mathbb{R}^{n_2}$). Write $z = \phi^{(1)}(x)$. Twice differentiating the composition and applying the multivariate chain rule gives

$$\frac{\partial^2 f_a}{\partial x_i \partial x_j} = \sum_{b,c} \left.\frac{\partial^2 \phi_a^{(2)}}{\partial z_b \partial z_c}\right|_{\phi^{(1)}(x)} \frac{\partial \phi_b^{(1)}}{\partial x_i}\frac{\partial \phi_c^{(1)}}{\partial x_j} + \sum_b \left.\frac{\partial \phi_a^{(2)}}{\partial z_b}\right|_{\phi^{(1)}(x)} \frac{\partial^2 \phi_b^{(1)}}{\partial x_i \partial x_j}. \quad (7)$$

This (Faà di Bruno for second-order) is the multivariate analog of Eq. 6.

Extending to multiple layers, we first write the layerwise Jacobian $J_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and Hessian $H_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1} \times n_{\ell-1}}$:

$$(J_\ell)_{ab} = \frac{\partial \phi_a^{(\ell)}}{\partial z_{\ell-1,b}},$$
$$(H_\ell)_{abc} = \frac{\partial^2 \phi_a^{(\ell)}}{\partial z_{\ell-1,b}\, \partial z_{\ell-1,c}}.$$

Then Eq. 7 becomes the matrix Hessian

$$\nabla^2 f_a = J_1^\top H_{2a} J_1 + \sum_c (J_2)_{ac} H_{1c}. \tag{8}$$

(Note that both J_ℓ and H_ℓ are functions of the layer's input $z_{\ell-1}$, with $z_0 = x$; for brevity we suppress evaluation arguments in the multi-layer expressions.) To iterate beyond the first two layers, introduce the upstream and downstream Jacobian products:

$$U_\ell = J_L J_{L-1} \cdots J_{\ell+1}, \quad U_L = I_{n_L}, \tag{9}$$
$$D_\ell = J_{\ell-1} J_{\ell-2} \cdots J_1, \quad D_1 = I_{n_0}. \tag{10}$$

Then by induction on L, we have

$$\nabla^2 f_a = \sum_{\ell=1}^{L} \sum_c (U_\ell)_{ac}\, D_\ell^\top H_{\ell c}\, D_\ell. \tag{11}$$

Each summand contains exactly one Hessian factor; products of Hessians from different layers only appear at derivatives of order ≥ 3.

5.3 KAN's layer Hessians are diagonal

A general vector-valued $\phi^{(\ell)}$ has a full Hessian tensor with $n_\ell n_{\ell-1}^2$ entries. KAN layers, however, take the form

$$\phi_a^{(\ell)}(z) = \sum_b \phi_{ab}^{(\ell)}(z_b), \tag{12}$$

where each activation function $\phi_{ab}^{(\ell)}$ is univariate. Consequently, all mixed second partials vanish:

$$(H_\ell)_{abc} = \delta_{bc}\, \phi_{ab}^{(\ell)\prime\prime}(z_b). \tag{13}$$

The Hessian is *diagonal* in its two input indices.

Plugging into Eq. 11 and collapsing the input pair via the delta, and letting $\Phi''_{\ell c}$ denote the diagonal matrix with entries $(\Phi''_{\ell c})_{bb} = \phi_{cb}^{(\ell)\prime\prime}(z_{\ell-1,b})$, gives

$$\nabla^2 f_a = \sum_{\ell=1}^{L} \sum_c (U_\ell)_{ac}\, D_\ell^\top \Phi''_{\ell c}\, D_\ell. \tag{14}$$

This is a substantial simplification over the analogous expansion for an MLP (where the post-activation Hessian through linear layers does not have this diagonal structure). Each term is a product of three transparent factors:

- $\phi_{cb}^{(\ell)\prime\prime}(z_{\ell-1,b})$: the pointwise curvature of edge $b \to c$ at layer ℓ, evaluated at the activation value $z_{\ell-1,b}$ that the edge sees during the forward pass.
- $(U_\ell)_{ac}$: the upstream sensitivity from node c at layer ℓ to output a; a sum over forward paths.
- $(D_\ell)_{bi}(D_\ell)_{bj}$: the downstream sensitivities from input directions i and j to node b at layer $\ell - 1$; a sum over backward paths, taken twice.

We argue that this clean structure is itself a noteworthy structural advantage of KANs over MLPs for principled smoothness analysis: the chain-rule decomposition stays interpretable (see Sec. 8).

5.4 The edge-wise penalty is an upper bound

Averaging the squared input-Hessian norm over the training distribution gives the composition-level curvature of the function we are fitting:

$$\mathcal{R}(f) := \mathbb{E}_{x \sim p_{\text{data}}} \left\| \nabla^2 f(x) \right\|_F^2. \tag{15}$$

As a stand-alone penalty, however, $\mathcal{R}$ has two drawbacks: it is coupled to the data distribution, evaluating curvature only at inputs the model has seen and leaving the function unconstrained off-data; and it competes with the loss wherever the target itself has curvature, risking over-restriction where expressivity is needed. Even so, $\mathcal{R}$ can be evaluated directly by autograd-through-Hessian or estimated cheaply with a Hutchinson trace estimator [25]—though both

require a double backward. We instead derive a coefficient-space upper bound from Eq. 14 that decouples the prior from the empirical input measure while preserving the chain-rule structure that ties hidden-layer curvature to the composition Hessian.

Substituting the indexed form of Eq. 14 into $\|\nabla^2 f\|_F^2 = \sum_{a,i,j} \left((\nabla^2 f)_{aij}\right)^2$ and applying Cauchy–Schwarz over the $E = \sum_\ell n_\ell n_{\ell-1}$ edges, $(\sum_e t_e)^2 \le E \sum_e t_e^2$, gives

$$\mathcal{R}(f) \;\le\; E \sum_{\ell=1}^{L} \sum_{b,c} \mathbb{E}_x\Big[\phi_{cb}^{(\ell)\prime\prime}(z_{\ell-1,b})^2\, w_{cb}^{(\ell)}(x)\Big], \tag{16}$$

where the data-dependent *path weight* is

$$w_{cb}^{(\ell)}(x) = \sum_a (U_\ell)_{ac}^2 \sum_{i,j} (D_\ell)_{bi}^2 (D_\ell)_{bj}^2. \tag{17}$$

Each edge $b \to c$ at layer ℓ contributes its squared curvature, weighted by the squared upstream sensitivity to the outputs and the squared downstream sensitivities to the inputs.

The right-hand side of Eq. 16 reduces to a coefficient-space penalty under three structural assumptions. For each edge e, let $\Omega_e \subset \mathbb{R}$ be the grid range.

- **(A1)** *Bounded coefficient of variation of the path weight:* $\sigma_{w_e}/\bar{w}_e \le \kappa$ for every edge e, where $\bar{w}_e := \mathbb{E}_x[w_e(x)]$.
- **(A2)** *Bounded density on each spline support:* the density of activation inputs reaching edge e under $x \sim p_{\text{data}}$ is bounded, $p_e(z) \le C/|\Omega_e|$ pointwise on Ω_e, with $C \ge 1$.
- **(A3)** *Knot spacing bounded by 1:* $h_e := |\Omega_e|/G \le 1$ for every edge e.

A1 and A2 are properties of the chain Jacobians and the input distribution, not of training, and both are post-hoc verifiable per edge. A3 is a mild architectural condition—it holds whenever the grid resolution is at least as fine as the input domain extent, which is the regime of any practically-trained KAN.

We now prove this section's main result:

Theorem 1 (Edge-wise penalty bounds composition curvature)**.** *Suppose (A1)–(A3). Then*

$$\mathcal{R}(f) \;\le\; K_\lambda\, R(f), \tag{18}$$

where

$$K_\lambda := 2EC \left(\max_e \gamma_e\right)\left(\max_e \bar{w}_e\right) \frac{G^3}{(\min_e |\Omega_e|)^4},$$
$$\gamma_e := 1 + \kappa\, \sigma_x(\phi_e''^2)/\mathbb{E}_x[\phi_e''^2].$$

Proof. Decompose $\mathbb{E}_x[w_e \phi_e''^2] = \bar{w}_e\, \mathbb{E}_x[\phi_e''^2] + \mathrm{Cov}_x(w_e, \phi_e''^2)$. Bound the covariance by Cauchy–Schwarz under (A1). Lift the data expectation to a function-space integral under (A2). Separate SiLU and spline terms with Young's inequality. Bound the basis curvature Gram via Gershgorin and the partition-of-unity identity ($M_e \preceq h_e^{-3} D_2^\top D_2$), then apply (A3) to absorb h_e^{-3}. Together:

$$\begin{aligned}
\mathcal{R}(f) &\overset{(A1)}{\le} E \sum_e \bar{w}_e\, \mathbb{E}_x[\phi_e''(z_e)^2]\, \gamma_e \\
&\overset{(A2)}{\le} E \sum_e \frac{C\, \gamma_e\, \bar{w}_e}{|\Omega_e|} \int_{\Omega_e} \phi_e''(z)^2\, dz \\
&\le E \sum_e \frac{2C\, \gamma_e\, \bar{w}_e}{|\Omega_e|} \big[\beta_e^2\, c_e^\top M_e\, c_e + K_{\text{silu}}\, \alpha_e^2\big] \\
&\overset{(A3)}{\le} K_\lambda \sum_e \big[\, \|D_2(\beta_e\, c_e)\|^2 + K_{\text{silu}}\, \alpha_e^2\big] \\
&= K_\lambda\, R(f).
\end{aligned}$$

□

Theorem 1 establishes that the edge-wise penalty $R(f)$ from Sec. 3 upper-bounds the composition curvature $\mathcal{R}(f)$ up to a multiplicative constant. Operationally, K_λ folds into the regularization weight λ at training time, so minimizing $R(f)$ minimizes a rigorous upper bound on $\mathcal{R}(f)$.

Remark. Ideally, a function-space smoothness prior should not depend on the training distribution. Equation 15 does, in three ways exposed by the proof of Theorem 1: a joint per-sample product $\phi_e''^2\, w_e$, a marginal density of z_e on Ω_e, and a marginal per-edge importance $\bar{w}_e$. The first lets the optimizer game the penalty by anti-correlating w_e and $\phi_e''^2$

pointwise rather than reducing curvature. The second makes the prior a property of the function's behavior *on the data* rather than its shape on Ω_e, so the activation can be arbitrarily rough on unvisited regions of Ω_e. The third weights each edge's curvature contribution by its average path weight under the data, making the prior's per-edge balance a function of p_{data}; it drops via $\bar{w}_e \le \max_e \bar{w}_e$. This final coupling is the mildest of the three, only a per-edge re-weighting rather than a gameable joint or a blind density. Does retaining $\bar{w}_e$ inside the sum, which targets the edges whose curvature most impacts the composition, give a tighter, more informative penalty than $R(f)$ in practice? We study this empirically in Sec. 6.

6 A richer curvature penalty

The curvature penalty (Eq. 5), by working edge-wise, ignores compositionality. Our derivation of the upper bound reveals a simple affordance for incorporating some of the lost information: retain the $\bar{w}_{e=(\ell,b,c)} = \mathbb{E}_x\big[w_{cb}^{(\ell)}(x)\big]$ terms and use them to compute a weighted edge-wise penalty, in which each edge's contribution scales with its expected impact on the composition Hessian via the chain rule. The one downside to this approach is that it reintroduces the data into the penalty, as the weighting term takes the expectation over the training data. But if that is acceptable, the diagonal structure of KAN Hessians (Sec. 5.3) makes this attractive:

$$w_{cb}^{(\ell)}(x) = \left\|D_{\ell,b,:}(x)\right\|^4 \left\|U_{\ell,:,c}(x)\right\|^2, \tag{19}$$

$$R_{\bar{w}}(f) = \sum_e \bar{w}_e \left(\|D_2(\beta_e\, c_e)\|^2 + K_{\text{silu}}\, \alpha_e^2\right). \tag{20}$$

To test this richer penalty, we do a 10-seed comparison between weighted and uniform edge-wise penalties, first selecting the best λ for each method as the penalty scales will differ. As Fig. 5 shows, the weighted penalty outperforms the uniform penalty in 9 of 10 seeds, yielding a 2.2× lower mean test RMSE (0.0119 ± 0.0086 vs. 0.0260 ± 0.0172; paired Wilcoxon $p = 0.014$).

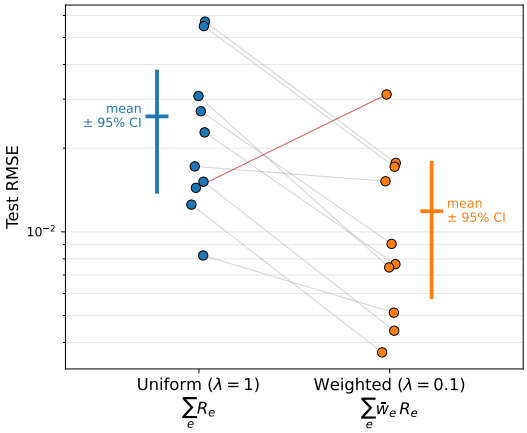

Figure 5: Weighted curvature penalty more than halves the mean test RMSE of uniform, $n = 10$ seeds at $[2, 5, 1]$, $G = 200$ on target $f(x, y) = \exp(\sin(\pi x) + y^2)$. The value of λ for each penalty was identified by a single-seed sweep over $\lambda \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$ followed by a fill at the minimum. CI bars at the sides.

7 Related work

Penalized splines. The Eilers–Marx P-spline framework [20, 21, 22] has 30 years of development including REML-based smoothing parameter selection, mixed-model duality, GAMs with P-splines, tensor-product P-splines for multidimensional smoothing, Bayesian P-splines, and shape-constrained variants. None of this work concerns nested or compositional spline structures.

KAN curvature. Two recent papers reach for second-derivative penalties on KANs without connecting to the penalized-spline literature. Zheng et al. [26] state an objective involving second derivatives of spline coefficients (their eq. 8), but as published the expression is a signed sum without integration or squaring and is not a curvature penalty; we were also unable to match it to their released code. Cang et al. [5], in contrast, also pursue an L^2 curvature penalty. However, their formulation penalizes curvature of the spline component only, not ϕ, and they do not connect to compositional curvature. This leaves the SiLU contribution unconstrained, so α_e can absorb arbitrary curvature over the composition with no penalty cost and the quantity does not

bound $\mathcal{R}$. Our derivation makes these issues explicit, and the form $\|D_2(\beta_e\, c_e)\|^2 + K_{\text{silu}}\, \alpha_e^2$ closes the gap by penalizing the full activation. Rigas *et al.* [27] introduce an adaptive grid method that places spline knots at high-curvature locations, but do not propose a penalty. To the best of our knowledge, no prior KAN work cites Eilers and Marx or draws from the broader penalized-spline literature.

Deep-net first-order smoothness. Drucker and LeCun's "double backpropagation" [28] introduced an input-Jacobian penalty. Subsequent work includes contractive autoencoders [29], Sobolev training [30], and a large literature on Lipschitz constraints [31, 32, 33, 34]. The Lipschitz-of-a-composition-as-product-of-layer-Lipschitz idea is the first-order analog of our second-order chain-rule decomposition. Recent work [35, 36] explores depth-dependent or non-uniform layer-wise Lipschitz allocation.

Input-Hessian penalty: function vs. loss curvature. Curvature penalties based on input-Hessians have been pursued for objectives quite different from ours, and on a different mathematical object than the one our framework targets. Both CURE [37] and CRR [38] penalize the input-Hessian of the *loss*, $\nabla_x^2 \mathcal{L}(f(x), y)$: CURE via finite-difference Hessian-vector products in the gradient direction — motivated by adversarial robustness, since the worst-case perturbation δ maximizes the second-order Taylor term $\delta^\top \nabla_x^2 \mathcal{L}\, \delta$ — and CRR via a curvature-rate measure of the loss landscape for sharpness-aware learning.

We instead penalize the input-Hessian of the *function output*, $\nabla_x^2 f$, which controls the smoothness of the learned function itself, not the response of the loss to input perturbation. The two are mathematically distinct: for regression with mean-squared error, $\nabla_x^2 \mathcal{L} = 2(f - y)\, \nabla_x^2 f + 2\, \nabla_x f\, \nabla_x f^\top$ contains both a function-Hessian piece and an input-Jacobian outer product, and at a perfect fit ($f = y$) reduces to pure first-order sensitivity — carrying no second-order information about f. The objectives also differ: CURE/CRR want f insensitive to input perturbations regardless of what the data demands; our setting wants f to fit smooth structure while preserving the chain rule's role in composing roughness from smooth atomic edges. None of these works derives a chain-rule decomposition of $\nabla_x^2 f$ to motivate per-layer penalty design for compositional architectures.

Loss-Hessian on parameters (sharpness-aware methods). A third, distinct object is the Hessian of the loss with respect to the *parameters* θ. Hessian-trace penalties [39, 40] penalize $\nabla_\theta^2 \mathcal{L}$ for sharpness-aware generalization (flat minima as a generalization proxy). This serves a separate purpose from the input-derivative penalties above and should not be confused with them.

8 Discussion

Do we want smooth activations when training KANs? If all we care about is predictive accuracy, it doesn't matter what the inner functions look like so long as their composition fits the data. Vitushkin's theorem [8] in fact shows that smooth representations need not exist; Samadi *et al.* [4] discuss its implications for KANs. The univariate inner functions are generically non-smooth even for a smooth target function. Smoothness, then, is a modeling choice. Visualizing the activations makes KANs more interpretable than alternatives such as MLPs, and we argue that smooth activations (or at least activations as smooth as possible to fit the data) are the right prior for maximally interpretable KANs. Interpretability may be valuable enough to justify a small accuracy cost. KANs open the black box of deep neural networks, and we should press this advantage.

The curvature penalty stabilizes overparameterized KAN training at fixed high G, without grid extension. A practical consequence is that KAN optimization remains a single end-to-end differentiable run, rather than a multi-stage curriculum with discrete grid changes and optimizer reinitializations. This matters whenever KANs compose with broader systems, such as joint training with other components, neural architecture search, and meta-learning, all of which assume a single optimization problem with stable gradient flow. Online or streaming settings, where one cannot revisit an early low-G phase, likewise rule out grid extension. While multi-stage grid

extension remains effective, the curvature penalty makes single-stage training a viable alternative.

A single structural property of KANs underlies both interpretability and our smoothness analysis: layer outputs are sums of univariate edges, which makes layer Hessians diagonal. This supports both per-edge inspectability (the visualization-based interpretability that motivates KANs) and per-edge attribution of compositional curvature (the principled smoothness penalty we develop here). MLPs have neither property: their dense layers entangle weights and nonlinearities so that curvature emerges from interactions that make attribution difficult. Smoothness is not a feature added to KANs but a property their architecture invites us to control.

Several limitations of our results merit future work. The focus on the edge-wise curvature penalty means variants should be explored more fully, beyond the initial work given in Sec. 6. In particular, it remains open how much of the compositional structure should be folded into the penalty. Another open issue is the treatment of grid spacing and grid updates. We focused on a fixed, uniform grid of knots (constant h_e), yet variable grid spacing, which is needed for techniques such as adaptive grids, requires a penalty that accounts for non-uniform h_e. The penalized-splines literature has also considered penalty norms besides the L^2 form we use. Eilers *et al.* note [22] that an L^1 penalty on second differences can promote piecewise-linear splines over globally smooth ones. Given the success of ReLU, this kink-tolerant variant may prove effective in KANs. On the theoretical side, our treatment leaves room for improvement: tighter bounds on the compositional curvature, for instance, may point toward novel penalties.

In summary, the curvature penalty gives KAN training a single, principled smoothness lever that promotes interpretability, stabilizes high-resolution fitting, and respects the network's compositional structure. Smoothness-aware KAN design is now a tractable problem rather than an aspiration. For an architecture whose appeal rests on interpretability, such advancements may further cement KANs as a credible foundation for interpretable scientific machine learning.

References

[1] Z. Liu, Y. Wang, S. Vaidya, F. Ruehle, J. Halverson, M. Soljačić, T. Y. Hou, and M. Tegmark, "KAN: Kolmogorov-Arnold networks," in *International Conference on Learning Representations (ICLR)*, 2025, arXiv:2404.19756. 1, 2, 4, 5

[2] S. Somvanshi, S. A. Javed, M. M. Islam, D. Pandit, and S. Das, "A survey on Kolmogorov-Arnold network," *ACM Computing Surveys*, vol. 58, no. 2, pp. 1–35, 2025. 1

[3] Z. Liu, M. Tegmark, P. Ma, W. Matusik, and Y. Wang, "Kolmogorov-Arnold networks meet science," *Physical Review X*, vol. 15, no. 4, p. 041051, 2025. 1

[4] M. E. Samadi, Y. Müller, and A. Schuppert, "Smooth Kolmogorov-Arnold networks enabling structural knowledge representation," *arXiv preprint arXiv:2405.11318*, 2024. 1, 10

[5] Y. Cang, Y. H. Liu, and L. Shi, "Can KAN work? Exploring the potential of Kolmogorov-Arnold networks in computer vision," *arXiv preprint arXiv:2411.06727*, 2024. 1, 3, 9

[6] A. N. Kolmogorov, "On the representations of continuous functions of many variables by superposition of continuous functions of one variable and addition," *Doklady Akademii Nauk SSSR*, vol. 114, pp. 953–956, 1957. 1

[7] V. I. Arnold, "On functions of three variables," in *Collected Works: Representations of Functions, Celestial Mechanics and KAM Theory, 1957–1965*. Springer, 2009, pp. 5–8. 1

[8] A. G. Vitushkin, "A proof of the existence of analytic functions of several variables not representable by linear superpositions of continuously differentiable functions of fewer variables," *Doklady Akademii Nauk SSSR*, vol. 156, pp. 1258–1261, 1964. 1, 10

[9] C. J. Vaca-Rubio, L. Blanco, R. Pereira, and M. Caus, "Kolmogorov-Arnold networks (KANs) for time series analysis," in *2024 IEEE Globecom Workshops (GC Wkshps)*. IEEE, 2024, pp. 1–6. 1

[10] M. Kiamari, M. Kiamari, and B. Krishnamachari, "GKAN: Graph Kolmogorov-Arnold networks," *arXiv preprint arXiv:2406.06470*, 2024. 1

[11] X. Yang and X. Wang, "Kolmogorov-Arnold transformer," *arXiv preprint arXiv:2409.10594*, 2024. 1

[12] J. Bagrow and J. Bongard, "Multi-exit Kolmogorov–Arnold networks: enhancing accuracy and parsimony," *Machine Learning: Science and Technology*, vol. 6, no. 3, p. 035037, 2025. 1

[13] ——, "Optimized architectures for Kolmogorov-Arnold networks," *arXiv preprint arXiv:2512.12448*, 2025. 1

[14] S. Rigas, M. Papachristou, T. Papadopoulos, F. Anagnostopoulos, and G. Alexandridis, "Adaptive training of

grid-dependent physics-informed Kolmogorov-Arnold networks," *IEEE Access*, vol. 12, pp. 176 982–176 998, 2024. 1

[15] S. Rigas, F. Anagnostopoulos, M. Papachristou, and G. Alexandridis, "Training deep physics-informed Kolmogorov-Arnold networks," *Computer Methods in Applied Mechanics and Engineering*, vol. 452, p. 118761, 2026. 1

[16] S. Rigas, D. Verma, G. Alexandridis, and Y. Wang, "Initialization schemes for Kolmogorov–Arnold networks: An empirical study," in *International Conference on Learning Representations (ICLR)*, 2026. [Online]. Available: https://openreview.net/forum?id=dwNXKkiP51 1

[17] S. Panahi, M. Moradi, E. M. Bollt, and Y.-C. Lai, "Data-driven model discovery with Kolmogorov-Arnold networks," *Physical Review Research*, vol. 7, no. 2, p. 023037, 2025. 1

[18] N. R. Panczyk, O. F. Erdem, and M. I. Radaideh, "Opening the black-box: symbolic regression with Kolmogorov-Arnold networks for energy applications," *arXiv preprint arXiv:2504.03913*, 2025. 1

[19] J. Bagrow and J. Bongard, "Softly symbolifying Kolmogorov-Arnold networks," *arXiv preprint arXiv:2512.07875*, 2025. 1

[20] P. H. C. Eilers and B. D. Marx, "Flexible smoothing with B-splines and penalties," *Statistical Science*, vol. 11, no. 2, pp. 89–121, 1996. 3, 9

[21] ——, "Splines, knots, and penalties," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 2, no. 6, pp. 637–653, 2010. 3, 9

[22] P. H. C. Eilers, B. D. Marx, and M. Durbán, "Twenty years of P-splines," *SORT - Statistics and Operations Research Transactions*, vol. 39, no. 2, pp. 149–186, 2015. 3, 9, 11

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. 5

[24] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical programming*, vol. 45, no. 1, pp. 503–528, 1989. 5

[25] M. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Communications in Statistics - Simulation and Computation*, vol. 19, no. 2, pp. 433–450, 1990. 7

[26] L. N. Zheng, W. E. Zhang, L. Yue, M. Xu, O. Maennel, and W. Chen, "Adaptive spline networks in the Kolmogorov-Arnold framework: Knot analysis and stability enhancement," in *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, ser. CIKM '25. New York, NY, USA: Association for Computing Machinery, 2025, pp. 4434–4443. [Online]. Available: https://doi.org/10.1145/3746252.3761135 9

[27] S. Rigas, T. Papaioannou, P. Trakadas, and G. Alexandridis, "A dynamic framework for grid adaptation in Kolmogorov-Arnold networks," *arXiv preprint arXiv:2601.18672*, 2026. 10

[28] H. Drucker and Y. LeCun, "Improving generalization performance using double backpropagation," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 991–997, 1992. 10

[29] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio, "Contractive auto-encoders: Explicit invariance during feature extraction," in *Proceedings of the 28th international conference on international conference on machine learning*, 2011, pp. 833–840. 10

[30] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Świrszcz, and R. Pascanu, "Sobolev training for neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, arXiv:1706.04859. 10

[31] C. Anil, J. Lucas, and R. Grosse, "Sorting out Lipschitz function approximation," in *International Conference on Machine Learning (ICML)*, 2019. 10

[32] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," in *International Conference on Learning Representations (ICLR)*, 2018, arXiv:1802.05957. 10

[33] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 10

[34] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing Lipschitz continuity," *Machine Learning*, vol. 110, no. 2, pp. 393–416, 2021. 10

[35] H.-T. D. Liu, F. Williams, A. Jacobson, S. Fidler, and O. Litany, "Learning smooth neural functions via Lipschitz regularization," in *ACM SIGGRAPH 2022 Conference Proceedings*, 2022. 10

[36] J. McGinnis, S. Shit, F. A. Hölzl, P. Friedrich, P. Büschl, V. Sideri-Lampretsa, M. Mühlau, P. C. Cattin, B. Menze, D. Rueckert, and B. Wiestler, "Beyond uniformity: Regularizing implicit neural representations through a Lipschitz lens," in *International Conference on Learning Representations (ICLR)*, 2026. 10

[37] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 10

[38] J. Poschl, "The curvature rate λ: A scalar measure of input-space sharpness in neural networks," *arXiv preprint arXiv:2511.01438*, 2025. 10

[39] Y. Liu, S. Yu, and T. Lin, "Hessian regularization of deep neural networks: A novel approach based on stochastic estimators of Hessian trace," *Neurocomputing*, vol. 536, pp. 13–20, 2023. 10

[40] T. Wu, T. Luo, and D. C. Wunsch, "CR-SAM: Curvature regularized sharpness-aware minimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024, arXiv:2312.13555. 10

[41] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019. 13

[42] Z. Li, "Kolmogorov–Arnold Networks are radial basis function networks," *arXiv preprint arXiv:2405.06721*, 2024. 13

A Methods

All experiments use PyTorch v2.5.1 [41] on CPU. FastKAN ablation in App. B uses the original implementation [42]. Every B-spline KAN edge uses cubic ($k = 3$) splines on a fixed, uniform knot grid of size G, with a SiLU base. Each experiment draws $n_{\text{train}} = 1024$ training and $n_{\text{test}} \in \{256, 1024\}$ test inputs uniformly from the target's domain $\Omega \subset \mathbb{R}^d$, and computes targets in closed form. New samples are drawn for each random seed. Where reported, best λ is selected by a single-seed scout sweep over a range adjusted per penalty type, then re-evaluated with multiple seeds at the winner. Total curvature $\sum_e \int_{\Omega_e} \phi_e''(z)^2\, dz$ is evaluated by trapezoidal integration on a dense grid ($n = 4096$) covering each edge's spline support, with ϕ_e'' computed via the analytic B-spline second-derivative identity for the spline term and the closed-form SiLU second derivative for the base.

Optimization. Adam: lr $= 10^{-3}$, no weight decay, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$. Trained for 3000 epochs with a 200-epoch warmup phase (no penalty during warmup), batch size 256 (64 for the smaller $[2, 2, 1, 1]$ anecdote setup). L-BFGS: lr $= 1.0$, max_iter $= 20$ inner iterations per outer step, history_size $= 100$, strong-Wolfe line search, tolerance_grad $= 10^{-9}$, tolerance_change $= 10^{-12}$. Trained for 500 outer steps without warmup on the full training batch.

B Beyond B-splines

The curvature penalty is basis-agnostic. To see an example beyond B-splines, here we briefly consider an RBF basis as implemented by FastKAN [42]. FastKAN uses a Gaussian radial basis function with uniform centers,

$$B_g(x) = \exp\!\left(-\left(\frac{x - \mu_g}{h}\right)^2\right), \quad \mu_g = \mu_0 + gh, \tag{21}$$

with bandwidth h equal to the center spacing. Each activation has the form

$$\phi(x) \;=\; \sum_g c_g B_g(x) \;+\; \alpha\, \mathrm{SiLU}(x), \tag{22}$$

with no separate spline scale. For a Gaussian RBF with equispaced centers the curvature Gram matrix has closed form:

$$M_{ij} \;=\; \frac{\sqrt{\pi/2}}{h^3}\, e^{-\Delta^2/2}\, \left(\Delta^4 - 6\Delta^2 + 3\right), \tag{23}$$

where $\Delta = j - i$. The diagonal entries are $M_{ii} = 3\sqrt{\pi/2}/h^3$ while the off-diagonal entries decay as a Gaussian in Δ, making M effectively banded.

For the full edge function (Eq. 22), we have curvature

$$\int (\phi'')^2\, dx \;=\; c^\top M\, c \;+\; 2\,\alpha\, c^\top k \;+\; \alpha^2\, K_{\text{silu}}, \tag{24}$$

where k has entries $k_g = \int B_g''(x)\, \mathrm{SiLU}''(x)\, dx$ and $K_{\text{silu}} \approx 0.443$ as before. Here k admits no fully elementary closed form but can be precomputed numerically. However, as before, we drop the cross-terms. (SiLU'' is a bump localized near zero, so $\|k\|$ is small except for those μ_g that overlap the bump, and the cross term is bounded by $2\,|\alpha|\,\|c\|\,\|k\|$.) The penalty for FastKAN comes from summing over all edges (i, j) in all layers:

$$R_{\text{FK}} = \sum_\ell \sum_{i,j} \left[\left(c_{ij}^{(\ell)}\right)^\top M\, c_{ij}^{(\ell)} \;+\; K_{\text{silu}} \left(\alpha_{ij}^{(\ell)}\right)^2 \right], \tag{25}$$

where $c_{ij}^{(\ell)} \in \mathbb{R}^G$ is the basis-coefficient vector for edge (i, j) in layer ℓ.

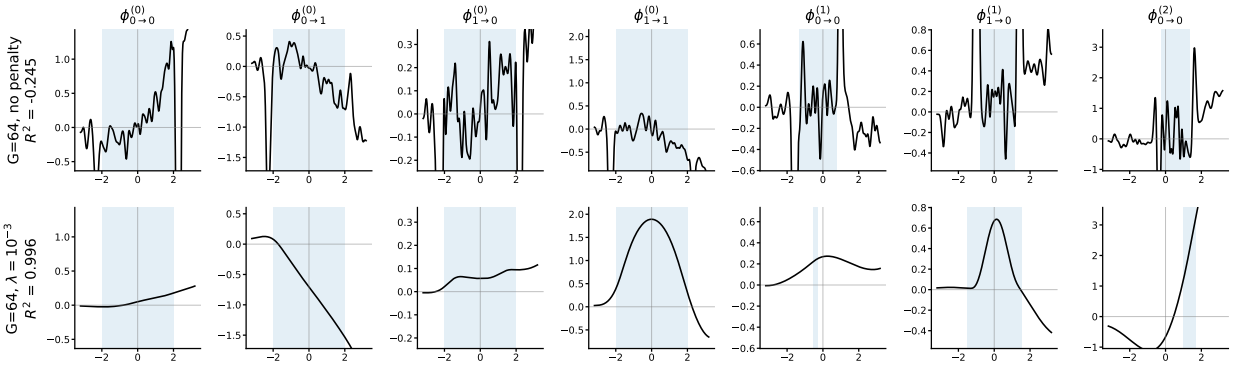

Figure 6: Activation functions for high resolution FastKAN trained on $f(x, y) = \sin\left(x + y^2\right)$ over $[-2, 2]^2$ with architecture $[2, 2, 1, 1]$ and Gaussian-RBF basis size $G = 64$. *Top*: no penalty, $R^2 = -0.24$. *Bottom*: same configuration with our curvature penalty (Eq. 25) at $\lambda = 10^{-3}$, $R^2 = 0.996$. The penalty makes the overparameterized regime trainable, exactly as in the B-spline setting. Shaded regions denote the range of activation inputs.

We applied our penalty to a FastKAN[3] training run with high resolution $G = 64$ on the target $f(x, y) = \sin\left(x + y^2\right)$ over $[-2, 2]^2$ with architecture $[2, 2, 1, 1]$, and visualized the activation functions in Fig. 6. The penalized model had notably smoother activation functions and was substantially more accurate ($R^2 = 0.996$ vs. -0.24). Figure 6, together with our prior results, demonstrates that the curvature penalty is effective across multiple KAN bases.

[3]We disable FastKAN's optional LayerNorm in both conditions: it is incompatible with the singleton $1 \to 1$ layer in $[2, 2, 1, 1]$, and its learnable affine introduces a gauge symmetry that our edge-wise curvature analysis does not account for.