Optimized Architectures for 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

December 13, 2025

Abstract Efforts to improve Kolmogorov–Arnold networks (KANs) with architectural enhancements have been stymied by the complexity those enhancements bring, undermining the interpretability that makes KANs attractive in the first place. Here we study overprovisioned architectures combined with sparsification to learn compact, interpretable KANs without sacrificing accuracy. Crucially, we focus on differentiable sparsification, turning architecture search into an end-to-end optimization problem. Across function approximation benchmarks, dynamical systems forecasting, and real-world prediction tasks, we demonstrate competitive or superior accuracy while discovering substantially smaller models. Overprovisioning and sparsification are synergistic, with the combination outperforming either alone. The result is a principled path toward models that are both more expressive and more interpretable, addressing a key tension in scientific machine learning.

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

1 Introduction

Deep learning has transformed scientific modeling [1, 2, 3], but its advances often work against interpretability [4, 5]. Skip connections [6], DenseNet blocks [7], and deeper architectures improve accuracy by adding complexity. This is precisely what makes models harder to understand. For scientific applications, where insight matters as much as prediction, the tension is acute.

Sparsification offers a way out [8, 9, 10, 11]. Starting from an overprovisioned model, learn which components can be removed without sacrificing (too much) accuracy. When sparsification is differentiable, structure and parameters can be learned jointly [11, 12], avoiding the expense of traditional, discrete architecture search [13, 14].

Recently, Kolmogorov–Arnold networks (KANs) have shown a unique combination of predictive accuracy and interpretability, making them useful for modeling scientific systems [15, 16]. Where standard networks learn weights, KANs learn activation functions (Fig. 1)—a shift that makes individual components inspectable. But KANs face the same tradeoff as standard networks. Overprovisioning improves expressiveness at the expense of interpretability. Again, sparsification is the way out.

In this paper, we show that KANs can be made more expressive without sacrificing interpretability, and often improving it. We combine DenseNet-style forward connections with differentiable ℓ_0 sparsification, allowing overprovisioned KANs to be pruned during training to compact subnetworks. A minimum description length objective provides principled guidance for the tradeoff between accuracy and complexity. Across symbolic benchmarks, dynamical systems modeling, and real-world prediction tasks, the combination typically yields sparser models at comparable or better accuracy than either technique alone.

The rest of this paper is organized as follows. Section 2 provides background on Kolmogorov–Arnold networks, differentiable ℓ_0 sparsification, and DenseNet-style forward connections. Section 3 presents our approach: edge and node gating, the MDL-based learning objective, and their combination with forward connections. Section 4 evaluates models with and without these enhancements. We conclude with a discussion in Sec. 5.

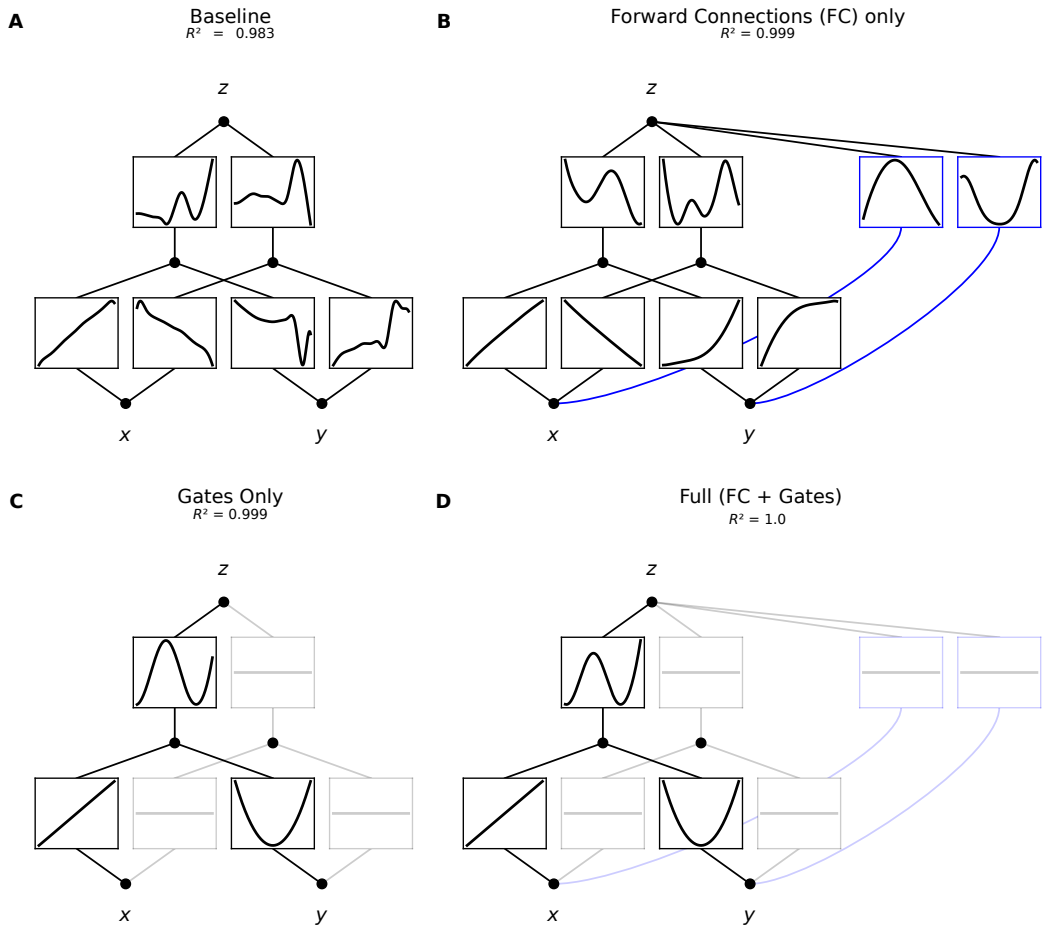

Figure 1: Learning the example function $z = \sin\left(x + y^2\right)$ with Kolmogorov–Arnold Networks (KANs). Forward connections are highlighted in blue.

2 Background

2.1 Kolmogorov–Arnold Networks

Kolmogorov–Arnold Networks (KANs), motivated by the Kolmogorov–Arnold Representation Theorem [17, 18, 19], consist of L layers with shapes $[n_0, n_1, \ldots, n_L]$. The layer update is given by:

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

where $\phi_{\ell ij} : \mathbb{R} \to \mathbb{R}$ is a learnable univariate activation function associated with the edge from neuron i in layer ℓ to neuron j in layer $\ell + 1$. In matrix form, the layer update becomes $\mathbf{x}^{(\ell+1)} = \Phi_\ell(\mathbf{x}^{(\ell)})$, where Φ_ℓ is the functional matrix containing the $\phi_{\ell ij}$'s connecting layers ℓ and $\ell + 1$, and $\mathbf{x}^{(0)} := \mathbf{x}$ is the network input. Composing the layers yields the full network:

$$\mathrm{KAN}(\mathbf{x}) = \left(\Phi_{L-1} \circ \Phi_{L-2} \circ \cdots \circ \Phi_1 \circ \Phi_0\right)(\mathbf{x}). \tag{2}$$

Thus the KAN architecture consists of fully-connected layers of activation functions joined by summation nodes, and the architecture is defined by the shape vector $[n_0 = q, n_1, \ldots, n_L = p]$, where q and p are the dimensions of the input and output, respectively. While summation nodes are theoretically sufficient to represent any function including multiplication, KAN 2.0 [16] introduces explicit multiplication nodes alongside summations that compute products of incoming activations, yielding more compact networks for multiplicative functions.

Many parameterizations of $\phi_{\ell ij}$ have been proposed (see, e.g., [20, 21, 22, 23, 24]); here we use the original B-spline

formulation [15, 16], which combines a B-spline with a fixed, nonpolynomial base function $b(x)$, typically SiLU ($b(x) = x\sigma(x)$):

$$\phi(x) = w_b b(x) + w_s \sum_{m=1}^{G+K} c_m B_m(x), \tag{3}$$

where w_b and w_s are learnable scale parameters, c_m are learnable spline coefficients, and B_m are B-spline basis functions of order K over G grid intervals. The base function $b(x)$ ensures that (2) does not collapse to a high-order polynomial, which would otherwise result from the polynomial B-splines alone.

When KANs were introduced, regularizations of various forms were proposed, mostly to prevent overfitting of the spline terms [15, 16]. Liu et al. also introduced pruning of the network as a sparsification technique [15, 16]. However, this approach pruned activation functions after training, in a post-hoc optimization step. In contrast, in this work we apply a differentiable sparsification process that learns both the KAN parameters and the sparsity (via gating terms) jointly, during training.

2.2 Differentiable ℓ_0 regularization

A classic approach to sparsifying an overprovisioned model is to use the ℓ_1-norm as a proxy for the ℓ_0-norm, as was famously pioneered by LASSO [25]. However, LASSO achieves sparse selection through the closed-form proximal update to the soft-thresholding operator, but in the context of a gradient-based optimization algorithm, ℓ_1 regularization alone would encourage small parameter values, not values of zero.

Louizos et al. [11] introduce a continuous relaxation of an ℓ_0 regularization term, which encourages sparsity in neural network weights (or any predictive model parameters). Their approach possesses two crucial properties: (1) it is differentiable and allows for end-to-end learning, and (2) it encourages true sparsity by setting parameters to exactly zero.

Consider training a predictive model $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$, where $\mathbf{y} \in \mathbb{R}^{n \times p}$ is the target, $\mathbf{x} \in \mathbb{R}^{n \times q}$ is the input, and $\boldsymbol{\theta}$ are the model parameters. Such a model can be trained by learning the $\boldsymbol{\theta}$ that minimizes a loss function regularized by an ℓ_0 penalty that sparsifies $\boldsymbol{\theta}$:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{data}}\big(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i\big) + \beta \|\boldsymbol{\theta}\|_0, \quad \|\boldsymbol{\theta}\|_0 = \sum_{j}^{|\boldsymbol{\theta}|} \mathbb{I}[\theta_j \neq 0]. \tag{4}$$

Now, use a binary gate $z_j \in \{0, 1\}$ to reparameterize each $\theta_j \leftarrow \theta_j z_j$ ($\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} \odot \mathbf{z}$) such that $\|\boldsymbol{\theta}\|_0 = \sum_j z_j$, where $\odot$ denotes elementwise multiplication. We can then learn the $\boldsymbol{\theta}, \mathbf{z}$ that minimizes the loss. Unfortunately, introducing binary gates discretizes the parameter space and makes the problem non-differentiable.

To make this sparse learning problem differentiable, Louizos et al. introduce a continuous relaxation of $\mathbf{z}$ that is then clamped to $[0, 1]$. Specifically, for each gate j, introduce a learnable parameter $\alpha_j \in \mathbb{R}$ and define a stochastic binary variable:

$$s_j = \sigma\left(\frac{\log u - \log(1-u) + \alpha_j}{\tau}\right), \quad u \sim \text{Uniform}(0, 1), \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function and τ is a temperature hyperparameter. Next, stretch and clamp s_j:

$$\bar{s}_j = s_j(\zeta - \gamma) + \gamma, \tag{6}$$

$$\tilde{z}_j = \min(1, \max(0, \bar{s}_j)). \tag{7}$$

To train the model with a gradient-based method such as Adam [26], use the reparameterization trick to differentiably sample $\{z_j\}$ and compute gradient updates to $\{\alpha_j\}$. For the regularization term, the number of open gates, the probability that j is open has a closed form [11],

$$\mathbb{E}[\tilde{z}_j] = \sigma\left(\alpha_j - \tau \log \frac{-\gamma}{\zeta}\right), \tag{8}$$

and the complexity loss $\mathcal{L}_C := \sum_j \mathbb{E}[\tilde{z}_j]$ becomes our proxy for $\|\boldsymbol{\theta}\|_0$ in $\mathcal{L}(\boldsymbol{\theta})$. Finally, at inference, use the following estimator for the trained parameters $\boldsymbol{\theta}^*$:

$$\hat{z}_j = \min\left(1, \max\left(0, \sigma(\alpha_j)(\zeta - \gamma) + \gamma\right)\right), \qquad \boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}^* \odot \hat{\mathbf{z}}. \tag{9}$$

This form of differentiable sparsity has been successfully applied to neural network architectures across areas [11, 27]. It has also been used to optimize Equation Learner Networks (EQLs) [28, 29], a fixed-activation function approach to using

deep learning for symbolic regression [30, 31]. However, it has also been remarked that this approach can exhibit problematic variance, particularly in large networks, and that its hyperparameters can be difficult to tune [27, 32].

2.3 DenseNet Forward Connections

Huang et al. [7] introduce dense blocks, where each layer in the block is connected to every subsequent layer in that block. These *forward connections* (FCs) allow inputs and learned features to be accessible to later layers, operating in a manner similar to ResNet-style skip connections [6] but accumulating across layers. They also provide deep supervision, allowing training gradients to flow directly from later layers to earlier ones. Similar to our work here, forward connections have been used successfully for other forms of neural network-based symbolic regression [31]. When combined with sparsification, FCs can be used for architecture search, as earlier features and inputs are transported directly to the output layer, bypassing the trunk, effectively regulating the depth of the network.

With FCs, the KAN layer update, Eq. (1), becomes

$$\mathbf{x}^{(\ell+1)} = \Phi_\ell \left(\left[\mathbf{x}^{(0)}, \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell)} \right] \right), \tag{10}$$

where the input to layer ℓ is the concatenation of the network input along with all previous layer outputs. The cumulative nature of FCs can result in a potentially large number of activation functions. With so many activation functions, the KAN becomes more difficult to train but more importantly, more difficult to interpret. This motivates the need for a sparsification mechanism (Sec. 2.2) that can prune unnecessary connections while retaining the benefits of FCs.

3 Optimizing KAN architectures

To optimize the architecture of a KAN, we introduce an overprovisioned architecture (Sec. 2.3) along with a sparsification mechanism (Sec. 2.2). A principled loss function based on minimum description length allows for optimization.

To overprovision the architecture, we add forward connections (FCs) between all layers including the inputs. FCs provide deep supervision where gradients can flow directly to earlier layers and they provide the network with more flexibility to learn complex functions and complex compositions of functions. However, this flexibility increases, potentially greatly, the number of activation functions in the KAN. To address this, we introduce gating terms that allow the network to sparsify. In terms of architecture search, because FCs allow for inputs and early features to be accessible to the final output, we can also consider them a form of depth selection: if a learned network sparsifies out most of the trunk in favor of the final FCs, this is equivalent to a network compression that eliminates extraneous layers.

We consider two types of gates, gates on activation functions/edges (egates) and gates on units/nodes (ngates). The latter allows for a group sparsity, although it is a special case of the former. Specifically, for each activation function (edge) $\phi_{\ell ij}(x) \leftarrow \phi_{\ell ij}(x) z_{\ell ij}$, where egate $z_{\ell ij} \in \{0, 1\}$ is relaxed and learned as described in Sec. 2.2. Likewise, each summation unit (node) can be gated with a corresponding ngate, yielding the new layer update terms:

$$x_j^{(\ell+1)} = z_{\ell+1,j} \sum_{i=1}^{n_\ell} z_{\ell ij} \phi_{\ell ij} \left(x_i^{(\ell)} \right), \tag{11}$$

where $z_{\ell+1,j}$ is the ngate on unit j in layer $\ell + 1$[1]. (We distinguish between egates and ngates using the number of indices.) Multiplication units are gated in the same way. In matrix form, this becomes $\mathbf{x}^{(\ell+1)} = \mathbf{z}_{\ell+1}^{(n)} \odot \left(\Phi_\ell \odot \mathbf{Z}_\ell^{(e)} \right) (\mathbf{x}^{(\ell)})$, with $\mathbf{z}^{(n)}$ collecting the ngates and $\mathbf{Z}^{(e)}$ the egates. We see that ngates provide structured or group sparsity [33, 34], which may be beneficial for KANs.

To learn KANs that are both accurate and sparse (parsimonious), following Bagrow and Bongard [35], we use the minimum description length and seek models that minimize the total number of bits required to encode the model and the data given the model:

$$\mathcal{L}_{\text{MDL}} = \mathcal{L}_{\text{model}} + \mathcal{L}_{\text{model|data}}. \tag{12}$$

For data loss, we take the MSE, $\mathcal{L}_{\text{model|data}}\big(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i\big) = (1/n) \sum_{i=1}^{n} \|f(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{y}_i\|^2$, which, assuming normally distributed iid errors, is proportional to the negative log-likelihood of the data given the model. And for model description length, making

[1] In theory we could include gates on inputs and implement variable selection; we save this for future work.

a BIC-style approximation [36], we have

$$\mathcal{L}_{\text{model}} = \frac{\log n}{n} \|\boldsymbol{\theta}\|_0, \tag{13}$$

with the $\|\boldsymbol{\theta}\|_0$ term capturing model complexity based on the number of open gates:

$$\|\boldsymbol{\theta}\|_0 \approx \sum_{\ell=0}^{L-1} \sum_j \mathbb{E}[z_{\ell+1,j}] \left(c_{\ell+1,j} + \sum_i \mathbb{E}[z_{\ell ij}]\, c_{\ell ij} \right) \tag{14}$$

where the $\{\mathbb{E}[z]\}$ are given by Eq. (8) and we take $z_{Lj} = 1$ (no ngates on outputs). Equation (14) introduces optional complexity costs: $c_{\ell j}$ is the complexity cost for node j in layer ℓ, and $c_{\ell ij}$ is the complexity cost for the edge from node i to node j in layer ℓ. For simplicity, in this work we fix $c = 1$ for all nodes and edges, but it may be useful to use $\{c\}$ to weight different parts of the model with different complexities accordingly.

Taken together, our training objective is

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \|f(\mathbf{x}_i; \boldsymbol{\theta}) - \mathbf{y}_i\|^2 + \beta \frac{\log n}{n} \|\boldsymbol{\theta}\|_0, \tag{15}$$

where $\|\boldsymbol{\theta}\|_0$ is given by Eq. (14). For training, we differentiably sample $\{z_{\ell ij}\} \cup \{z_{\ell j}\}$ and proceed per Sec. 2.2. At inference, following Bagrow and Bongard [35], we deterministically threshold gates $\hat{z} = \mathbb{I}[\mathbb{E}[\tilde{z}] > 1/2]$. This differs from Louizos et al. [11] in that the original estimator (Eq. (9)) could still produce values $0 < \hat{z} < 1$ if the corresponding logit has not converged to a value sufficiently large in magnitude, something that for our purposes we wish to avoid.

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

4 Results

To study the effects of gate sparsity and forward connections, we adopt a 2x2 grid experiment, comparing KANs with and without forward connections and with and without gate sparsity. We refer to these four conditions as Baseline, FC Only, Gate Only, and Full. For simplicity, here we consider edge gates (egates) only, noting that node gates (ngates) are a special case of egates and thus edge-level sparsity is the more general mechanism. We leave ngates to future work. For specific details of architectures, other hyperparameters, and training, see Methods (Sec. A).

4.1 Function approximation

We began with an example function, $z = \sin\left(x + y^2\right)$, shown in Fig. 1. While simple, this function tests for composition, as the network must learn to compose sin and $(\cdot)^2$. The non-gated conditions (Baseline and FC Only) fit the function well, not surprising given the performance characteristics of KANs. However, the number of activation functions leads to a pathological decomposition across activations functions; the KANs lack symbolic fidelity [35]. The gated conditions (Gate Only and Full), however, fit the function as well or better, while also yielding activation functions that clearly resemble the underlying expression.

Next, Table 1 shows results on the first ten problems (8 univariate, 2 bivariate) of the Nguyen symbolic regression benchmark [37]. For each problem we generate 1024 training and 256 testing points. We intentionally over-provisioned the KANs with the goal of determining how well we can find smaller, more interpretable subnetworks and whether doing so incurs a performance penalty. In all cases, KANs found highly accurate networks, with test $R^2 \approx 1$. The gated conditions were able to find smaller networks for all problems, with effectively no loss in accuracy. Unlike the results shown in Fig. 1, here we see that FCs were used in all cases of the Full condition. FCs were often, for the simpler problems, the only retained activation functions.

4.2 Forecasting dynamical systems

KANs have shown success modeling dynamical systems [38, 39, 40] so we conducted our 2x2 experiment on two exemplar dynamical systems. The first is the *Ikeda map* [41, 42]:

$$\begin{aligned} x_{n+1} &= 1 + \mu \left(x_n \cos\left(\phi_n\right) - y_n \sin\left(\phi_n\right) \right), \\ y_{n+1} &= \mu \left(x_n \sin(\phi_n) + y_n \cos(\phi_n) \right), \end{aligned} \tag{16}$$

Table 1: Fitting overprovisioned KANs to the Nguyen benchmark. We report test R^2 and active functions (trunk+FC). Architecture: $[n, 5, 5, 5, 1]$ (no multiplication units).

			Baseline		FC Only		Gates Only				Full (FC + Gates)			
							β=0.01		β=0.1		β=0.01		β=0.1	
ID	Expression	Domain	R^2	#Act	R^2	#Act	R^2	#Act	R^2	#Act	R^2	#Act	R^2	#Act
F1	$x^3 + x^2 + x$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	0.9984	22+0	0.9990	12+0	1.0000	0+1	1.0000	0+1
F2	$x^4 + x^3 + x^2 + x$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	1.0000	19+0	1.0000	12+0	1.0000	2+1	1.0000	0+1
F3	$x^5 + x^4 + x^3 + x^2 + x$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	1.0000	28+0	0.9998	12+0	0.9999	1+1	1.0000	0+1
F4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	0.9998	29+0	0.9996	12+0	0.9999	1+1	1.0000	0+1
F5	$\sin(x^2)\cos(x) - 1$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	0.9942	13+0	0.0033	4+0	0.9921	0+11	0.9999	0+1
F6	$\sin(x) + \sin(x + x^2)$	$[-1, 1]$	1.0000	60+0	1.0000	60+46	0.9999	16+0	0.9999	8+0	1.0000	0+1	1.0000	0+1
F7	$\log(x + 1) + \log(x^2 + 1)$	$[0, 2]$	1.0000	60+0	1.0000	60+46	1.0000	19+0	0.9995	8+0	1.0000	0+11	1.0000	0+1
F8	$\sqrt{x}$	$[0, 4]$	1.0000	60+0	1.0000	60+46	0.9997	17+0	0.9978	6+0	0.9987	1+11	0.9999	0+1
F9	$\sin(x) + \sin(y^2)$	$[-1, 1]^2$	1.0000	65+0	1.0000	65+57	0.9996	19+0	0.9999	5+0	1.0000	0+2	1.0000	0+2
F10	$2\sin(x)\cos(y)$	$[-\pi, \pi]^2$	0.9990	65+0	1.0000	65+57	0.9998	40+0	0.9966	18+0	0.9998	3+6	0.9997	2+3

where $\phi_n = 0.4 - 6\left(1 + x_n^2 + y_n^2\right)^{-1}$ and bifurcation parameter $\mu = 0.9$. KANs are well-suited to modeling the Ikeda map due to its compositional nature whereas sparse regression methods such as SINDy [10] struggle [39].

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

$$\begin{aligned} \frac{dN}{dt} &= N\left(1 - \frac{N}{K}\right) - x_p y_p \frac{NP}{N + N_0}, \\ \frac{dP}{dt} &= x_p P\left(y_p \frac{N}{N + N_0} - 1\right) - x_q y_q \frac{PQ}{P + P_0}, \\ \frac{dQ}{dt} &= x_q Q\left(y_q \frac{P}{P + P_0} - 1\right), \end{aligned} \tag{17}$$

where N, P, and Q denote the populations of primary producers, herbivores, and carnivores, respectively, with carrying capacity K acting as the bifurcation parameter. Following [43], we use $K = 0.98$, $x_p = 0.4$, $y_p = 2.009$, $x_q = 0.08$, $y_q = 2.876$, $N_0 = 0.16129$, and $P_0 = 0.5$, which produces chaotic dynamics.

Data for both systems were generated and partitioned into training and testing sets following Panahi et al. [39]. We used architectures of $[2, 4, 4, 4, 2]$ for the Ikeda map and $[3, 3, 3, 3]$ for the ecosystem, consistent with prior work [40, 35].

Table 2 reports the one-step prediction (test) R^2, one-step and multi-step (closed-loop) prediction RMSE, the number of active functions (open egates), and the sparsity. For both systems, most conditions achieve accurate one-step prediction ($R^2 \approx 1$). The ungated conditions achieved higher 1-step accuracy than the gated conditions, but at the cost of far greater complexity. Interestingly, for the Ikeda map, gated conditions achieved higher multi-step accuracy than the baseline at most values of β. However, the FC-only condition achieved the highest multi-step accuracy, but again at the cost of complexity: the Full condition with $\beta = 0.1$ reduced the network size to only 19% the number of functions, while incurring a rise in MS RMSE of 2.3%, a reasonable tradeoff to make in the name of interpretability.

In comparison to Ikeda, KANs applied to the Ecosystem model benefited less from gating. The degree of sparsification was stronger, but often at a greater cost of accuracy. For the shorter training run, the ungated conditions performed best for both 1-step and multi-step prediction, and examining phase plots showed that the networks failed to capture the underlying attractor dynamics. Only if we extended the training run by 50% did we observe gated conditions that captured the underlying dynamics. While the best closed-loop model for the ecosystem was a gated condition (Gates only (15k) at $\beta = 0.0$ achieved MS RMSE = 0.087), it did so with effectively no sparsification. This combined with the very similar performance between Baseline and FC only and we cannot conclude that either gating or FCs provided a meaningful benefit for this system.

Turning from prediction accuracy to sparsity, we observe that in the sparsest Full configurations, forward connections dominate: at $\beta = 0.1$ on the Ecosystem, all three retained edges are FCs (Trunk = 0, FC = 3), indicating the network has learned to bypass intermediate layers entirely. Additionally, consistent with prior findings [35], both systems exhibited self-sparsification, where gates close even without explicit regularization ($\beta = 0$). This effect was more pronounced for the Ecosystem, with Full at $\beta = 0$ retaining 50% or fewer edges.

Table 2: Test performance on dynamical systems. Architectures: [2, 4, 4, 4, 2] (Ikeda map), [3, 3, 3, 3] (Ecosystem). MS: Multi-step.

				RMSE		Active		Sparsity
System	Condition (Epochs)	β	R^2	1-step	MS	Trunk	FC	(%)
Ikeda Map	Baseline (4k)	–	1.0000	0.0009	0.886	48	–	100
	FC Only (4k)	–	1.0000	0.0007	0.852	48	52	100
	Gates Only (4k)	0	1.0000	0.0047	0.868	46	–	95.8
		0.01	1.0000	0.0044	0.879	44	–	91.7
		0.10	1.0000	0.0089	0.862	34	–	70.8
	Full (FC+Gates) (4k)	0	1.0000	0.0020	0.867	44	48	92.0
		0.01	1.0000	0.0031	0.893	10	14	24.0
		0.10	1.0000	0.0042	0.872	8	11	19.0
Ecosystem	Baseline (10k)	–	1.0000	0.0003	0.104	27	–	100
	FC Only (10k)	–	1.0000	0.0002	0.094	27	27	100
	Gates Only (10k)	0	1.0000	0.0025	0.164	26	–	96.3
		0.01	0.9964	0.0153	0.203	16	–	59.3
		0.10	0.8796	0.0781	0.197	9	–	33.3
	Full (FC+Gates) (10k)	0	1.0000	0.0011	0.145	9	18	50.0
		0.01	0.9996	0.0037	0.213	2	9	20.4
		0.10	0.9980	0.0110	0.207	0	3	5.6
	Baseline (15k)	–	1.0000	0.0004	0.162	27	–	100
	FC Only (15k)	–	1.0000	0.0001	0.160	27	27	100
	Gates Only (15k)	0	1.0000	0.0009	0.087	26	–	96.3
		0.01	0.9967	0.0146	0.198	16	–	59.3
		0.10	0.8794	0.0781	0.196	9	–	33.3
	Full (FC+Gates) (15k)	0	1.0000	0.0011	0.138	7	18	46.3
		0.01	0.9996	0.0037	0.224	2	9	20.4
		0.10	0.9980	0.0111	0.161	0	3	5.6

4.3 Real-world data

Lastly, we applied our 2x2 experiments to two real-world datasets:

Compressive strength of concrete The task is to predict compressive strength (in MPa) of concrete from sample properties. The dataset comprises 1030 samples with eight features: cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate (all measured in kg/m^3), and age in days. Since compressive strength is known to depend on the water-to-cement ratio [44], we included this as a derived feature. We also included total binder (cement plus slag and fly ash), total aggregate (coarse plus fine), and the water-to-binder ratio, which extends the water-to-cement relationship to account for supplementary cementitious materials. Because strength gain follows an approximately logarithmic relationship with curing time [44], we additionally include log(age + 1) and sqrt(age) as transformed age variables. An 80/20 train/test split was used for modeling. Data were collected by Yeh [45, 46].

Critical temperature of superconductors The task is to predict the critical temperature T_c (in K) of superconductors from their material properties. The original dataset contains numerous features and derived statistics; following prior KAN modeling [40], we focus on five representative features that capture composition, electronic structure, and bonding: number of elements, weighted mean valence, valence entropy, weighted mean first ionization energy, and mean electron affinity. We drew 1000 samples each for training and testing, from Japan's National Institute for Materials Science superconductor database [47, 48].

As shown in Table 3, gates and FCs improved predictive accuracy while also reducing KAN size for both datasets. For Concrete strength, Full at $\beta = 0.01$ achieved a 15.6% reduction in test RMSE compared to Baseline (4.25 MPa compared to 5.04 MPa) while using only 64 activation functions versus Baseline's 351. Likewise, for Superconductor, the Full model at $\beta = 0.1$ reduced RMSE by 8% versus Baseline (17.83 K compared to 19.38 K) with 22 activation functions compared to Baseline's 80 activation functions. FC only worsened performance for both datasets, while Gates only improved performance and reduced KAN size, but not to the extent of Full. We note that both datasets were also sensitive to the choice of β; for

Table 3: Gates and FCs improved predictive accuracy and reduced KAN size. Architectures: $[13, 13, 13, 1]$ (Concrete), $[5, 5, 5, 5, 1]$ (Superconductor).

			Test		Active		Sparsity
Dataset	Condition	β	R^2	RMSE	Trunk	FC	(%)
Concrete strength	Baseline	–	0.9015	5.04 MPa	351	–	100
	FC Only	–	0.8951	5.20 MPa	351	195	100
	Gates Only	0.01	0.9122	4.76 MPa	73	–	20.8
		0.1	0.8127	6.96 MPa	18	–	5.1
	Full (FC + Gates)	0.01	0.9302	4.25 MPa	49	15	11.7
		0.1	0.8214	6.79 MPa	0	7	1.3
Superconductor	Baseline	–	0.6811	19.38 K	80	–	100
	FC Only	–	0.5784	22.28 K	80	90	100
	Gates Only	0.01	0.6719	19.66 K	63	–	78.8
		0.1	0.6864	19.22 K	14	–	17.5
	Full (FC + Gates)	0.01	0.6359	20.71 K	55	14	40.6
		0.1	0.7301	17.83 K	17	5	12.9

Concrete a smaller value was necessary than for Superconductor, and suboptimal β harmed model performance for both gated conditions. In practice, then, it is likely necessary to tune β using a validation set. This is expected and standard practice in machine learning, and should be considered a small price to pay for such improvements.

5 Discussion

We investigated how KANs can be made more expressive without losing interpretability through the combination of DenseNet-style forward connections and differentiable ℓ_0 sparsification. Forward connections alone increased capacity but also complexity; gating alone reduced size but the combination condition (Full) achieved the best tradeoff, greater sparsity at comparable or superior accuracy. The synergy arises because FCs provide alternative pathways that allow aggressive pruning of the trunk: for simpler problems, such as Nguyen, FCs dominate entirely (Table 1). Yet complex tasks still required participation of the trunk, signaling that the sparsification process can judiciously retain important pathways in both the trunk and FCs. Of course, complex tasks also required tuning β. The optimal β varied across tasks and should be tuned via validation—a standard hyperparameter search representing a small price for interpretability gains. Lastly, as noted by Bagrow and Bongard [35], we also observed self-sparsification, at $\beta = 0$, suggesting implicit regularization benefits from the gating mechanism itself.

We note some limitations of our study. Edge-level gating (egates) is the more general mechanism, but node-level gating (ngates) may offer benefits for structured or group sparsity that we did not fully explore here. For some systems (e.g., Ecosystem), achieving both high sparsity and accurate multi-step prediction remained challenging, suggesting limits to how far models can be compressed for complex dynamics. The differentiable ℓ_0 approach can exhibit hyperparameter sensitivity and variance issues that may destabilize training [27, 32]. Whether these played a role in the Ecosystem results remains open for investigation. Future work could also explore mitigation strategies or other improvements to differentiable sparsification.

Beyond improvements to differentiable sparsification, our results suggest several directions for future work. As noted in Sec. 3, gating inputs could allow KANs to do variable selection, letting them scale better to problems with large numbers of predictors. Combining edge gates, node gates, and multi-exit architectures [40] would enable simultaneous optimization of width, depth, and connectivity, giving a more complete form of architecture search. Likewise, it may be worth comparing to different gating mechanisms, such as categorical gates [49, 50]. Complexity weights in the MDL objective (the $c_{\ell ij}$ terms) could be tuned to reflect computational cost or preferences for certain structures. Finally, work on better "symbolifying" KAN activation functions [35] could naturally combine with these architectures, allowing for more reliable discovery of symbolic expressions, particularly with complex compositions.

We have shown that overprovisioning KANs with forward connections, combined with differentiable sparsification guided by MDL, produces compact architectures without sacrificing predictive performance. Overprovisioning and sparsification were synergistic, with the combination typically outperforming either component alone. Differentiable sparsification transforms architecture search from a discrete hyperparameter choice into a learnable component of training, requiring only standard gradient-based optimization. More broadly, these results offer a principled mechanism to effectively incorporate future ad-

vancements in deep learning into the interpretable domain of scientific machine learning.

A Methods

KAN networks were implemented in PyTorch v2.8.0. Splines used $G = 10$ grid intervals and degree $K = 3$ for all experiments; grid refinement was not used. Spline grids and coefficients were initialized as described in Liu et al. [15]. Unless otherwise noted, spline grids were updated 10 times during the first 50 epochs.

For the differentiable gates, we used temperature $\tau = 2/3$ and stretch parameters $\gamma = -0.1$, $\zeta = 1.1$ throughout. Unless otherwise noted, gates were initialized with $\alpha_i = -1$. For baseline conditions (Baseline and FC only), all gates were fixed open with $\alpha_j = 20$, and α_j gradients disabled, and $\beta = 0$.

Training used Adam [26] with default parameters and a constant learning rate of 10^{-3} for both coefficients and gates. Unless otherwise noted, models were trained with a batch size of 128 and a warm-up period of 200 epochs with $\beta = 0$. Forward connections, if present, were warmed up for 100 epochs after the warm-up period of the trunk. Early stopping, if used, terminated training when gate decisiveness [35] exceeded 0.99, with patience $\min(500, 0.05 \times \#\,\text{epochs})$.

Code will be made available upon publication.

Function approximation (Sec. 4.1, Table 1)— We generated 1024 training and 256 test points on $x, y \in [-2, 2]$, with target $z = \sin(x + y^2)$. Models used architecture $[2, 2, 1]$ and trained for 3000 epochs with batch size 64 and $\beta = 0.2$. warm-up and early stopping were not used. For the Nguyen benchmark, we evaluated the first 10 Nguyen problems [37], generating 1024 training and 256 test points per problem. Models were trained for 10k epochs.

Dynamical systems (Sec. 4.2, Table 2)— For models of both systems, gates were initialized with $\alpha_j = -2$, and spline grid updates and early stopping were not used. For the Ikeda map, we used architecture $[2, 4, 4, 4, 2]$. For the ecosystem, we used architecture $[3, 3, 3, 3]$. Models for Ikeda map were trained 4k epochs; for the ecosystem, models were trained for 10k and 15k epochs. The data generating process and parameter values are given in Sec. 4.2 and follow Panahi et al. [39].

Real-world data (Sec. 4.3, Table 3)— For concrete compressive strength prediction task, we used the UCI concrete dataset [45, 46] 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 a $[13, 13, 13, 1]$ KAN architecture. For superconductor critical temperature prediction, we used 5 features from the UCI superconductor dataset [47, 48]: 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 used a $[5, 5, 5, 5, 1]$ architecture. These architectures were deeper than those previously used [40] to test sparsification of overprovisioned KANs. Models for both tasks were trained for 5000 epochs with batch size 64 and 500 warm-up epochs.

References

[1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 1

[2] 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

[3] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023. 1

[4] 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

[5] 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

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 4

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1, 4

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

[9] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989. 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, 6

[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, 5

[12] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations*, 2019. 1

[13] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 1

[14] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21, 2019. 1

[15] 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, 3, 9

[16] 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

[17] 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

[18] 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

[19] 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

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

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

[22] 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

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

[24] 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

[25] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, December 2018. ISSN 0035-9246. doi: 10.1111/j.2517-6161.1996.tb02080.x. URL https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. 3

[26] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 3, 9

[27] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019. 3, 4, 8

[28] Georg Martius and Christoph H Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, 2016. 3

[29] 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. 3

[30] Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE transactions on neural networks and learning systems*, 32(9):4166–4177, 2020. 4

[31] Michael Zhang, Samuel Kim, Peter Y. Lu, and Marin Soljačić. Deep learning and symbolic regression for discovering parametric equations. *IEEE Transactions on Neural Networks and Learning Systems*, 35(11):16775–16787, 2024. 4

[32] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021. 4, 8

[33] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006. 4

[34] Junzhou Huang and Tong Zhang. The benefit of group sparsity. *The Annals of Statistics*, 38(4):1978 – 2004, 2010. doi: 10.1214/09-AOS778. URL https://doi.org/10.1214/09-AOS778. 4

[35] James Bagrow and Josh Bongard. Softly symbolifying kolmogorov-arnold networks. *arXiv preprint arXiv:2512.07875*, 2025. 4, 5, 6, 8, 9

[36] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. ISSN 00905364, 21688966. 5

[37] 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, 9

[38] Benjamin C Koenig, Suyong Kim, and Sili Deng. KAN-ODEs: Kolmogorov–Arnold network ordinary differential equations for learning dynamical systems and hidden physics. *Computer Methods in Applied Mechanics and Engineering*, 432:117397, 2024. 5

[39] 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. 5, 6, 9

[40] James Bagrow and Josh Bongard. Multi-exit kolmogorov–arnold networks: enhancing accuracy and parsimony. *Machine Learning: Science and Technology*, 6(3):035037, aug 2025. 5, 6, 7, 8, 9

[41] 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

[42] 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

[43] Kevin McCann and Peter Yodzis. Nonlinear dynamics and population disappearances. *The American Naturalist*, 144(5): 873–879, 1994. 6

[44] Adam M. Neville. *Properties of Concrete*. Pearson, 5th edition, 2011. 7

[45] 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. 7, 9

[46] 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. 7, 9

[47] Kam Hamidieh. A data-driven statistical model for predicting the critical temperature of a superconductor. *Computational Materials Science*, 154:346–354, 2018. 7, 9

[48] Center for Basic Research on Materials. MDR SuperCon datasheet ver.240322. 7, 9

[49] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=rkE3y85ee. 8

[50] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=S1jE5L5gl. 8